SEPARATING EXPONENTIAL TIME CLASSES FROM POLYNOMIAL TIME CLASSES

A Thesis

Presented to the Faculty of the Graduate School

of the College of Computer Science

of Northeastern University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Sarah Easton Mocas

October 22, 1997

Acknowledgments

I would like to thank both Steve Homer and Luc Longpré who have jointly advised me throughout the preparation of this thesis. Especially I thank Steve who has served as my primary advisor with patience and most importantly a continued willingness to explain. The extent to which Steve was available to discuss even my most outlandish attempts constantly motivated me to work harder. I have thoroughly enjoyed working with Steve and appreciate his steady, experienced direction. In addition, the results in Chapter 4 were done jointly with Steve. I thank Luc particularly for guiding me through the writing of my first paper, which we did jointly, and for his intuitive explanations of almost any proof. His ability to simplify a complex subject was invaluable.

I would like to thank Agnes Chan and Karl Lieberherr for serving on my dissertation committee. Additionally, I thank Agnes for being a friend and counselor to myself and the graduate community at Northeastern in general.

Alan Selman introduced me to complexity theory and gave me an invaluable background during the first few years of my graduate studies. As my advisor during this time period he taught me many of the basic techniques and instilled in me a sense of the history of complexity theory. I thank him for both his academic and financial backing.

I thank George Smith, who, at Tufts University, introduced me to computational theory and passed his love of the subject on to me.

The members of the joint Boston University and Northeastern University Complexity Theory Seminar also have my gratitude. A few of the people who have made this seminar especially stimulating and enjoyable are Steve Fenner, Fred Green, Harry Buhrman, Zhixiang Chen, Krishnamurthy Ganesan, Jie Wang, Roy Rubinstein, Judy Goldsmith and Andy Klapper. With particular thanks to Harry and Judy for their lively discussions.

In addition, a special thanks goes to Cynthia Brown for her friendship and understanding. I owe a great deal to both Cindy and Arlin Brown for helping me gracefully go through this process.

With deepest gratitude I thank Swami Chetanananda. His continued encouragement and support in my work and life are immeasurable and invaluable.

Sarah Mocas

Boston, August 1993

Contents

List of Figures

Abstract

This thesis examines several of the most central and fundamental complexity classes. These classes are defined by polynomial and exponential time bounds both uniform and nonuniform. Showing classes to be distinct or separate has been a long standing objective in structural complexity theory. It is this objective that we address. Specifically we are interested in separating classes in the exponential time hierarchy, $EXPH$, from classes in the polynomial time hierarchy, PH. We show that, for any fixed integer c, $P^{NP[O(n^c)]} \subsetneq NEXP$. This improves a previous result by Fu, Li and Zhong. Further we generalize this separation to related levels of PH and $EXPH$ showing that, for any fixed integer c and $i \geq 1$, $\Delta_i^{P[O(n^c)]} \subsetneq \Sigma_{i-1}^{EXP}$.

There is also an interest in separating exponential time classes from classes of sets which are nonuniformly computable in polynomial time. By considering polynomial advice classes we show that $EXP \not\subseteq DTIME(2^{O(n^{c_1})})/n^{c_2}$ for fixed integers c_1 and c_2. This implies, for example, that $EXP \not\subseteq E/lin$.

Usually complexity theory is concerned with questions of set membership. An alternative is to allow a model which computes a partial function and outputs a value if one exists. In this way the time and space complexity of classes of partial functions is studied. We show that our results relating both uniform and nonuniform exponential and polynomial classes are true for the corresponding classes of function. Further we show that $PF^{NP[\log]} \subsetneq EXPF_{PB}$. This proof is then generalized to show that $PF^{\Sigma_i^P[\log]} \subsetneq EXPF_{PB}$, for $i \geq 1$. Neither of these results is known for the corresponding classes of sets and can not be shown using proof techniques which relativize. Also we note that, for $i \geq 1$, $PF^{\Sigma_i^P[\log]} \subseteq PF^{NP}$ unless $P = NP$ which demonstrates that the structure of the polynomial hierarchy over function classes is very different from the polynomial hierarchy over sets unless PH collapses to P^{NP}.

Chapter 1

Introduction

1.1 Computation and Complexity

Gödel showed that no consistent system of logic could describe proofs of all true assertions of arithmetic [Göd31]. This left as a question: Is there a method, or mechanical process, which applied to a mathematical statement would give an answer as to whether or not it was provable? Working on this question, both Turing and Church independently answered in the negative by outlining rigorous system which additionally isolate what has come to be accepted as the computable functions [Tur36, Chu36]. Turing specifically developed a model for computation, the Turing machine, which gave the theoretical underpinnings of the computer. So from a very deep philosophical question, which lead to a series of truly landmark theoretical papers, came an understanding of the theory of today's computer.

It is not enough just to examine the nature of computation but one is also interested in what can be computed in a reasonable world. Turing's theoretical work, after all, lead to the construction of one of the most widely used machines today so, likewise, the practical limits of these machines is of interest. Alas, a very practical question again leads to an abstract theory, computational complexity theory.

Thus the study of Turing's model under the restriction of a reasonable amount of computation time and memory space entered the picture.

One of the early steps in the study of resource-bounded computation was to define a restricted model. This was succinctly done by Hartmanis and Sterns [HS65]. As Richard Karp [Kar86] explains:

> ... but it is the 1965 paper by Juris Hartmanis and Richard Sterns that marked the beginning of the modern era of complexity theory. Using Turing machines as their model of an abstract computer, Hartmanis and Sterns provide a precise definition of the "complexity class" consisting of problems solvable in a number of steps bounded by some given function of the input length. ... we now had a satisfactory formal framework for pursuing the questions that Edmonds had raised earlier in an intuitive fashion - questions about whether, for instance, the traveling salesman problem is solvable in polynomial time.

Other early papers that contributed to the discussion of resource-bounded computation are [Edm65, Rab59, Rab60, Rit63]. This brief history leaves out many important references and is intended only to give a broad outline of the development of structural complexity theory.

Intuitively, we would like to know if a particular problem is computable in a certain amount of time or space. This intuition translates into an examination of classes of problems which can be shown to have common structural properties indicating that they will be solvable using the same amount of time or space. A major contribution towards this end was made in Cook's paper [Coo71] where he laid the foundation for the theory of NP-completeness. Independently, Levin obtained similar results [Lev73]. As a part of Cook's work he established the importance of polynomial time many-one reductions as a tool for showing that two problems require the same amount of time. Subsequently, a paper by Karp [Kar72] showed that many interesting problems are NP-complete, establishing NP as a complexity class of immense interest.

A primary focus of structural complexity theory is showing that a class contains a language which in fact can not be computed with less resources. In this way complexity classes are shown to be distinct or separate. Intuitively, increasing the amount of resources should allow us to recognize more languages. In fact various separation results are known between nondeterministic time classes [Coo73, SFM73, BG70] and between deterministic time classes [HS65]. Separations between nondeterministic and deterministic classes of the same time complexity have not been proved. This thesis continues this line of inquiry specifically looking at the possible separation of exponential time-bounded classes from both uniform and nonuniform polynomial time-bounded classes.

1.2 Exponential Time

Computational complexity theory is the study of resource-bounded computation. Two of the most commonly studied resources are the time and space needed to recognize the elements of a set or language. The complexity classes P and NP, polynomial time-bounded deterministic and nondeterministic computation respectively, have been a major focus of this study as they are known to contain many practical problems [GJ79]. Although there are many open questions concerning P and NP which are considered to be of great practical interest, these questions have thus far resisted solutions and many of them have been shown, via relativization, not to be solvable with many of the current techniques used in the field. Often questions concerning P and NP can be related to questions about other complexity classes. As a result, classes which appear to have less practical value are studied in hopes of better understanding P, NP and resource-bounded computation in general.

Problems for which there is no known solution in polynomial time; i.e., no known efficient solution, are considered to be intractable [1]. These problems have been broadly categorized as exponential time problems even though this includes problems in subexponential classes such as $TIME(n^{\log n})$ [GJ79]. There are natural relationships between many of the structural properties of P, NP, the classes in the polynomial time hierarchy, PH, and related exponential time classes. For example, it is known that if higher deterministic and nondeterministic classes are not equal then there are corresponding lower classes that are also not equal. In many cases showing that there exists a relationship between higher and lower classes will indicate that resolving a question for exponential time is as hard as resolving a related question for polynomial time.

Still, as will be discussed, the structure of exponential time classes does not always mirror that of related polynomial time classes. Specifically, some of the techniques which are used to demonstrate results in the polynomial time hierarchy do not generalize to exponential time hierarchies. For one, it has not been shown that general downward separating results which apply to the polynomial time hierarchy can be duplicated in natural exponential time hierarchies. This forces us to examine more closely the semantics of oracle access.

Also the structure of the exponential time classes is of interest on its own. It appears that many of the techniques from recursive function theory, which fail to give positive results when applied to polynomial time classes, in fact can be used to establish positive results about exponential time classes [KMR90]. So in some respects, more is known about exponential time.

Exponential time is also of interest as natural complete sets for EXP are known.

[1]In different parts of the literature problems in both P, deterministic polynomial time, and BPP, the class of languages recognized by polynomial time probabilistic Turing machines whose error is bounded above by some positive constant $\epsilon < 1/2$, have been considered tractable.

In [SC78] it is shown that some combinatorial games are complete for EXP. Also the "circularity problem for attribute grammars" is complete for EXP [JOR75] and in [EPS87] a specific class of attribute problems is shown to be equal to EXP.

In the process of studying exponential time classes and their relationship to polynomial time classes several questions have come to the forefront. Specifically, the following questions have directly motivated different chapters of this thesis.

1. Is P^{NP} properly contained in $NEXP$?
2. Is P^{NP} properly contained in P^{NEXP}?
3. Is BPP properly contained in EXP?
4. Is $EXP \not\subseteq P/poly$?
5. Does $co-NEXP \subseteq NEXP/poly$ imply that the exponential time hierarchy collapses?
6. Can the structure of function classes be used to show separations between classes of sets?

None of these questions has been completely answered but a discussion of each is presented. It turns out that questions 1 and 2 are equivalent. These questions are the focus of Chapter 3. Questions 3 and 4 are related as every language in BPP is in $P/poly$. Chapter 4 examines these questions. Chapter 5 examines question 5. Lastly, Chapter 6 deals with relating polynomial time-bounded function classes to exponential time-bounded function classes.

It should be noted here that there are several different ways to define exponentially time-bounded classes. In this thesis EXP will be used to refer to the classes $DTIME(2^{p(n)})$, where p is a polynomial and $NEXP$ will be used to refer to the classes $NTIME(2^{p(n)})$. Likewise, E, (NE) will be used to refer to the classes

$DTIME(2^{cn})$, $(NTIME(2^{cn})$ respectively), where c is a constant. Definitions will be presented in Chapter 2. The following containments are known:

$$P \subseteq NP \subseteq PH \subseteq PSPACE \subseteq EXP \subseteq NEXP.$$

The following containments are proper [HS65, HLS65, Coo73]:

$$P \subsetneqq E \subsetneqq EXP \text{ and } NP \subsetneqq NE \subsetneqq NEXP.$$

Even though $PH \subseteq PSPACE \subseteq EXP$, it is not generally believed that $PSPACE$ is a subset of E or even that the polynomial hierarchy is contained in E. It is known that $PSPACE \neq E$ and that $PH \neq E$ as both $PSPACE$ and PH are closed under polynomial time many-one reductions and E is not. However, it is not known if either $PSPACE$ or PH is contained in E.

1.3 Overview of Techniques and Results for Exponential Time

Previous results concerning exponential time classes and their relationship to complexity classes in the polynomial time hierarchy will now be presented. This history is intended to highlight results and specific techniques that have been used in the study of exponential time classes and is not comprehensive.

1.3.1 If $P = NP$ and Other Assumptions

In the study of complexity classes the technique of adding any easily recognizable suffix to each word in a language, known as *padding* a language, is used to force a language that is "hard" to recognize to be easier to recognize. Because the length of input words is increased by the pad, and since the pad can be easily removed, the time complexity of a machine recognizing a padded language can be less than that of one recognizing the unpadded language. This technique proves to be very useful in showing relationships between classes in the polynomial time hierarchy and related exponential time classes. For example, Book used padding arguments to show that if $P = NP$, then for every time constructible function f, $DTIME(2^{O(f)}) = NTIME(2^{O(f)})$ and $DTIME(f^{O(1)}) = NTIME(f^{O(1)})$ [Boo74a]. Consequently, if $P = NP$ then both $EXP = NEXP$ and $E = NE$. Book also showed via padding that if $P = PSPACE$ then, for every time computable function f, $DTIME(2^{O(f)}) = DSPACE(2^{O(f)})$ and $DTIME(f^{O(1)}) = DSPACE(f^{O(1)})$ [Boo74a]. So $P = PSPACE$ implies $EXP = EXPSPACE$. In general, padding arguments have been used to show that if higher deterministic and nondeterministic classes are not equal, then there are corresponding lower classes which are not equal. These are called *downward separation* results.

Another method which exploits increasing the length of strings is the use of tally sets. A tally set is a set over a one symbol alphabet. For example, if A is any language over $\{0,1\}^*$, then $tally(A)$ is defined to be $\{0^n \mid n \in A\}$. Notice that the length of 0^n is exponential in the length of n. It is known that A is a member of E, (NE) if and only if $tally(A) \in P$, $(NP$ resp.) and that $E \neq NE$ if and only if there is a tally set in $NP - P$ [Boo74b]. Along this same line, *sparse sets*, sets with at most a polynomial number of strings of length n for any n, have been studied. In [HIS85], Hartmanis, Immerman and Sewelson explored the relationship between sparse sets and lower and higher complexity classes. They show, in part via an *upward separation* method, that there exists a sparse set in $NP - P$, $PSPACE - NP$ or $PSPACE - P$ if and only if, respectively, $NE \neq E$, $ESPACE \neq NE$, and $ESPACE \neq E$.

1.3.2 With No Assumptions

By the time hierarchy theorems it is know that $P \neq EXP$ and $NP \neq NEXP$ [HS65, Coo73, BG70]. Surprisingly, there have not been many results which improve these separations, especially for deterministic classes. The proper containment of NP in EXP is an open question and, as will be discussed, it appears to be difficult to prove. In this thesis we show that $P^{NP[n^c]} \subsetneq NEXP$ for any integer c. In general, showing further separation of EXP and $NEXP$ from polynomial time classes is considered difficult [2].

A related result by Kannan separates the second level of the exponential time hierarchy, $NEXP^{NP}$, from the polynomial time probabilistic class BPP. This result will be discussed in Section 1.3.5.

1.3.3 What Oracles Say

An oracle Turing machine is a Turing machine with an additional oracle tape and 3 additional types of states, QUERY, YES and NO. If an oracle TM enters a QUERY state with a string w written on the query tape, then if w is in a fixed oracle set the YES state is entered next, otherwise the NO state is entered. Each such string w is a query.

In the general case no restrictions are placed on the number of queries made by an oracle Turing machine to an oracle set or on when during the computation queries are made. Oracle machines where the number of queries made to the oracle is restricted are considered in several areas of this thesis. The restriction is usually specified as a function $f(n)$, for inputs of length n. In the case where queries are made at any time during the computation, the oracle machine is said to query the oracle *adaptively* or in *serial*; i.e., answers to early queries can be used to determine later queries. If all

[2]Showing $P^{NP} \subsetneq NEXP$ may in fact be possible with techniques which relativize. This is discussed in Chapter 3.

queries are made at one time, in other words in *parallel*, then the oracle is said to be queried *nonadaptively*. Specifically in Chapter 3 we consider some consequences of separating deterministic time classes relative to a fixed oracle where the oracle is queried nonadaptive. Also in Chapter 3 we examine deterministic time classes relative to a fixed oracle such that, for any input x where $|x| = n$, and function f, at most $f(n)$ adaptive queries are made.

The exact relationships between many of the central complexity classes, specifically, deterministic and nondeterministic classes are not known. To gain some insight into these relationships they have been examined relative to oracle sets. Relativized results are used as a tool to indicates the difficulty of resolving a proposition in the unrelativized case. In some cases relationships which are generally believed to be true can be shown to be true relative to an oracle set. If there is an oracle relative to which a proposition is true, then this can be viewed as evidence that the proposition is in fact true. Results relative to an oracle can also be counterintuitive. There may be one oracle relative to which a proposition is true and another oracle relative to which the same proposition is false. These contradictory results indicate that proof methods that do relativize will not be useful in determining the truth of a proposition. As many of the techniques which have been used do relativize, resolving such propositions is often said to be hard [3].

It has been shown by both Wilson and Dekhtyar [Dek76, Wil80, BWX82] that there exist oracle sets A such that $P^A \neq NP^A$ but $E^A = NE^A$. This implies that a proof showing $P \neq NP$ would not necessarily show $E \neq NE$. On the other hand, as already mentioned, we know that if $P = NP$ then $E = NE$.

There are several results which give insight into the possibility of separating exponential time-bounded classes from polynomial time classes using techniques which relativize. First Dekhtyar showed that there is an oracle A such that $NP^A = EXP^A$

[3]Techniques which are borrowed from recursive function theory are widely used and do relativize.

[Dek76]. This gives evidence that separating EXP from NP will be difficult. On the other hand, in [GH83] it is shown that for each $k > 0$ oracles A, B and C exist such that (i) $P^A \subsetneq NP^A \subsetneq EXP_k^A$, (ii) $EXP_k^B \subsetneq NP^B$ and (iii) NP^C and EXP_k^C are incomparable with respect to inclusion [4].

We are especially interested in the possibility of separating $NEXP$ from some level of the polynomial hierarchy. A reasonable assumption is that $PH \subsetneq NEXP$ as this is a consequence of $EXP \subsetneq NEXP$, but it appears that proper containment of PH in $NEXP$ will be difficult to prove. From Heller we know that there is an oracle A such that $\Delta_2^{P^A} \subsetneq \Sigma_2^{P^A} = \Delta_2^{EXP^A} \subsetneq \Sigma_2^{EXP^A}$ [Hel86] so there is a set A such that $NP^{NP^A} = EXP^A$. As we know that $P^{NP[n^c]} \subsetneq NEXP$, for any fixed integer c, this leaves the relationship of P^{NP} to $NEXP$ as an open question. To date, no oracle has been constructed such that P^{NP^A} is equal to $NEXP^{NP^A}$.

If we consider probabilistic polynomial time then again there is an oracle A, due to Heller, such that $BPP^A = EXP^A$ [Hel86] [5] Once again this gives evidence that separating classes from EXP may be hard.

Oracle results also give evidence that the structure of the polynomial hierarchy, PH, differs from that of exponential time hierarchies. Hartmanis, Immerman and Sewelson [HIS85] show that there exists an oracle A such that $E^A = NE^A$ but $NE^A \neq \Sigma_2^{E^A}$. This leads them to comment that the upward collapse of the polynomial hierarchy, which is implied if there is a collapse at any level of the polynomial hierarchy, may be a peculiarity of the structure of the polynomial hierarchy and not the general case. Nevertheless, Hartmanis, Immerman and Sewelson still conjecture that if $E = NE$ then this implies the collapse of the entire exponential time hierarchy.

[4] $EXP_k = \bigcup_{c=0} DTIME(2^{cn^k})$.

[5] In fact Heller shows that there is an oracle A such that $R^A = \Delta_2^{EXP^A}$.

1.3.4 Strong Exponential Time Hierarchies

In the literature both $EXPH$ and EH are frequently studied exponential time hierarchies. These hierarchies are both defined by considering exponential time classes using oracle sets that are in the polynomial time hierarchy. Two other hierarchies defined via exponential time classes are SEH and $SEXPH$. They are called *strong exponential time hierarchies* [6]. These hierarchies are defined by considering classes in the polynomial hierarchy using sets in $NEXP$ or NE as oracles. The first four levels of $SEXPH$ are

$$EXP,\ NEXP,\ NP^{NEXP},\ NP^{NP^{NEXP}}.$$

SEH is defined similarly using E and NE instead of EXP and $NEXP$. In both hierarchies the Δ_i classes are defined in an analogous way; i.e., $\Delta_1^{SEXP} = P^{NEXP}$. These classes are extensively studied by Hemachandra in his thesis [Hem87]. In his thesis Hemachandra shows that these hierarchies collapse,

$$E \neq P^{NE} = SEH = P^{NEXP} = SEXPH$$

and gives the following downward separation results.

If either $E = NE$ or $EXP = NEXP$ then $EXP = SEXPH$.

If either $NE = co{-}NE$ or $NEXP = co{-}NEXP$ then $NEXP = SEXPH$.

Hemachandra also shows that $EXP_{tt}^{NP} \subseteq P^{NEXP}$, so there is a collapse at the lowest level of the $EXPH$ under the assumption that either $EXP = NEXP$ or $NEXP = co{-}NEXP$. In Chapter 3 we will further exploit this collapse.

[6]They are called strong because there exists an oracle A such that SEH^A is not contained in EH^A (see [Hem87] for a discussion).

1.3.5 Advice Classes and Exponential Time

We have already mentioned some results obtained using sparse sets, now we look at results concerning sparse sets, the study of circuit complexity and advice classes. It is well known that a language has small (polynomial-size) circuits if and only if it is polynomial time Turing reducible to a sparse set. Equivalently, such a language is said to be in the nonuniform complexity class $P/poly$ also referred to as an advice class. A set A is in $P/poly$ if there is an *advice function* $f : \mathrm{N} \to \Sigma^*$ and a $B \in P$ such that $x \in A$ if and only if $\langle x, f(|x|)\rangle \in B$. So the *advice string*, $f(|x|)$, depends only on the length of x.

Kannan showed that $NEXP^{NP}$ does not have polynomial-size circuits [Kan82]. As every set in BPP has polynomial-size circuits this implies that $BPP \subsetneq NEXP^{NP}$. Beside giving a separation of $NEXP^{NP}$ from bounded probabilistic polynomial time this result indirectly relates $NEXP^{NP}$ to PH as $BPP \subseteq \Sigma_2^P \bigcap \Pi_2^P$ [Lau83, Sip83].

In Chapter 4 we further explore the relationship of exponential time classes and advice classes. We show that $EXP \not\subseteq DTIME(2^{O(n^{c_1})}/n^{c_2})$, for any fixed integers c_1, c_2. This implies, for example, that $EXP \not\subseteq E/lin$. We also show that $EXP^{NP} \not\subseteq P^{NP}/n^c$, for any fixed integer c. As was mentioned, there is an oracle A such that $BPP^A = EXP^A$ so, since $BPP \in P/poly$ nonrelativizing techniques will be needed to improve these results.

In the context of an examination of superpolynomial circuits and almost sparse oracles Buhrman and Homer [BH92] examine small circuits in relationship to the EXP-time hierarchy. Buhrman and Homer show that if EXP^{NP} is in $P/poly$ then $EXP^{NP} = \Sigma_2^P \bigcap \Pi_2^P$ and that if E^{NP} is in $P/poly$ then $E^{NP} \subseteq \Sigma_2^P \bigcap \Pi_2^P$. Buhrman and Homer accomplish this by showing that if any Δ level of the EXP hierarchy is contained in $EXP/poly$, then it is contained in EXP. Formally, if Δ_i^{EXP} is in $EXP/poly$, $i \geq 1$ then $\Delta_i^{EXP} = EXP$. Unfortunately, the techniques used in Buhrman and Homer's results do not appear to be applicable to $NEXP/poly$.

In [BFNW91] BPP is related to EXP and to probabilistically checkable proofs. It is shown that BPP has weak subexponential simulations, i.e. a simulation in time 2^{n^ϵ} for infinitely many values of n and every $\epsilon > 0$, unless EXP has polynomial size circuits and is contained in the class MA of languages with probabilistically checkable proofs. Since $MA \in \Sigma_2^P \bigcup \Pi_2^P$ then subexponential time simulations for BPP exist unless EXP is in the second level of the polynomial-time hierarchy.

1.3.6 Further Work on Exponential Time Classes

This overview is limited to results which explore the relationship between exponential time classes and polynomial time classes. It should be noted that interesting work has been done strictly on the structure of exponential time classes. A few of these papers are [Fu93, FLZ92, Gan90, LM93, TFL91, Wan90].

Although the structure of the NP complete sets is not well understood one of the best understood classes of polynomial-time many-one complete sets are those in EXP. Kurtz, Mahaney and Royer give a comprehensive overview of these results in [KMR90]. Results pertaining to the relationship of the isomorphism conjecture, one-way functions and exponential time classes are also presented here. In general they show that many of the proof techniques that are used in recursive function theory apply nicely to exponential time classes.

Further, exponential time classes have been related to the probabilistic model which describes interactive proof systems. In [BFL90] it is shown that the class of languages having two-prover interactive proof systems, MIP is $NEXP$. This is in direct contrast to an oracle result which states that relative to some oracle $co-NP$ does not have multi-prover interactive proof systems [FRS88].

There has also been a series of papers devoted to the systematic investigation of the internal, measure-theoretic structure of E and EXP. These results can be found in [Lut93, LM93].

1.4 The Boolean Hierarchy

The Boolean hierarchy has not been extensively examined with regard to exponential time classes. As Chapter 4 of this thesis begin this study, a brief overview is presented next.

The classes which form the lowest level of the Boolean hierarchy over NP, were introduced initially in [PY82] and the complete hierarchy was studied under different definitions in [KSW87, CH86, WW85]. We will characterize it as the hierarchy of nested difference classes over NP. It is known that the classes in the Boolean hierarchy, BH, are intertwined with the class of sets in $\bigcup_{k\geq 1} P^{NP||[k]}$ [Bei91, KSW87]. (This notation indicates that for any language L in $P^{NP||[k]}$ the oracle Turing machine computing L makes at most k parallel queries to an NP oracle on any input string.) These classes are referred to as parallel query classes. If queries are not restricted to being in parallel then the resulting classes are referred to as bounded query classes. Kadin [Kad88] showed that if the bottom two levels of BH collapse then $NP/poly = co{-}NP/poly$. This, combined with a result by Yap [Yap83] that if $NP/poly = co{-}NP/poly$ then $PH \subseteq \Sigma_3^P$ and a close observation of Kadin's proof gives that if $BH_i = co{-}BH_i$ then $PH \subseteq \Delta_3^P$. This result links the collapse of both the Boolean hierarchy and bounded query hierarchy to the collapse of the polynomial hierarchy. In [CK90] this result is refined to show that if the Boolean hierarchy collapses to level i, then PH is equal to the i^{th} level of the Boolean hierarchy over Σ_2^P. For a detailed examination of Boolean hierarchies see [Cha91, Sit90] [CGH$^+$88] [CGH$^+$89] .

In Lozano's thesis the Boolean hierarchy over $NEXP$ is considered in his closing remarks. Lozano observes that if the bounded query hierarchy over NE collapses then this implies, via a padding argument , that $co{-}NEXP \subseteq NEXP/poly$ [Loz92] He leaves as an open question: Does this imply that $NEXP = co{-}NEXP$?

1.5 The Complexity of Function Classes

In general, problems in complexity theory have been stated as questions concerning set membership. [7] An alternative is to consider the more general case of partial functions. The advantage of considering functions as opposed to set membership questions is that a function value can be an instance of a solution to a problem, whereas, a corresponding set membership question just states the existence of an instance. In the process of studying optimization problems Krentel [Kre88] began a very interesting examination of partial functions with oracles, particularly $PF^{NP[f(n)]}$. He directly related the complexity of specific function classes to set recognition problems. In addition, Krentel showed that if $PF^{NP[\log]} = PF^{NP}$ then $P = NP$. A study of the classes of partial functions that are obtained by limiting the number of queries to some $k \geq 1$, $PF^{NP[k]}$ and $PF_{tt}^{NP[k]}$ was carried on by Beigel [Bei88]. Selman continued this by formalizing the basic relationships between function classes, presenting all known inclusion relations of these classes and examining function classes with both adaptive and nonadaptive queries [Sel92]. For instance, Selman shows that $P_{tt}^{NP} = P^{NP}$ if and only if $PF_{tt}^{NP} = PF^{NP}$ and that if $PF^{NP[\log n]} = PF_{tt}^{NP}$, then $R = NP$. A further study of function classes, where the oracle may also be a partial multivalued function such as PF^{NPMV}, can be found in [FHOS92].

1.6 Overview of Results

Each chapter is motivated by a specific question relating exponential time classes to polynomial time classes. In Chapter 3 this question is: Is P^{NP} properly contained in $NEXP$? An equivalent question is: Is P^{NP} properly contained in P^{NEXP}? In

[7]Even in [HS65] set recognition was not clearly the preferred model but was gaining popularity probably eventually due to the simplification of the theory of nondeterminism [GJ79, Ste90].

answer we show that

$$\text{for } i \geq 1,\ \Delta_{i+1}^{P[n^c]} \subsetneq \Sigma_i^{EXP}, \text{ where } c \text{ is a fixed integer.}$$

A consequence of this is that $P^{NP[lin]} \subsetneq NEXP$. This improves a corollary to Fu, Li and Zhong's result that $NE \not\subseteq P^{NP[n^{o(1)}]}$, namely that $P^{NP[n^{o(1)}]} \subsetneq NEXP$ [FLZ92]. The general theorem which we prove is:

Theorem 1.6.1 *Given $C = \bigcup_{f \in F} DTIME(f(n))$, where F is a fixed family of time constructible functions,*

$$\textit{if } C \subsetneq EXP \textit{ then, for any fixed constant } c,\ C^{\Sigma_i^P[n^c]} \neq \Sigma_i^{EXP} \textit{ for } i \geq 1.$$

In the same vein, we also show a similar result for truth-table reductions.

Theorem 1.6.2 *Given $C = \bigcup_{f \in F} DTIME(f(n))$, where F is a fixed family of time constructible functions,*

$$\textit{if } C \subsetneq EXP \textit{ then } c,\ C_{tt}^{\Sigma_i^P} \neq \Sigma_i^{EXP} \textit{ for } i \geq 1.$$

This implies, for example, that $P_{tt}^{NP} \subsetneq NEXP$.

We end Chapter 3 with a general hierarchy theorem. This theorem is stated in terms of alternating Turing machines in which the number of alternations in bounded.

The motivating questions for Chapter 4 are: Is BPP properly contained in EXP? and Is $EXP \not\subseteq P/poly$? These are related as every set in BPP is in $P/poly$. With regard to advice classes and exponential time classes we show:

Theorem 1.6.3 *$EXP \not\subseteq DTIME(2^{O(n^{c_1})})/n^{c_2}$ for any fixed integers c_1 and c_2.*

This implies for example that $EXP \not\subseteq E/lin$. Along these same lines we show that this theorem relativizes.

Theorem 1.6.4 *For any oracle A $EXP^A \not\subseteq DTIME(2^{O(n^{c_1})}, A)/n^{c_2}$ for any fixed integers c_1 and c_2.*

This implies $EXP^{NP} \not\subseteq P^{NP}/n^c$. This theorem is also, with a little work, modified to give the same result with respect to truth table reductions implying $EXP_{tt}^{NP} \not\subseteq P_{tt}^{NP}/n^c$. These results were done jointly with Steve Homer.

Chapter 5 generalizes Kadin's collapses of the polynomial time Boolean hierarchy to the exponential time Boolean hierarchy over $NEXP$. It is shown that if the lowest levels of the EXP-time Boolean hierarchy are equal then $NEXP/poly = co-NEXP/poly$. This leads to the question: Does $co-NEXP \subseteq NEXP/poly$ imply that the exponential time hierarchy collapses? This question is discussed here.

In Chapter 6 the motivating question is: Can the structure of function classes be used to show separations between classes of sets? Our answer is somewhat inconclusive in that what we show is yes in some cases and probably not in other cases. Even though it is not known if $P^{NP[\log]}$ is properly contained in EXP we show that,

$$\text{for all } i,\ PF^{\Sigma_i^P[\log]} \subsetneq EXPF_{PB}.[8]$$

This result does not imply that $PF^{\Sigma_i^P} \neq EXPF_{PB}$ because, unlike the corresponding classes in PH, it does not appear that $PF^{\Sigma_i^P} \subsetneq PF^{\Sigma_{i+1}^P[\log]}$, for any i, as this implies $P = NP$. What we would like to show is that if $PF^{NP[\log]}$ is properly contained in $EXPF_{PB}$, then $P^{NP[\log]}$ is properly contained in EXP. Unfortunately, this does not follow from the techniques used in Chapter 6. We do show that if $P_{CB}^{NP[\log n]}$ is properly contained in EXP_{CB}, the class of constant-bounded exponential functions, then $P^{NP[\log n]} \subsetneq EXP$. We end by showing that $PF^{NEXP} = EXPF_{PB}^{NP[poly]}$ and by examining the relationship of $PF^{NEXP[\log]}$ to $EXPF_{PB}$.

[8] $PF^{\Sigma_i^P[\log]}$ is the set of all partial functions which can be computed deterministically by a polynomial time-bounded oracle transducer which, on input x, makes at most $O(\log|x|)$ queries.

Chapter 2

Preliminaries

2.1 Basic Definitions and Notation

This chapter gives the basic notation and definitions used in this thesis. The standard deterministic/nondeterministic multi-taped Turing machine is our basic computation model. Variations of this model will be defined in this chapter. A knowledge of the definitions of Turing machines, basic time and space-bounded computation, and common reductions between classes ($\leq_m^{\mathrm{P}}$, $\leq_T^{\mathrm{P}}$) is assumed. For further information on these topics see [HU79, BDG88, BDG90].

All languages considered in this thesis are subsets of $\Sigma = \{0,1\}^*$. Languages are denoted by capital letters $A, B, \ldots$. The *complement* of a language A is $\Sigma^* - A$ and is denoted by $\bar{A}$. Strings are elements of Σ^* and are denoted by small letters $w, x, y, \ldots$ or by $\alpha, \beta, \gamma, \ldots$. We will use $|x|$ to designate the length of x, where x is an element of a language. Let N be the set of natural numbers. For a set A and $n \in \mathrm{N}$ define

$$A_n = \{x \in A \mid |x| = n\}.$$

Let $\langle \cdot, \cdot \rangle$ denote a natural encoding of two strings into one string. We may assume that this pairing function is polynomial-time computable and invertible.

A set S is *sparse* if $||\{x \in S \mid |x| \leq n\}|| \leq n^c$ for some fixed c and all n.

The class of languages of space complexity $f(n)$ is denoted by $DSPACE(f(n))$. The class of languages of time complexity $f(n)$ is denoted by $DTIME(f(n))$ and class of languages of nondeterministic time complexity $f(n)$ is denoted by $NTIME(f(n))$. The following complexity classes appear through out this thesis.

$PSPACE = \bigcup_{c \geq 0} DSPACE(n^c)$

$EXPSPACE = \bigcup_{c \geq 0} DSPACE(2^{n^c})$

$P = \bigcup_{c \geq 0} DTIME(n^c)$

$NP = \bigcup_{c \geq 0} NTIME(n^c)$

$E = \bigcup_{c \geq 0} DTIME(2^{cn})$

$NE = \bigcup_{c \geq 0} NTIME(2^{cn})$

$EXP = \bigcup_{c \geq 0} DTIME(2^{n^c})$

$NEXP = \bigcup_{c \geq 0} NTIME(2^{n^c})$

For each class of languages C, let $co-C$ be the set of complements of the sets in C, $co-C = \{A \mid \bar{A} \in C\}$. Classes which are defined via deterministic time or space bounded Turing machines are closed under complementation. Various nondeterministic space classes are closed under complementation [Imm88, Sze88]. For classes defined via nondeterministic time-bounded machines this is an open question. If C is strictly contained in C', then we write $C \subsetneqq C'$.

By the time hierarchy theorems [HS65, Coo73, BG70] :

$$P \subsetneqq E \subsetneqq EXP \text{ and } NP \subsetneqq NE \subsetneqq NEXP.$$

By the space hierarchy theorem [HLS65] :

$$PSPACE \subsetneqq EXPSPACE.$$

References for the following inclusions can be found in [HU79, BDG88]

$$P \subseteq NP \subseteq PSPACE \subseteq EXP \subseteq NEXP \subseteq EXPSPACE.$$

Let $M_0, M_1, M_2, \ldots$ be a fixed enumeration of Turing machines. Let $L_i = L(M_i)$, the language accepted by machine M_i. For any of the complexity classes defined above we may consider an enumeration $\{M_e\}$ of machine for that class. Specifically we will consider an enumeration of nondeterministic Turing machines for $NEXP$ and define $K_{\Sigma_1^{EXP}} = \{\langle e, x, t\rangle \mid M_e \text{ accepts } x \text{ in } \leq t \text{ steps }\}$. The set $K_{\Sigma_1^{EXP}}$ is $\leq_m^{\mathrm{P}}$-complete for $NEXP$ and for NE.

The following function classes, as previously defined in [BDG90], will be used. In each case the function f is from Σ^* to Σ^*.

$log = \{f \mid f(n) = c \cdot \log_2 n$ for some constant c $\}$

$lin = \{f \mid f(n) = c \cdot n$ for some constant c $\}$

$poly = \{f \mid f(n) = c \cdot n^k$ for some constants c, k $\}$

2.2 Computing Relative to an Oracle

Machine based complexity classes *relativize* by allowing a set to be used as an oracle. An oracle Turing machine is a multi-taped Turing machine that also has a separate work tape for queries and three distinguished states: QUERY, YES and NO. Given a fixed oracle set, if, on input x, a QUERY state is entered then the oracle Turing machine will enter the YES or NO state depending on whether or not the string currently on the query tape is in the oracle set. If some language L is accepted by an oracle Turing machine M using the set A as an oracle and if M is time-bounded (space-bounded) by some function f then we say that L is accepted in time (space) $f(n)$ *relative* to A.

If L is accepted in time $f(n)$ relative to A and $TIME(f(n)) \subseteq C$ for some complexity class C, then $L \in C^A$. We will specifically be interested in the classes P^A, NP^A, EXP^A and $NEXP^A$ for various oracle sets A. The definitions for space-bounded classes are analogous except in this case the size of the oracle tape must also be specified [1].

Clearly, if a language L is in C^A and A is in C' then L is in $C^{C'}$. For any language $L \in C^{C'}$, if C is a class of languages that are at least polynomial time-bounded and A is polynomial time many-one complete for C' then $L \in C^A$. For example, if $L \in P^{NP}$ then $L \in P^{SAT}$.

The number of queries to the oracle can be limited. If L is in C^A for a fixed oracle set A via Turing machine M and on input x at most $f(n)$ queries are made to A, then we write $L \in C^{A[f(n)]}$ or simply $L \in C^{A[F]}$ where F is a family of functions and $f \in F$. In particular the classes $P^{A[f(n)]}$, $NP^{A[f(n)]}$, $EXP^{A[f(n)]}$ and $NEXP^{A[f(n)]}$ will be considered for various sets A and functions f.

We will also consider *polynomial time truth-table reductions* denoted by $\leq_{tt}^{\mathrm{P}}$. We say $A \leq_{tt}^{\mathrm{P}} B$ if and only if $A \leq_{T}^{\mathrm{P}} B$ via an oracle Turing machine which writes down all of the queries on the query tape before any queries are made. In other words, all queries are made without considering the answer to previously asked queries; i.e., nonadaptively. The definition of polynomial time truth-table reductions is generalized to other time-bounds as follows. $L \leq_{tt}^{\mathrm{C}} A$ if and only if $A \leq_{T}^{\mathrm{C}} B$ via a $f(n)$ time-bounded oracle Turing machine which writes down all of the queries on the query tape before any queries are made and such that $TIME(f(n)) \subseteq C$. If $L \leq_{tt}^{\mathrm{C}} A$, then we write $L \in C_{tt}^{A}$. The classes P_{tt}^{NP} and EXP_{tt}^{NP} will be of interest.

[1]For example, consider $PSPACE^{EXP}$. If the size of the query tape is not bounded by some polynomial then it is possible to write a string which is exponentially long, relative to the input length, on the query tape. But this is then equivalent to querying an oracle in double exponential time.

2.3 Hierarchies

2.3.1 Using Alternating Turing Machines to Define Hierarchies

Alternating machines will be used to define time-bounded hierarchies. An alternating Turing machine, or ATM, is a generalization of a multi-tape nondeterministic Turing machine in the following way. The computation of a nondeterministic Turing machine can be thought of as answering an existential question; i.e., does there exist a computation path through a computation tree that results in an accepting state. Likewise, we can think of the universally quantified question; i.e., do all computation paths through a computation tree lead to accepting states. An alternating Turing machine is a Turing machine in which each state is labeled either existential, universal, accepting or rejecting. [BDG90].

A *configuration* of an ATM, M on input x, is defined in the same way that a configuration of a Turing machine is defined. In addition, if the current state is a universal (resp., existential, accepting, rejecting) state then the configuration is said to be a universal (resp., existential, accepting, rejecting) configuration. Likewise, *computation path, initial configuration, immediate successor, successor* and *computation tree* are all defined as for Turing machines.

We will define an accepting computation for an alternating Turing machine via a labeling of a computation tree as in [BDG90].

Definition 1 *Given a tree in which internal nodes are either universal or existential, we denote by l the result of the following labeling procedure:*

1. *the accepting leaves are labeled 1;*
2. *an existential node is labeled 1 if at least one of it's sons is labeled 1;*
3. *a universal node is labeled 1 if all of it's sons are labeled 1.*

We say that an ATM M accepts x if there is a computation tree of M on x in which the root gets label 1. The subtree of the computation tree which has all the nodes labeled 1 is called the accepting computation tree of M on x.

An ATM, M, is time-bounded by t if for any input x in $L(M)$, with $|x| = n$, there exists an accepting computation subtree of M on x with height at most $t(n)$. Similarly, an ATM, M, is space-bounded by s if for any input x in $L(M)$, with $|x| = n$, there exists an accepting computation subtree of M on x such that the space used by each configuration in this computation subtree is bounded above by $s(n)$. The class of languages accepted by time-bounded alternating machines is defined as follows (ASPACE(t(n)) is defined analogously).

Definition 2 *For any time bound function $t(n)$ denote by $ATIME(t(n))$ the class of languages accepted by $t(n)$ time-bounded ATMs.*

By bounding the number of alternations a general definition for time and space-bounded hierarchies arises.

Definition 3 *Let M be an ATM and let x be an input. We say M is $A(n)$-alternation bounded on x if any path of maximum length of any accepting computation tree of M on x alternates universal and existential configurations at most $A(n) - 1$ times.*

So we say, for any $k > 0$, a $A\Sigma_k$-machine (resp. $A\Pi_k$-machine) is a k-alternation bounded ATM which starts with an existential (resp. universal) state.

Then, for a function f, let Σ_k^f (Π_k^f) denote the class of languages accepted by a $A\Sigma_k(A\Pi_k)$ alternating Turing machine which runs in time $f(n)$. For a class of functions F, $\Sigma_k^F = \bigcup_{f\in F} \Sigma_k^f$ and $\Pi_k^F = \bigcup_{f\in F} \Pi_k^f$. Using this definition we define three hierarchies that will be studied in this paper:

1.) Letting $F = \{$ polynomial-time computable functions $\}$ we obtain the usual levels of the polynomial-time hierarchy, PH.

2.) Letting $F = \{2^{p(n)} \mid p$ is a polynomial $\}$ we obtain the levels of the exponential-time hierarchy $EXPH$.

3.) Letting $F = \{2^{cn} \mid c$ is a constant$\}$ we obtain the levels of the exponential-time hierarchy EH.

The i^{th} sigma levels of these hierarchies will be denoted Σ_i^P, Σ_i^{EXP} and Σ_i^E. The pi levels will be denoted Π_i^P, Π_i^{EXP} and Π_i^E. Each of these hierarchies defined inductively can also be represented using the classes usually denoted by Δ_i^F. For PH, $\Delta_i^P = P^{\Sigma_{i-1}^P}$. The delta classes in both exponential hierarchies are defined in terms of classes in the polynomial hierarchy [2]. For $EXPH$, $\Delta_i^{EXP} = EXP^{\Sigma_{i-1}^P}$ and for E, $\Delta_i^E = E^{\Sigma_{i-1}^P}$.

The inductive definition of each of these hierarchies is based on using the class Σ_i^P as an oracles to build the $i+1^{st}$ delta, sigma and pi levels. As these classes are modeled using oracle Turing machines we can consider the variations of the Turing machine model that are presented in Section 2.2. Specifically, in terms of the polynomial hierarchy:

$\Sigma_i^{P[F]}$ is the set of languages in $NP^{\Sigma_{i-1}^P[F]}$.

$\Pi_i^{P[F]}$ is the set of languages in $co\!-\!NP^{\Sigma_{i-1}^P[F]}$.

$\Delta_i^{P[F]}$ is the set of languages in $P^{\Sigma_{i-1}^P[F]}$.

$\Sigma_{i,tt}^P$ is the set of languages in $NP_{tt}^{\Sigma_{i-1}^P}$.

$\Pi_{i,tt}^P$ is the set of languages in $co\!-\!NP_{tt}^{\Sigma_{i-1}^P}$.

$\Delta_{i,tt}^P$ is the set of languages in $P_{tt}^{\Sigma_{i-1}^P}$.

[2]Unlike polynomial functions, an exponential function composed with an exponential does not yield another exponential function but will be doubly exponential. It follows then that EXP^{NEXP} is not even in $EXPSPACE$.

The definitions for $EXPH$ are similar.

$\Sigma_i^{EXP[F]}$ is the set of languages in $NEXP^{\Sigma_{i-1}^P[F]}$.

$\Pi_i^{EXP[F]}$ is the set of languages in $co{-}NEXP^{\Sigma_{i-1}^P[F]}$.

$\Delta_i^{EXP[F]}$ is the set of languages in $EXP^{\Sigma_{i-1}^P[F]}$.

The definitions for EH are analogous to those for $EXPH$.

2.3.2 The Strong Exponential Hierarchy

The most frequently studied hierarchies can be defined via alternating Turing machines but not all hierarchies are known to be definable in this way. An open question in [Hem87] asks if the strong exponential hierarchies can be defined via alternating Turing machines. The following inductive definition of the strong exponential hierarchies will be used.

Definition 4 *The strong exponential hierarchy.*

$$\Sigma_0^{SEH} = E$$
$$\Sigma_1^{SEH} = NE$$
$$\Sigma_i^{SEH} = NP^{\Sigma_{i-1}^{SEH}} \quad \textit{for } i \geq 2$$
$$\Delta_i^{SEH} = P^{\Sigma_{i-1}^{SEH}} \quad \textit{for } i \geq 2$$

$$SEH = \bigcup_{i \geq o} \Delta_i^{SEH} = \bigcup_{i \geq o} \Sigma_i^{SEH}$$

Substituting EXP for E and $NEXP$ for NE, the classes of $SEXPH$ are defined analogously. As was mentioned in the previous chapter both hierarchies collapse to the Δ_2 level and $SEH = SEXPH = P^{NE} = P^{NEXP}$ [Hem87].

2.4 Computing with Probabilistic Models and Computing with Advice

A nondeterministic Turing machine can compute a probabilistic algorithm by giving the machine access to an random source such as an ideal random number generator. Further, complexity classes can be defined in terms of these probabilistic machines. In this thesis we will be concerned with two such class denoted BPP, bounded error probabilistic polynomial time, and R, random polynomial time. These class were originally defined in [Gil77, AM77] and references to all of the properties stated here can be found in [Sch85, BDG88].

First we describe a general probabilistic machine. A *probabilistic* Turing machine is a nondeterministic Turing machine where each nondeterministic choice is considered a random experiment in which the outcome has equal probability. We assume, with out loss of generality, that each nondeterministic branch has two possible outcomes each with probability 1/2. Now each nondeterministic computation of length n has probability 2^{-n}. A probabilistic Turing machine has three types of final states: accepting or a-state, rejecting or r-states and undetermined or ?-state. The outcome of the machine on input x is now a random variable whose range is $(a, r, ?)$. Let $Pr[M(x) = y]$ denote the probability that machine M on input x halts in a y-state. The probability that M accepts an input x is the sum of the probabilities of all accepting paths.

Definition 5 *BPP is the class of languages recognized by polynomial time probabilistic Turing machines whose error probability is bounded above by some positive constant $\epsilon < 1/2$.*

Definition 6 *R is the class of languages recognized by polynomial time probabilistic Turing machines which have zero error probability for inputs not in the language, and* *error probability bounded above by some positive constant $\epsilon < 1/2$ for words in the language.*

It is easy to see that $R \subseteq BPP$ and as BPP is closed under complementation which implies that $co{-}R \subseteq BPP$. In addition, it is known that

$$P \subseteq R \subseteq NP.$$

The relationship of NP to BPP is not known. It is known that $NP \subseteq BPP$ implies that $NP = R$ [Ko82] and also that $NP \subseteq BPP$ implies that $PH = \Sigma_2^P$ [KL80]. Neither of these consequences is believed to be true.

It is known that every language in BPP is in the nonuniform complexity class $P/poly$ which is defined using the following general definition for advice classes.

Definition 7 *An* advice *functions is a functions $f : \mathbf{N} \to \Sigma^*$. Let C be a complexity class, and F a family of advice functions. The class C/F is the collection of all sets A such that for some $B \in C$ and a function $f \in F$*

$$x \in A \textit{ if and only if } \langle x,\ f(|x|)\rangle \in B.$$

2.5 The Boolean Hierarchy over $NEXP$

As was mentioned in the introduction there are several equivalent definitions for the Boolean hierarchy. We will start by defining the bottom levels of both the polynomial time and exponential time Boolean hierarchies. The bottom levels of the Boolean hierarchy over NP are D^P and $co-D^P$. These classes were originally defined in [PY82].

$$D^P = \{L_1 \cap \overline{L_2} \mid L_1, L_2 \in NP\}$$

$$co{-}D^P = \{\overline{L_1} \cup L_2 \mid L_1, L_2 \in NP\} = \{\overline{L} \mid L \in D^P\}$$

Complete sets for D^P and $co{-}D^P$ are:

$$SAT \wedge \overline{SAT} = \{(F_1, F_2) \mid F_1 \in SAT \text{ and } F_2 \in \overline{SAT}\}$$

$$\overline{SAT} \vee SAT = \{(F_1, F_2) \mid F_1 \in \overline{SAT} \text{ or } F_2 \in SAT\}$$

where F_1 and F_2 are boolean formulas. The following classes are the analogous classes for exponential time.

$$D^E = \{L_1 \cap \overline{L_2} \mid L_1, L_2 \in NEXP\}$$

$$co{-}D^E = \{\overline{L_1} \cup L_2 \mid L_1, L_2 \in NEXP\}$$

As will be shown in Chapter 4, complete sets for D^E and $co{-}D^E$ are:

$$\begin{aligned}
K_{\Sigma_1^{EXP}} \wedge \overline{K_{\Sigma_1^{EXP}}} &= \{(\langle e_1, x_1, t_1\rangle, \langle e_2, x_2, t_2\rangle) \mid \langle e_1, x_1, t_1\rangle \in K_{\Sigma_1^{EXP}} \text{ and} \\
&\qquad\qquad \langle e_2, x_2, t_2\rangle \in \overline{K_{\Sigma_1^{EXP}}}\} \\
\overline{K_{\Sigma_1^{EXP}}} \vee K_{\Sigma_1^{EXP}} &= \{(\langle e_1, x_1, t_1\rangle, \langle e_2, x_2, t_2\rangle) \mid \langle e_1, x_1, t_1\rangle \in \overline{K_{\Sigma_1^{EXP}}} \text{ or} \\
&\qquad\qquad \langle e_2, x_2, t_2\rangle \in K_{\Sigma_1^{EXP}}\}
\end{aligned}$$

where e is treated as the index of a $NEXP$ machine, x as an input string and t as a time bound. Following the definition of the Boolean hierarchy for polynomial time in [CGH+88] define the Boolean hierarchy for exponential time, $EXPBH$, as follows.

Definition 8 *The Boolean hierarchy over $NEXP$.*

$$\begin{aligned}
&EXPBH_1 = NEXP \\
&EXPBH_{2i} = \{L_1 \bigcap \overline{L_2} \mid L_1 \in EXPBH_{2i-1} \textit{ and } L_2 \in NEXP\} \\
&EXPBH_{2i+1} = \{L_1 \bigcup L_2 \mid L_1 \in EXPBH_{2i} \textit{ and } L_2 \in NEXP\} \\
&co{-}EXPBH_i = \{L \mid \overline{L} \in EXPBH_i\}
\end{aligned}$$

$$EXPBH = \bigcup_{i\geq 1} EXPBH_i$$

2.6 Function Classes

The standard definitions for function classes will be used. Many of these definitions appear in [Sel92] where there is a complete discussion of function classes.

The computation model we use generalizes the basic definition for deterministic/nondeterministic multi-tape Turing machines by allowing a value to be written on an output tape. A transducer T is a nondeterministic Turing machine with a read-only input tape and a write-only output tape. On input string x, T computes a value y if there is an accepting computation path of T on x for which y is the final contents on the output tape. Nondeterministic transducers compute partial, multivalued functions. Accordingly, for an input string x, T may have many different possible output values each resulting from a different accepting computation path. If T on x outputs y then we write $T(x) \mapsto y$.

The following function classes appear throughout this thesis.

$PSPACEF$ = the set of all partial functions which can be computed in polynomial space.

PF = the set of all partial functions which can be computed deterministically by polynomial time-bounded transducer.

$NPMV$ = the set of all partial functions which can be computed nondeterministically by polynomial time-bounded transducer.

$NPSV$ = the set of all $f \in NPMV$ that are single valued.

$EXPF$ = the set of all partial functions which can be computed deterministically by an 2^{n^c} time-bounded transducer where c is a constant.

$NEXPMV$ = the set of all partial functions which can be computed nondeterministically by an 2^{n^c} time-bounded transducer where c is a constant.

$NEXPSV =$ the set of all $f \in NEXPMV$ that are single valued.

A function f is *constant-bounded* if there is a constant c such that for all x in the domain of f $|f(x)| \leq c$. For any classes of functions, let $C_{CB} =$ the set of all constant-bounded functions in C.

A function f is *polynomial-bounded* if there is a polynomial p such that, for all x in the domain of f, $|f(x)| \leq p(|x|)$. For any classes of functions, let $C_{PB} =$ the set of all polynomial-bounded functions in C.

We will assume that all functions in $PSPACEF$ are in fact in $PSPACEF_{PB}$ unless specifically stated otherwise; i.e., that the function output is also bounded by the space-bound.

Given partial multivalued functions f and g, define g to be a *refinement* of f if the domain of g is equal to the domain of f and for all x in the domain of g and all y, if $g(x) \mapsto y$, then $f(x) \mapsto y$.

Let F and G be classes of partial multivalued functions. If f is a partial multivalued function, we define $f \in_c G$ if G contains a refinement g of f, and we define $F \subseteq_c G$ if for every $f \in F$, $f \in_c G$.

We also allow a transducer T to access an oracle set in the same way that we have already defined for Turing machines. In this way we may consider a function f in $PF^{NP[\log]}$. In this thesis oracle sets will always be subsets of Σ^* and will never return a value other than 0 or 1.

If alternating machines are allowed to write a string to an output tape, in other words, the alternating machine is now a transducer, then we would like to generalize the notion of defining hierarchies based on alternating machines with output. In fact, this is not exactly correct as a particular nondeterministic transducer may compute a multivalued function. In this case there may be several different possible output values for some input. As the hierarchy of function classes that will be examined in this thesis are the Δ classes, we will use an inductive definition which does not rely on ATM's.

Definition 9 *The polynomial hierarchy for function classes.*

$$\Delta_0^{PF} = PF$$
$$\Delta_{i+1}^{PF} = PF^{\Sigma_i^P},\ i \geq 0$$

$$PHF = \bigcup_{i \geq 0} \Delta_i^{PF}$$

Likewise, exponential hierarchies can be defined for function classes. As in the case of EXP and E, these hierarchies are defined in terms of the polynomial hierarchy. The definition for $EXPHF$ is as follows and EHF is defined analogously.

$$\Delta_0^{EXPF} = EXPF$$
$$\Delta_{i+1}^{EXPF} = EXPF^{\Sigma_i^P},\ i \geq 0$$

$$EXPHF = \bigcup_{i \geq 0} \Delta_i^{EXPF}$$

Chapter 3

Exponential Time Classes and PH

In this chapter we relate the polynomial time hierarchy to the exponential time hierarchy. The main motivation is to answer the following open question:

$$\text{Is } P^{NP} \text{ properly contained in } NEXP?$$

Recently progress was made on this question by Fu, Li and Zhong using standard translation methods. In [FLZ92] Fu, Li and Zhong show that $NE \not\subseteq P^{NP[n^{o(1)}]}$. A consequence of this is that $P^{NP[n^{o(1)}]}$ is properly contained in $NEXP$. This follows easily as $P^{NP[n^{o(1)}]} \subseteq NEXP$ and if we assume that $NEXP \subseteq P^{NP[n^{o(1)}]}$ then this of course implies that $NE \subseteq P^{NP[n^{o(1)}]}$ which contradicts Fu, Li and Zhong's result. Further progress is made in this chapter.

How practical is it to assume that Fu, Li and Zhong's result can be improved? For example, how hard is it to show that P^{NP} is properly contained in $NEXP$? We answer this question by the convoluted reasoning that there is a lack of evidence indicating that a separation of P^{NP} from $NEXP$ will in fact be hard to prove. In other words, no proof has been given to show that these two classes are equal relative to an oracle set. It has not been proved that there is an oracle A such that $P^{NP^A} = NEXP^A$. There is an oracle due to Heller [Hel86] such that $P^{NP^A} \subsetneqq NP^{NP^A} =$

$NEXP^A = EXP^{NP^A}$ so we can't easily hope to separate NP^{NP} form $NEXP$ but this only makes extending Fu, Li and Zhong's separation to P^{NP} more interesting. In fact Fu, Li and Zhong mention that they have tried unsuccessfully to show either that $NE \not\subseteq P^{NP}$ or that there is an oracle A such that $NE^A \subseteq P^{NP^A}$ [FLZ92].

We now present a related question. In Hemachandra's thesis [Hem87] , as a part of his examination of the strong exponential time hierarchy, he shows that answering the question:

Is P^{NP} properly contained in P^{NEXP}?

is equivalent to answering the question:

Is P^{NP} properly contained in $NEXP$?

Obviously, if P^{NP} is properly contained in $NEXP$ then it is also properly contained in P^{NEXP}. So if the second question is answered in the affirmative then so is the first. What Hemachandra observed is that if $NEXP = co\text{–}NEXP$ then $NEXP = P^{NEXP}$. Using this downward separation result, we see that a positive answer to the first question implies a positive answer to the second question. If in fact $P^{NP} \neq P^{NEXP}$, then under the assumption that $P^{NP} = NEXP$ we see that, since $NEXP$ is closed under complementation, $NEXP = co-NEXP \Rightarrow NEXP = P^{NEXP}$. We conclude from this that $P^{NP} = P^{NEXP}$ which would be a contradiction. Therefore, if P^{NP} is properly contained in P^{NEXP} then P^{NP} is also properly contained in $NEXP$.

Again, to date, it has not been shown that P^{NP} is equal to P^{NEXP} relative to an oracle so we have no evidence that separating P^{NP} from P^{NEXP} will be hard. Although results in complexity theory can often be counterintuitive, since we know that $NP \neq NEXP$ we expect that P^{NP} is not equal to P^{NEXP}.

In this chapter, we improve Fu, Li and Zhong's result. First, using the fact that the time hierarchy theorem relativizes and that this can be generalized so that the relativized time hierarchy theorem is true even if only nonadaptive queries are considered, we show that $\Delta^{\mathrm{P}}_{i+1,tt} \subsetneq \Sigma^{\mathrm{EXP}}_{\mathrm{i}}$. This implies, for example, that $P^{NP}_{tt} \subsetneq NEXP$. Next we give a hierarchy theorem which diagonalizes over time-bounded oracle Turing machines while also bounding the number of queries made to the oracle. This is then used to show that $\Delta^{\mathrm{P}[n^c]}_{i+1} \subsetneq \Sigma^{\mathrm{EXP}}_{\mathrm{i}}$, where c is any constant. For example, this yields $P^{NP[n^c]} \subsetneq NEXP$ which implies $P^{NP[lin]} \subsetneq NEXP$. In the last section of this chapter we give a general hierarchy theorem in terms of alternating Turing machines. This theorem is used to demonstrate some simple relationships between the polynomial hierarchy and hierarchies for classes which use more than polynomial time.

3.1 Using Truth-table and Bounded Query Classes to Separate Classes in the PH from Classes in the Exponential Hierarchy

A consequence of the fact that the time hierarchy theorem relativizes is that every level of the polynomial hierarchy separates from the corresponding level of the exponential time hierarchy (see Section 3.2 for this proof). In this section we improve this separation by showing that for any fixed constant c, $\Delta^{P[n^c]}_{i+1}$ is properly contained in Σ^{EXP}_i. This implies, for example, that for any i, $\Delta^{P[lin]}_{i+1}$ is properly contained in Σ^{EXP}_i and specifically that $P^{NP[lin]}$ is properly contained in $NEXP$.

First we give a slightly weaker result that uses the same general proof technique. We show that every $\Delta^{P}_{i+1,tt}$ level of the polynomial hierarchy is properly contained in the Σ^{EXP}_i level of $EXPH$. This implies, for example, that P^{NP}_{tt} is properly contained in $NEXP$. This result is developed by generalizing and extending several observations that Hemachandra made about the class P^{NEXP}. In [Hem87], Hemachandra gives a downward separation result showing that

$$\text{if } NEXP = co{-}NEXP \text{ then } NEXP = P^{NEXP}.$$

Further he shows that EXP_{tt}^{NP} is contained in P^{NEXP}. Consequently if $NEXP = co{-}NEXP$ then $NEXP = EXP_{tt}^{NP}$. We explore these observations with respect to the classes $P^{\Sigma_i^{\mathrm{EXP}}}$ for any i. In this section, we show that

$$\text{if } \Sigma_i^{EXP} = co{-}\Sigma_i^{EXP} \text{ then } \Sigma_i^{EXP} = P^{\Sigma_i^{\mathrm{EXP}}} \text{ for any } i \geq 1.$$

Similarly we show that $\Delta_{i+1,tt}^{EXP}$ is contained in $P^{\Sigma_i^{\mathrm{EXP}}}$ and in fact $\Delta_{i+1}^{EXP[poly]} = \Delta_{i+1,tt}^{EXP} = P^{\Sigma_i^{EXP}}$ for $i \geq 1$. Now under the assumption that $\Sigma_i^{EXP} = \Delta_{i+1,tt}^{P}$ the class Σ_i^{EXP} will be closed under complementation and therefore equal to $\Delta_{i+1,tt}^{EXP}$. This implies that $\Delta_{i+1,tt}^{P} = \Delta_{i+1,tt}^{EXP}$ which contradicts the time hierarchy theorem with respect to truth-table reductions.

Consequently, P_{tt}^{NP} is properly contained in $NEXP$. This is not a new result as this is implied by a corollary of Fu, Li and Zhong's result that $NE \not\subseteq P^{NP[n^{o(1)}]}$ [FLZ92]. It follows from their result that $P^{NP[n^{o(1)}]}$ is properly contained in $NEXP$. Since $P^{NP[\log]} = P_{tt}^{NP}$ and $n^{o(1)}$ majorizes $\log(n)$ Fu, Li and Zhong's result implies that P_{tt}^{NP} is already properly contained in $NEXP$.

The time (space) hierarchy theorem relativizes so it has been observed that, if time classes T_1 and T_2 are distinct via the time hierarchy theorem, then so are T_1^A and T_2^A for any oracle A. This can be extended further to show that the classes of sets which are T_1 truth-table reducible to A are distinct from the class of set which are T_2 truth-table reducible to A. To see this we consider the proof of the time hierarchy theorem. In this theorem, one machine M treats its input $\langle i, w\rangle$ as a tuple representing the index to a Turing machine M_i (in a fixed enumeration of machines) and an input string w. The machine M then proceeds to simulate the computation of M_i on w. Given a particular time bound t on M, the machine M can then be used to diagonalize over all machines in an enumeration. Observe, all that is needed to adapt the time hierarchy theorem to consider only nonadaptive oracle machines is to reject any input $\langle i, w\rangle$ such that during the simulation of M_i on w, all queries are not made one step of $M_i's$ computation on w.

The following proposition is a straight forward generalization of a similar result covering the polynomial time hierarchy.

Proposition 3.1.1 *For any oracle A, $EXP^{A[poly]} \subseteq EXP^A_{tt}$.*

Proof. Given any set L in $EXP^{A[poly]}$ there are polynomials p and p' such that there exists a $2^{p(n)}$ time-bounded oracle Turing machine M accepting L that makes at most $p'(n)$ queries to A on any input of length n. Given any input of length n, if we consider all possible queries that can be made by M to the oracle then there are at most $2^{p(n)} * p'(n)$ queries. Since $2^{p(n)} * p'(n) \leq 2^{p(n)+p'(n)}$, then all queries in the computation can be made nonadaptively by a EXP^A_{tt} machine. □

Lemma 3.1.1 *For all $i \geq 1$, $\Delta^{EXP}_{i+1,tt} \subseteq P^{\Sigma_i^{EXP}}$.*

Proof. Given $L \in \Delta^{EXP}_{i+1,tt}$ via oracle machine M with an oracle set A in Σ_i^P. Let $2^{p(n)}$ be the running time of M. Consider input x with $|x| = n$. If we know how many of the $2^{p(n)}$ queries made to A receive a YES answer then a $\Sigma_{\mathrm{i}}^{\mathrm{EXP}}$ machine can guess which queries receive a YES, verify that they are in fact in A and then simulate the computation of M on x substituting the correct oracle answers. So given the correct number of YES answers only one query to a $\Sigma_{\mathrm{i}}^{\mathrm{EXP}}$ oracle is needed to determine if $x \in L$. But the number of correct YES answers can be found in polynomial time given a $\Sigma_{\mathrm{i}}^{\mathrm{EXP}}$ oracle via binary search, so $\Delta^{EXP}_{i+1,tt} \subseteq P^{\Sigma_i^{EXP}}$. □

Lemma 3.1.1 can be found in [Hem87] for $i = 1$.

In general, given a fixed enumeration, $\{M_e\}$, of nondeterministic oracle Turing machines which run in $NEXP$ time we have the following complete set for Σ_i^{EXP}:

$$K_{\Sigma_i^{\mathrm{EXP}}} = \{\langle e,\, x,\, t\rangle \mid M_e^{\Sigma_{i-1}^P} \text{ on input } x \text{ accepts in time } t\}.$$

We can also consider the set

$$D = \{\langle\langle e,\, x,\, t\rangle,\, 0^t\rangle \mid \langle e,\, x,\, t\rangle \in K_{\Sigma_i^{\mathrm{EXP}}}\}$$

which is formed by padding elements of $K_{\Sigma_i^{\mathrm{EXP}}}$ with a possibly exponentially long pad. Clearly, given an oracle Turing machine, M_1, which computes $K_{\Sigma_i^{\mathrm{EXP}}}$ then there exists a nondeterministic oracle Turing machine which accepts D . One such machine, on input $\langle\langle e,\, x,\, t\rangle,\, 0^t\rangle$, just simulates the computation of M_1 on $\langle e,\, x,\, t\rangle$ and accepts or rejects accordingly. It is easy to see that the language accepted by such a machine will be in Σ_i^P and that $\langle e,\, x,\, t\rangle \in K_{\Sigma_i^{\mathrm{EXP}}}$ if and only if $\langle\langle e,\, x,\, t\rangle,\, 0^t\rangle \in D$.

Lemma 3.1.2 *For all $i \geq 1$, $\Delta_{i+1}^{EXP[poly]} = \Delta_{i+1,tt}^{EXP} = P^{\Sigma_{\mathrm{i}}^{\mathrm{EXP}}}$.*

Proof. From Proposition 3.1.1 and Lemma 3.1.1 we have that $\Delta_{i+1}^{EXP[poly]} \subseteq \Delta_{i+1,tt}^{EXP} \subseteq P^{\Sigma_{\mathrm{i}}^{\mathrm{EXP}}}$. It remains to show that $P^{\Sigma_{\mathrm{i}}^{\mathrm{EXP}}} \subseteq \Delta_{i+1}^{EXP[poly]}$. Assume L is a language in $P^{\Sigma_{\mathrm{i}}^{\mathrm{EXP}}}$ and that M is a polynomial time oracle machine computing L. Without loss of generality, we may assume the oracle set is $K_{\Sigma_i^{\mathrm{EXP}}}$, a complete set for $\Sigma_{\mathrm{i}}^{\mathrm{EXP}}$. Define a deterministic oracle Turing machine M' as follows. Assume that D, as defined above, is the oracle set used by M'. On input w, M' simulates M on w. If during the simulation M queries $q = \langle e,\, x,\, t\rangle$ then M' queries $\langle\langle e,\, x,\, t\rangle,\, 0^t\rangle$ to D. If $\langle\langle e,\, x,\, t\rangle,\, 0^t\rangle \in D$ then M' proceeds with the simulation of M as if $\langle e,\, x,\, t\rangle$ were in $K_{\Sigma_i^{\mathrm{EXP}}}$. Otherwise, M' proceeds as if the query were not in $K_{\Sigma_i^{\mathrm{EXP}}}$. If the simulation ends in an accepting state then M' accept, otherwise M' reject. Since $\langle e,\, x,\, t\rangle \in K_{\Sigma_i^{\mathrm{EXP}}} \leftrightarrow \langle\langle e,\, x,\, t\rangle,\, 0^t\rangle \in D$ and $\langle\langle e,\, x,\, t\rangle,\, 0^t\rangle$ can be constructed in time exponential in the length of w, all queries are answered correctly in exponential time. As only polynomial many queries are made $L(M) \in \Delta_{i+1}^{EXP[poly]}$. $\square$

For example, this gives that $EXP^{NP[poly]} = EXP^{NP}_{tt} = P^{NEXP}$. Note that $P^{NEXP} = EXP^{NP}_{tt}$ is a peculiarity of these classes. In fact, since $E^{NP}_{tt} \subsetneqq EXP^{NP}_{tt}$ and $P^{NE} = P^{NEXP}$ we can conclude that $E^{NP}_{tt} \neq P^{NE}$.

Lemma 3.1.3 *For $i \geq 1$, $\Sigma_i^{EXP} = co{-}\Sigma_i^{EXP}$ implies that $\Sigma_i^{EXP} = P^{\Sigma_i^{EXP}}$.*

Proof. Assume for some i, that $\Sigma_i^{EXP} = co{-}\Sigma_i^{EXP}$. Let $L \in P^{\Sigma_i^{EXP}}$. Consider the oracle Turing machine computing L. For ever query q made to a $\Sigma_{\mathrm{i}}^{\mathrm{EXP}}$ oracle, since $\Sigma_i^{EXP} = co{-}\Sigma_i^{EXP}$, it is sufficient to guess and check a witness for q to determine the oracle answer. As this is a $\Sigma_{\mathrm{i}}^{\mathrm{EXP}}$ computation, this implies $\Sigma_i^{EXP} = P^{\Sigma_i^{EXP}}$. $\square$

Theorem 3.1.1 *Given $C = \bigcup_{f \in F} DTIME(f(n))$, where F is a fixed family of time-constructible functions,*

$$\textit{if } C \subsetneqq EXP \textit{ then } C_{tt}^{\Sigma_i^P} \neq \Sigma_i^{EXP} \textit{ for } i \geq 1.$$

Proof. Let $C = \bigcup_{f \in F} DTIME(f(n))$, where F is some fixed family of functions and $C \subseteq EXP$. Assume $C_{tt}^{\Sigma_i^P} = \Sigma_i^{EXP}$. As $C_{tt}^{\Sigma_i^P}$ is closed under complementation we have that $\Sigma_i^{EXP} = co{-}\Sigma_i^{EXP}$ which by Lemma 3.1.3 implies $\Sigma_i^{EXP} = P^{\Sigma_i^{EXP}}$. We are given by Lemma 3.1.2, that $\Delta_{i+1,tt}^{EXP} = P^{\Sigma_i^{EXP}}$ so this implies that $C_{tt}^{\Sigma_i^P} = \Delta_{i+1,tt}^{EXP}$ which is equivalent to saying $C_{tt}^{\Sigma_i^P} = EXP_{tt}^{\Sigma_i^P}$. This last statement contradicts the time hierarchy Theorem with regard to truth-table reductions. $\square$

For example, if we assume that $P^{NP}_{tt} = NEXP$ then, since P^{NP}_{tt} is closed under complementation, $NEXP = co\text{–}NEXP$. This in turn implies that $NEXP = P^{NEXP}$ by Lemma 3.1.3. Since, by Lemma 3.1.2, $P^{NEXP} = EXP^{NP}_{tt}$ we conclude that $P^{NP}_{tt} = EXP^{NP}_{tt}$ which contradicts the time hierarchy theorem with respect to truth-table reductions, therefore $P^{NP}_{tt} \subsetneqq NEXP$.

As was already mentioned, Fu, Li and Zhong give a result which implies that $P^{NP[n^{o(1)}]} \not\subseteq NEXP$ [FLZ92]. As $P^{NP}_{tt} = P^{NP[\log n]}$ and $\log n$ is contained in $n^{o(1)}$ for all n, then it is already known that $P^{NP}_{tt} \subsetneq NEXP$. In fact Fu, Li and Zhong show that $P^{NPL}_{tt} \subsetneq NEXP$ which is also implied by our theorem [1]. To see this consider that by a simple proof in which the length of queries is padded we know that $P^{NPL}_{tt} \subseteq PL^{NP}_{tt}$. Then from Theorem 3.1.1 we get that $PL^{NP}_{tt} \subsetneq NEXP$ which implies that $P^{NPL}_{tt} \subsetneq NEXP$.

In the proof of Theorem 3.1.1 we use that if $\Sigma_{\mathrm{i}}^{\mathrm{EXP}} = co-\Sigma_{\mathrm{i}}^{\mathrm{EXP}}$ then this implies that $\Sigma_{\mathrm{i}}^{\mathrm{EXP}} = \Delta^{EXP}_{i+1,tt}$. As Lemma 3.1.2 shows that, for any i, $\Delta^{EXP}_{i,tt} = \Delta_i^{EXP[poly]}$ then we can also use that if $\Sigma_{\mathrm{i}}^{\mathrm{EXP}} = co-\Sigma_{\mathrm{i}}^{\mathrm{EXP}}$ then $\Sigma_{\mathrm{i}}^{\mathrm{EXP}} = \Delta_{i+1}^{EXP[poly]}$. In fact, using this observation Theorem 3.1.1 can be improved to show that for any fixed constant c, $\Delta_{i+1}^{P[n^c]}$ is properly contained in the Σ_i^{EXP} level of $EXPH$. To do this a general hierarchy theorem which separates relativized classes while also considering the number of queries made to the oracle is needed.

Theorem 3.1.2 *Let t_1, t_2 be time constructible functions, where for all n , $t_2(n) \leq t_1(n)$; $\inf_{n\to\infty} \frac{t_3(n)\log t_3(n)}{t_1(n)} = 0$ and , for all n, $t_4(n) \leq t_2(n)$ then, for any oracle $A[f(n)]$, where $f(n)$ is the number of queries made to A for any input length n, $DTIME(t_1(n),\ A[t_2(n)])$ contains a language which is not in $DTIME(t_3(n),\ A[t_4(n)])$.*

Proof. We construct via diagonalization a set L in $DTIME(t_1(n),\ A[t_2(n)])$ which is not in $DTIME(t_3(n),\ A[t_4(n)])$. For x in $\{0,1\}^*$, let M_x denote an oracle Turing machine which has x as its Gödel number. Without loss of generality, we give a proof for oracle machines on input alphabet $\{0,1\}$.

Fix an oracle set A. We construct an oracle TM, M, that runs in time $t_1(n)$ and makes at most $t_2(n)$ queries to A and disagrees on at least one input with any $t_3(n)$ time bounded oracle TM making at most $t_4(n)$ queries to A.

[1] $NPL = \bigcup_{c\geq 0} NTIME(2^{\log^c n})$.

On input w, M constructs both a $t_1(n)$ and a $t_2(n)$ counter, where $n = |w|$. As both these functions are fully time constructible so this is possible. Now M simulates M_w on input w. If during the simulation M_w queries an oracle on some query q then M queries A on q and proceeds with the simulation based on the result of the query to A. Since M has a fixed number of tapes and tape symbols then time $c * t_3(n) \log t_3(n)$, where c is a constant, is needed to complete the simulation.

In order to insure that the simulation of M_w is t_1 time bounded the $t_1(n)$ counter is decremented with each step of the simulation. Likewise, the $t_2(n)$ counter is decremented with each oracle query. After $t_1(n)$ steps M halts and accepts only if the simulation of M_w on w is completed; no more than $t_2(n)$ queries were made to an oracle and M_w rejects w.

So w is in $L(M,\ A[t_2(n)])$ if and only if w is not in $L(M_w,\ A[t_4(n)])$. Thus $L(M,\ A) \neq L(M_w,\ A)$ for any M_w that is $t_3(n)$ time bounded and queries A at most $t_4(n)$ times. $\Box$

Corollary 3.1.1

1. $P^{NP[n^c]} \subsetneq EXP^{NP[poly]}$ *for any fixed constant* c.

2. $P^{NP[lin]} \subsetneq EXP^{NP[poly]}$.

Theorem 3.1.3 *Given* $C = \bigcup_{f \in F} DTIME(f(n))$, *where* F *is a fixed family of time-constructible functions,*

$$\text{if } C \subsetneq EXP \text{ then, for any fixed constant } c, C^{\Sigma_i^P[n^c]} \neq \Sigma_i^{EXP} \text{ for } i \geq 1.$$

Proof. Let $C = \bigcup_{f \in F} DTIME(f(n))$, where F is a fixed family of time-constructible functions, and $C \subsetneq EXP$. Fix a constant c. Assume that $C^{\Sigma_i^P[n^c]} = \Sigma_i^{EXP}$. Since $C^{\Sigma_i^P[n^c]}$ is closed under complementation then $\Sigma_i^{\mathrm{EXP}} = co{-}\Sigma_i^{\mathrm{EXP}}$ which in turn, by

Lemmas 3.1.3 and 3.1.2, implies $\Sigma_{\mathrm{i}}^{\mathrm{EXP}} = \Delta_{i+1}^{EXP[poly]}$. Now $C^{\Sigma_i^P[n^c]} = EXP^{\Sigma_i^P[poly]}$. Since $C^{\Sigma_i^P[n^c]} \subseteq EXP^{\Sigma_i^P[n^c]} \subseteq EXP^{\Sigma_i^P[poly]}$ this implies that $C^{\Sigma_i^P[n^c]} = EXP^{\Sigma_i^P[n^c]}$ which contradicts Theorem 3.1.2. $\square$

Corollary 3.1.2

1. *$P^{NP[n^c]} \subsetneqq NEXP$ for any fixed constant c.*
2. *$P^{NP[lin]} \subsetneqq NEXP$.*
3. *$PL^{NP[lin]} \subsetneqq NEXP$.*

This improves all previous results.

Considering just the polynomial time hierarchy and $EXPH$ the next question is: Can this result be improved? In [Hel86] Heller gives an oracle A such that $NP^{NP^A} = EXP^{NP^A}$. Consequently any result which shows that $NP^{NP} \subsetneqq NEXP$ will not relativize. This leaves as an open question the relationship of P^{NP} and $NEXP$. There is an easy oracle such that $\Delta_{i+1}^{PA} \neq EXP^A$ for $i \geq 1$ but an oracle showing that Δ_{i+1}^P and Σ_i^{EXP} are equal appears to be difficult. To see that there is an oracle A such that $\Delta_i^{PA} \neq EXP^A$ consider using the set $K_{\Sigma_1^{EXP}}$, a complete set for $NEXP$, as an oracle. Let $A = K_{\Sigma_1^{EXP}}$. Due to the collapse of the strong exponential hierarchy $\Delta_i^{PA} = P^A$ for any i. This gives $P^A \subsetneqq EXP^A$ by the relativization of the time hierarchy theorem.

In related work, Beigel examines the relationship of P^{NP} to the probabilistic class PP [Bei92]. This relates to our discussion as $PP \subseteq PSPACE$. In his paper Beigel gives an oracle relative to which P^{NP} is not contained in PP. In fact he shows that if $f(n) \notin O(\log n)$ then there is an A such that $P^{NP^A[f(n)]}$ is not contained in PP^A. Previously Beigel, Hemachandra and Wechsung showed that $P^{NP[\log]} \subseteq PP$ [BHW91].

3.2 A General Hierarchy Theorem for Alternating Machines with Bounded Alternations

In the proof of the next theorem we use one alternating Turing machine to simulate the computation of another alternating Turing machine. In [PPR78], it is shown that every language that is accepted by a k-tape, $t(n)$ time-bounded ATM is accepted by a 1-tape, $t(n)$ time-bounded ATM, but, as this proof relies on the use of additional alternations, we will use a 2-tape simulation that does not introduce additional alternations. In [CS76], the authors claim that if $L \in ATIME(t(n))$, then L is accepted by a 2-tape alternating TM within time $ct(n)$. This proof is similar to the corresponding proof for nondeterministic time-bounded Turing machines [BG70]. Chandra and Stockmeyer also claim that standard diagonalization arguments give complexity hierarchies for ATM's. Under the assumption that $t_1(n)$ and $t_2(n)$ are countable and $\inf_{n\to\infty} \frac{t_2(n)}{t_1(n)} = 0$ then $ATIME(t_1(n)) - ATIME(t_2(n)) \neq \emptyset$. Next we give a similar theorem but we also bound the number of alternations. Intuitively, as keeping track of alternations is a matter of keeping track of which type of state the simulation is currently in this alone should not change the simulation.

Theorem 3.2.1 *Let* t_1 *be a time constructible function, and* $\inf_{n\to\infty} \frac{t_2(n)}{t_1(n)} = 0$. *Then for any constant* $k \geq 1$ $A\Sigma_k^{t_1}$ *contains a language which is not in* $A\Pi_k^{t_2}$.

Proof. We construct via diagonalization a set L in $A\Sigma_k^{t_1}$ which is not in $A\Pi_k^{t_2}$ For x in $\{0,\ 1\}^*$, let M_x denote a alternating Turing machine which has x as its Gödel number. We will assume that if $L = L(M_x)$ for some x, then there are encodings of machines which accept L infinitely often in the enumeration of machines. Without loss of generality, we give the proof for alternating machines on input alphabet $\{0,\ 1\}$.

Consider the following four-tape alternating machine M.

input $\langle w,\ x\rangle$

```
let $n$ be the length of $w$.
if $w \neq x\#y$ for some $y \in \{0, 1\}^*$, then reject
else
    1. compute the binary representation of $t_1$ and using the time constructibility
       of $t_1$, write it on tape 1. This will be used as a $t_1$ "clock".
    2. compute the binary representation of $k$ and write it on tape 2. This will be
       used as a counter, subtracting one at each state where a quantifier is alternated.
    3. If the initial configuration of $M_x$ is an existential configuration then existentially
       guess $z$ where $|z| \leq p(n)$ and decrement the counter
    4. simulate the computation of $M_x$ on $w$
            interchange the accepting and rejecting states
            interchange the existential and universal states
    if the counter $= 1$; the clock reaches 0 and the simulation accepts then accept
       (note, the simulation accepts when $M_x$ on $w$ rejects)
    else reject
```

Using the 2-tape simulation of a multitape ATM and allowing a constant c, depending on the machine M_x, which accounts for the encoding of some fixed number of symbols used by M_x we see that the simulation requires time $ct_2(n)$. For long enough w encoding M_x, $ct_2(n) \leq t_1(n)$, so M accepts within time $t_1(n)$; starts with an existential configuration and makes less than k alternations. $L(M) \in A\Sigma_k^{t_1}$.

Assume $L(M) \in A\Pi_k^{t_2}$. Then there will be an alternating machine $M_{x'}$ accepting every string in $L(M)$ within t_2 steps; k alternations and starts with a universal configuration. As $\inf_{n\to\infty} \frac{t_2(n)}{t_1(n)} = 0$ this implies that M will have enough time to complete the simulation of $M_{x'}$ on w and accept if and only if $M_{x'}$ rejects. Thus $L(M) \neq L(M_{x'})$ contradicting our assumption. $L(M)$ is in $A\Sigma_k^{t_1}$ but is not in $A\Pi_k^{t_2}$.

□

Corollary 3.2.1 *For all constants k, $\Sigma_k^P \neq \Sigma_k^{EXP}$ and $\Pi_k^P \neq \Pi_k^{EXP}$.*

Proof. We will show the proof for $\Sigma_k^P \neq \Sigma_k^{EXP}$. Assume $\Sigma_k^{EXP} = \Sigma_k^P$. Given $\Sigma_k^P \subseteq EXP \subseteq \Sigma_k^{EXP}$, then, by assumption, $\Sigma_k^P = EXP$. This implies that Σ_k^P is closed under complementation. Since $\Sigma_k^P = \Pi_k^P$ and $\Sigma_k^P = \Sigma_k^{EXP}$ then $\Pi_k^P = \Sigma_k^{EXP}$ which contradicts Theorem 3.2.1. □

Although it is believed that both $PH \subsetneq NEXP$ and $EXP = NEXP \Rightarrow NEXP = EXPH$ there is a relativized result in terms of NE and EH contradicting the later statment. By Hartmanis, Immerman and Sewelson [HIS85] there is an oracle A such that $E^A = NE^A$ but $\Sigma_2^{E^A} \neq NE^A$. If, in fact, $EXP = NEXP \Rightarrow NEXP = EXPH$ then by the theorem just proved this would imply that $PH \subsetneq NEXP$.

Corollary 3.2.2 *If $EXP = NEXP \Rightarrow NEXP = EXPH$ then the polynomial hierarchy is properly contained in $NEXP$.*

Proof. Assume $EXP = NEXP \Rightarrow NEXP = EXPH$. Assume that $NEXP \subseteq \Sigma_k^P$ for some k. Since $\Sigma_k^P \subseteq EXP \subseteq NEXP$ this implies $EXP = NEXP$. By the first assumption $NEXP = EXPH$ so $\Sigma_k^{EXP} = NEXP = \Sigma_k^P$ contradicting Corollary 3.2.1. □

Let PL, (NPL) denote the class of problems computable in deterministic (non-deterministic) $2^{polylog}$ time.

Letting $F = \{2^{\log^k n} \mid k \in N\}$ we obtain the subexponential hierarchy based on $2^{polylog}$ length alternating computations. Call this hierarchy PLH. This hierarchy obeys downward separation: if $\Sigma_{i+1}^{PL} \subseteq \Sigma_i^{PL}$ then for all $j > i$, $\Sigma_j^{PL} \subseteq \Sigma_i^{PL}$.

Corollary 3.2.3 *For all $i \geq 0$, $NPL \neq \Sigma_i^P \bigcup \Pi_i^P$.*

Proof. Assume that for some i, $NPL = \Sigma_i^P \bigcup \Pi_i^P$. Then as $\Sigma_i^P \bigcup \Pi_i^P$ is closed under complementation, $NPL = co-NPL$ and PLH collapses to NPL, but then $\Sigma_i^{PL} \bigcup \Pi_i^{PL} = \Sigma_i^P \bigcup \Pi_i^P$. □

This corollary is not new. A direct result of the fact that for all oracles A, $P^A \neq PL^A$ is that $NPL \neq P^{\Sigma_i^P}$ for $i \geq 0$. Assume $NPL = P^{\Sigma_i^P}$. Then again NPL is closed under complementation which implies $NPL = PLH$. This in turn implies $P^{\Sigma_i^P} = PL^{\Sigma_i^P}$ which contradicts the relativized version of the time hierarch theorem.

Theorem 3.2.1 can also be stated in terms of space bounded alternating Turing machines. In [CS76], it is stated that if $L \in ASPACE(S(n))$, then L is accepted by a 1-tape alternating TM within space $s(n)$.

Theorem 3.2.2 *Let s_1, s_2 be space constructible functions, where $s_1(n) \geq \log n$ and $\inf_{n\to\infty} \frac{s_2(n)}{s_1(n)} = 0$, then, for any constant k, $A\Sigma_k^{s_1}$ contains a language which is not in $A\Pi_k^{s_2}$.*

Chapter 4

The Exponential Hierarchy and Advice Classes

4.1 BPP, P^{NP} and $EXPH$

Until about a decade ago, the notion of tractable problems was synonymous to the notion of polynomial time computable, or the class P. As randomized algorithms became more central, the class BPP also became associated with the notion of tractable. In this case *tractable* means the capability of determining set membership correctly with arbitrarily high probability in polynomial time. In many ways the questions that are of interest concerning P and NP are as enticing when phrased in terms of BPP. As this thesis is primarily concerned with the relationships between exponential time-bounded classes and lower classes it is natural to examine the relationship of exponential classes to BPP. It is known that $P \subseteq BPP \subseteq \Sigma_2^P \bigcap \Pi_2^P \subseteq EXP$ but even though $P \neq EXP$ none of these containments is known to be proper. A long standing open question concerning the relationship of BPP to EXP:

Is BPP properly contained in EXP?

motivates the results presented in this chapter.

We are also interested in the relationship of BPP to $NEXP$. Even though the classes NP and BPP may be incomparable (see [Sch85] for a discussion on this) the relationship of BPP to $NEXP$ is still of interest. It is know that BPP is contained in $NEXP$ but, unlike NP, this containment is not known to be proper. Therefore, we also consider the question:

Is BPP properly contained in $NEXP$?

Since $BPP \subseteq \Sigma_2^P \bigcap \Pi_2^P$ [Sip83, Lau83] the motivating questions in this chapter are similar to the question addressed in Chapter 3, namely: Is P^{NP} properly contained in $NEXP$? Both BPP and P^{NP} are located below $\Sigma_2^P \bigcap \Pi_2^P$ but there is no inclusion relationship known between these two classes . On the one hand, it is unlikely that $P^{NP} \subseteq BPP$. If $P^{NP} \subseteq BPP$ then $NP \subseteq P/poly$ which implies $PH = \Sigma_2^P$ [Sch85]. This is not generally believed to be true. The consequences of $BPP \subseteq P^{NP}$ are less clear. Stockmeyer [Sto83] has constructed an oracle A such that $BPP^A \not\subseteq P^{NP^A}$. One crucial difference between the questions about BPP and the question about P^{NP} is that there is an oracle A such that $BPP^A = \Delta_2^{EXP^A}$ [Hel86], hence $BPP^A = NEXP^A$, whereas, there is no know oracle result which states that P^{NP} is equal to $NEXP$.

In this chapter, although ultimately we are concerned with the relationship of BPP to EXP and $NEXP$, we do not directly consider BPP but the nonuniform class $P/poly$. As every set in BPP is in $P/poly$ [Sch85] we can restate our questions in terms of the relationship of the classes EXP and $NEXP$ to the nonuniform class $P/poly$.

Is $EXP \subseteq P/poly$?

Is $NEXP \subseteq NP/poly$?

Wilson has constructed oracles relative to which EXP^{NP} (and hence EXP and $NEXP$) is in $P/poly$ [Wil85]. So we cannot answer these questions using proof techniques that relativize. Along this same line of inquery Lutz and Mayordomo have proved related results using measure theoretic techniques. They show that almost every language in E, and almost every language in EXP, is statistically unpredictable by feasible deterministic algorithms, even with linear nonuniform advice [LM93]. In this chapter we give, using techniques that do relativize, several results which address these last two questions.

The main result of this section is that $EXP \not\subseteq DTIME(2^{O(n^{c_1})})/n^{c_2}$, for fixed integers c_1 and c_2. This implies that $EXP \not\subseteq E/lin$. The results in this chapter were done jointly with Steve Homer.

4.2 Separating EXP from Advice Classes

This section will use a variation on a proof technique that was used by Fu to show that EXP is not Turing reducible to a sparse set where the reduction is restricted to at most n^a queries for $a < 1$ [Fu93]. Formally, Fu showed that $EXP \not\subseteq R_{n^a-T}(SPARSE)$, for $a < 1$. This proof is based on resource-bounded Kolmogorov complexity.

The following is a standard definition for Kolmogorov time-bounded complexity [BDG90]. Fix any reasonable universal transducer, U. For any string x, let $U(x)$ denote the output of U on input x.

Definition 10 *The Kolmogorov time-bounded complexity set is*

$$\begin{aligned} K[f,\ g] \quad = \quad & \{u \mid \exists w(|w| \leq f(|u|)), U(w) = u \textit{ and this result is obtained in at most} \\ & g(|u|) \textit{ steps } \}. \end{aligned}$$

So $K[f,\ g]$ is the set of strings u each of which can be retrieved from some string w that represents a compression of u by a factor of $f(|u|)$ and for which the retrieval process takes time $g(|u|)$. We assume that the reader is familiar with the fundamental results in Kolmogorov complexity.

We will need to divide some string x into substrings each of length k. Formally, for any string $x \in \Sigma^*$ and integer n if $x = x_1 x_2 \ldots x_k$, for some integer k, and if, for $1 \leq i \leq k$, $|x_i| = n$ then we say that x_i is a *n-block* of x.

Theorem 4.2.1 *$EXP \not\subseteq DTIME(2^{O(n^{c_1})})/n^{c_2}$ for any fixed integers c_1 and c_2.*

The intuition behind the proof is the following. First we assume that $EXP \subseteq DTIME(2^{O(n^{c_1})})/n^{c_2}$. Then we construct a set A in $DTIME(2^{3m})$, where $m = n^{3+c_1 c_2}$. Clearly $A \in EXP$. The set of strings in A of length n, A_n, is constructed by selecting a string, τ, of length m which is not in $K[m-1, 2^{2m}]$, then breaking this string up into n-blocks. The n-blocks are the elements of A_n. So strings in A_n are n length segments of a sting τ which has *high* Kolmogorov complexity. Under the assumption that $EXP \subseteq DTIME(2^{O(n^{c_1})})/n^{c_2}$, and hence $A \subseteq DTIME(2^{O(n^{c_1})})/n^{c_2}$, we show that there is a string ξ in $K[m-1, 2^{2m}]$ from which τ can be generated. This contradicts the choice of τ and therefore our initial assumption.

Proof. Assume $EXP \subseteq DTIME(2^{O(n^{c_1})})/n^{c_2}$ for some fixed integers c_1 and c_2. Construct the set A as follows:

```
Stage $n$
    Let $m = n^{3+c_1 c_2}$
    Choose the first string $\tau \in \Sigma^m$ such that $\tau \notin K[m-1,\ 2^{2m}]$
    Set $A_n = \{w_i \mid w_i$ is a $n$-block of $\tau\}$
end of stage $n$
```

Let $A = \bigcup_{n=1}^{\infty} A_n$.

Claim 4.2.1 $A \in DTIME(2^{3n^{3+c_1 c_2}})$.

Proof of Claim. We need to find the first string τ of length m which is not in $K[m-1,\ 2^{2m}]$. This can be done by simulating $U(z)$ on all $z \in \Sigma^{m-1}$ for 2^{2m} steps and then lexicographically listing all resulting strings of length m. The first string in Σ^m that is not in the listing is τ. The n-blocks of τ then give the elements of A_n. In this way $x \in A_n$ can be decided in time $2^{m-1}2^{2m} + O(2^{2m})$. If n is large enough, then $2^{m-1}2^{2m} + O(2^{2m}) \leq 2^{3m} = 2^{3n^{3+c_1 c_2}}$. $\square$

By assumption $A \in DTIME(2^{O(n^{c_1})})/n^{c_2}$ so by definition there exists an advice function f, which on inputs of length n outputs an advice string of length n^{c_2}, and a $B \in DTIME(2^{dn^{c_1}})$, for some fixed integer d, such that

$$x \in A \leftrightarrow \langle x,\ f(|x|)\rangle \in B.$$

Let the string $\alpha_1\alpha_2\cdots\alpha_{n^{c_2}}$, where $\alpha_i \in \{0,1\}$, be the advice for inputs of length n.

Given the string $\alpha_1\alpha_2\cdots\alpha_{n^{c_2}}$ we can generate all of the strings in A_n in time $2^{n^{c_1 c_2+1}}$ as follows. Let M be a Turing machine which computes B. For each string u in Σ^n simulate the computation of M on $\langle u,\ \alpha_1\cdots\alpha_{n^{c_2}}\rangle$. If M accepts then u is in A, otherwise u is in $\bar{A}$.

The length of $\langle u,\ \alpha_1\cdots\alpha_{n^{c_2}}\rangle$ in terms of n is $2(n^{c_2}+n+2)$ allowing for pairing. There are 2^n strings to check and the simulation for each string takes at most time $2^{d(2n^{c_2}+2n+4)^{c_1}}$ and so for all strings $2^n 2^{d(2n^{c_2}+2n+4)^{c_1}} < 2^{n^{c_1 c_2+1}}$ for large enough n. Therefore, given $\alpha_1\alpha_2\cdots\alpha_{n^{c_2}}$ we can generate all of the n-blocks of τ by first generating all of the strings in A_n in time $2^{n^{c_1 c_2+1}}$.

Now we can show that $\tau \in K[m-1,\ 2^{2m}]$. What we need are an encoding of the advice $\alpha_1\alpha_2\cdots\alpha_{n^{c_2}}$, from which we can generate all of the n-blocks of τ, and an encoding of the order in which the n-blocks appear in τ.

Let $code(x) = 1a_1 1a_2\cdots 1a_j$, where $x = a_1a_2\cdots a_j$.

Set $\beta = code(\alpha_1\alpha_2\cdots\alpha_{n^{c_2}})$. Clearly $\alpha_1\alpha_2\cdots\alpha_{n^{c_2}}$ can be retrieved from β in $2*|\beta|$

steps and the length of β is $2n^{c_2}$.

Let LEX be a lexicographic listing of all of the n-blocks that form strings in A_n. Let $z_1,\ z_2,\ \ldots,\ z_{n^{2+c_1c_2}}$ be such that z_i is the position in LEX of the i^{th} n-block of τ.

Let $\gamma = code(z_1,\ z_2,\ \ldots,\ z_{n^{2+c_1c_2}})$. Clearly, $z_1,\ z_2,\ \ldots,\ z_{n^{2+c_1c_2}}$ can be retrived from γ in $2 * |\gamma|$ steps. The length of γ is $2(n^{2+c_1c_2}\log(n^{2+c_1c_2}) + 0^{\log n^{2+c_1c_2}})$ where we also account for the encoding of a list.

Let $\xi = \beta 00\gamma$. So $|\xi| = 2n^{c_2} + 2(n^{2+c_1c_2}\log(n^{2+c_1c_2}) + 0^{\log n^{2+c_1c_2}}) + 2$ which is less than m. This means that, for sufficiently large n, there is a string with length less than m that generates τ. Now we need to show that this can be done in less than or equal to 2^{2m} steps.

Given ξ we can retrieve τ in time 2^{2m} as follows. If we have ξ then in a linear amount of time we can get both $\alpha_1\alpha_2\cdots\alpha_{n^{c_2}}$ and $z_1,\ z_2,\ \ldots,\ z_{n^{2+c_1c_2}}$. Given the advice $\alpha_1\alpha_2\cdots\alpha_{n^{c_2}}$ we can generate all n-blocks in A_n in lexicographical order in $2^{n^{c_1c_2+2}}$ steps and construct LEX. From LEX and $z_1,\ z_2,\ \ldots,\ z_{n^{2+c_1c_2}}$ the string τ can be generated in $2^{n^{c_1c_2+3}}$ steps. and for large values of n, $2^{n^{c_1c_2+3}} \leq 2^{2m}$.

Therefore, for large enough n, τ is generated by a string of length $< m$ in 2^{2m} steps. So $\tau \in K[m-1,\ 2^{2m}]$ contradicting the requirement for constructing A_n. $\square$

Corollary 4.2.1

1. *$EXP \not\subseteq P/n^c$, for a fixed c.*

2. *$EXP \not\subseteq E/lin$.*

3. *$NEXP \not\subseteq E/lin$.*

It was already known that $EXP \not\subseteq P/log$ as this would imply $P = NP$ which in turn implies that $P = PH = EXP$ [KL80].

Observing that the set A which is constructed in the proof of Theorem 4.2.1 is sparse we have the following corollary.

Corollary 4.2.2 *There exists a sparse set $S \in EXP$ such that for fixed constants c_1 and c_2, $S \notin DTIME(2^{O(n^{c_1})})/n^{c_2}$.*

In [PY93] the class $LOGSNP$ is defined and several problems are shown to be complete for this class. Also it is shown that $LOGSNP \subseteq NP[\log^2 n] \bigcap DSPACE(\log^2 n)$. The class $NP[\log^2 n]$ is the subclass of NP problems in which only the first $\log^2 n$ steps are nondeterministic. Kintala and Fischer [KF80] and Diáz and Torán [DT90] studied classes that use polylogarithmic nondeterminism.

Corollary 4.2.3 *$EXP \not\subseteq NP[\log^2 n]/n^{c_1}$.*

Proof. Clearly $NP[\log^2 n] \subseteq DTIME(\log(2^{\log^2 n} p(n))p(n)2^{\log^2 n})$, where p is a polynomial, since there are at most $2^{\log^2 n}$ possible paths in the entire computation tree of any problem in $NP[\log^2 n]$ and the length of each of these paths is at most polynomial in the length of n. Since, for large enough n, $\log(2^{\log^2 n}p(n))p(n)2^{\log^2 n} \leq 2^{n^{c_2}}$, for some fixed constant c_2, then $NP[\log^2 n]/n^{c_1} \subseteq DTIME(2^{n^{c_2}})/n^{c_1}$. This implies, by Theorem 4.2.1, that $EXP \not\subseteq NP[\log^2 n]/n^{c_1}$. $\square$

Theorem 4.2.2 *$E \not\subseteq DTIME(2^{c_1 n})/c_2 n$ for fixed integers c_1 and c_2.*

With care, the proof of Theorem 4.2.1 can be modified to prove theorem 4.2.2.

4.3 Separating EXP^{NP} and EXP_{tt}^{NP} from Advice Classes

Next, using relativized resource-bounded Kolmogorov complexity, we present a relativized version of Theorem 4.2.1. We can assume that a universal machine used to measure Kolmogorov complexity can query an oracle. Let $K^A[f,\ g]$ denote the set of strings u such that there exists a word w, $|w| \leq f(|u|)$, so that U with oracle A on input w outputs u and this result is obtained in at most $g(|u|)$ steps.

Theorem 4.3.1 *For any oracle D, $EXP^D \not\subseteq DTIME(2^{O(n^{c_1})},\ D)/n^{c_2}$ for any fixed integers c_1 and c_2.*

Proof Outline. Assume for some fixed oracle set D and for fixed integers c_1 and c_2, $EXP^D \subseteq DTIME(2^{O(n^{c_1})},\ D)/n^{c_2}$. Construct the set A as follows:

```
Stage $n$
    Let $m = n^{3+c_1c_2}$
    Choose the first string $\tau \in \Sigma^m$ such that $\tau \notin K^D[m-1,\ 2^{2m}]$
    Set $A_n = \{w_i \mid w_i$ is a $n$-block of $\tau\}$
end of stage $n$
```

Let $A = \bigcup_{n=1}^{\infty} A_n$.

Claim 4.3.1 $A \in DTIME(2^{3n^{3+c_1c_2}},\ D)$.

Proof of Claim. As in the proof of Theorem 4.2.1 we need to find the first string τ of length m which is not in $K^D[m-1,\ 2^{2m}]$. This can be done by simulating $U(z)$ with D as an oracle on all $z \in \Sigma^{m-1}$ for 2^{2m} steps and then lexicographically listing all resulting strings of length m. The first string in Σ^m that is not in the listing is τ. The n-blocks of τ then give the elements of A_n. In this way $x \in A_n$ can be decided in time $2^{(m-1)}2^{2m} + O(2^{2m})$. If n is large enough, then $2^{(m-1)}2^{2m} + O(2^{2m}) \leq 2^{3m}$. $\square$

By assumption $A \in DTIME(2^{O(n^{c_1})},\ D)/n^{c_2}$ so by definition there exists an advice function f, which on inputs of length n outputs an advice string of length n^{c_2}, and a $B \in DTIME(2^{dn^{c_1}},\ D)$, for some fixed integer d, such that

$$x \in A \leftrightarrow \langle x,\ f(|x|)\rangle \in B.$$

Let the string $\alpha_1\alpha_2\cdots\alpha_{n^{c_2}}$, where $\alpha_i \in \{0,1\}$, be the advice for inputs of length n.

Given the string $\alpha_1\alpha_2\cdots\alpha_{n^{c_2}}$ we can generate all of the strings in A_n in time $2^{n^{c_1 c_2+1}}$ using an oracle Turning machine and oracle set D as follows. Let M be an oracle Turing machine which runs in deterministic time $2^{dn^{c_1}}$ and using D as an oracle set computes B. For each string u in Σ^n simulate the computation of M with D on $\langle u,\ \alpha_1\cdots\alpha_{n^{c_2}}\rangle$. If M with D accepts then u is in A, otherwise u is in $\bar{A}$.

The length of $\langle u,\ \alpha_1\cdots\alpha_{n^{c_2}}\rangle$ in terms of n is $2(n^{c_2}+n+2)$ allowing for pairing. There are 2^n strings to check and the simulation for each string takes at most time $2^{d(2n^{c_2}+2n+4)^{c_1}}$ and so for all strings $2^n 2^{d(2n^{c_2}+2n+4)^{c_1}} < 2^{n^{c_1 c_2+1}}$ for large enough n. Therefore, given $\alpha_1\alpha_2\cdots\alpha_{n^{c_2}}$ we can generate all of the n-blocks of τ by first generating all of the strings in A_n in time $2^{n^{c_1 c_2+1}}$ using oracle D.

Now we can show that $\tau \in K^D[m-1,\ 2^{2m}]$. What we need are an encoding of the advice $\alpha_1\alpha_2\cdots\alpha_{n^{c_2}}$, from which we can generate all of the n-blocks of τ, and an encoding of the order in which the n-blocks appear in τ.

Refer to the proof of Theorem 4.2.1 for the details of this portion of the proof.

Therefore, for large enough n, τ is generated by a string of length $< m$ in 2^{2m} steps by an oracle Turing machine using D as an oracle. So $\tau \in K^D[m-1,\ 2^{2m}]$ contradicting the requirement for constructing A_n. $\square$

Corollary 4.3.1

1. *$EXP^{NP} \not\subseteq NP/n^c$, for any fixed integer c.*

2. *$EXP^{NP} \not\subseteq P^{NP}/n^c$, for any fixed integer c.*

3. *$EXP^{NP} \not\subseteq E^{NP}/n^c$, for any fixed integer c.*

In a related paper, Buhrman and Homer show that if $EXP^{NP} \subseteq EXP/poly$ then $EXP^{NP} = EXP$ and that if $EXP^{NP} \subseteq P/poly$ then $EXP = \Sigma_2^P \bigcup \Pi_2^P$ [BH92].

The proof of Theorem 4.3.1 can be modified to show that the same result holds with reguard to truth-table reductions, $EXP_{tt}^{D} \not\subseteq DTIME(2^{0(n^{c_1})},\ D,\ tt)/n^{c}$ [1]. First we restrict the universal machine used to measure the Kolmogorov complexity so that all queries are made in one step of the computation. We will denote this by $K_{tt}^{D}[f,\ g]$, where D is an oracle set. Then there are two portions of the proof of Theorem 4.3.1 that need to be considered. The first is the claim that A is in $DTIME(2^{3n^{2+c}},\ D)$. It must be shown that all queries are made to the oracle in one computation step so that $A \in EXP_{tt}^{D}$. Secondly we must show that under the assumption that $EXP_{tt}^{D} \subseteq DTIME(2^{0(n^{c_1})},\ D,\ tt)/n^{c}$, the elements of A_n can be decided in $DTIME(2^{n^{c_1 c_2+1}},\ D)$ by a machine that makes all queries in one computation step. With a little thought it can be seen that both propositions hold. The general outline of this proof is given next.

Theorem 4.3.2 *For any set D, $EXP_{tt}^{D} \not\subseteq DTIME(2^{0(n^{c_1})},\ D,\ tt)/n^{c}$, for fixed integers c_1 and c_2.*

Proof Outline. Restrict the oracle Turing machine, U, which measures the Kolmogorov complexity so that it makes all queries in one computation step. Assume for some fixed oracle set D, $EXP_{tt}^{D} \not\subseteq DTIME(2^{0(n^{c_1})},\ D,\ tt)/n^{c}$, for fixed integers c_1 and c_2. Construct the set A as follows:

```
Stage $n$
    Let $m = n^{2+c}$
    Choose the first string $\tau \in \Sigma^m$ such that $\tau \notin K_{tt}^{D}[m-1,\ 2^{2m}]$
    Set $A_n = \{w_i \mid \text{ each } w_i \text{ is a } n\text{-block of } \tau\}$
end of stage $n$
```

Let $A = \bigcup_{n=1}^{\infty} A_n$.

[1] $DTIME(2^{0(n^{c_1})},\ D,\ tt)$ is the set of all languages in $DTIME(2^{0(n^{c_1})},\ D)$ that are accepted by oracle Turing machines which make only nonadaptive queries.

Claim 4.3.2 $A \in EXP_{tt}^{D}$.

Proof of Claim. The proof of this is the same the proof of Claim 4.3.1 except, on every string $z \in \Sigma^{m-1}$, we first simulate U on z up to the step where D is queried on some string $w = \langle w_1, w_2, \cdots w_m \rangle$. At this point we save the query string w. When all such queries have been collected then U queries D on all queries at once. Since all query answers are now known then, for each $z \in \Sigma^{m-1}$, the simulation of U on z is repeated using the correct query answers. In this way A is accepted by an oracle Turing machine that makes all queries to D in one computation step. Since this procedure only increases the time of the procedure used in the proof of Claim 4.3.1 by a constant factor then $A_n \in EXP_{tt}^{D}$. □

By assumption $A \in DTIME(2^{0(n^{c_1})},\ D,\ tt)/n^c$, so by definition there exists an advice function f, which on inputs of length n outputs an advice string of length n^c, and a $B \in DTIME(2^{dn^{c_1}},\ D,\ tt)$, for a fixed integer d such that

$$x \in A \leftrightarrow \langle x,\ f(|x|) \rangle \in B.$$

Let the string $\alpha_1 \alpha_2 \cdots \alpha_{n^c}$ be the advice for inputs of length n.

Given the advice string $\alpha_1 \alpha_2 \cdots \alpha_{n^c}$ we can generate all of the strings in A_n in $DTIME(2^{n^{c_1 c_2 + 1}},\ D,\ tt)$ as follows. Let M be an oracle Turing machine which runs in deterministic time $2^{dn^{c_1}}$ and using D as an oracle set computes B. We are given that M makes all queries to D in one computation step.

As in the proof of Theorem 4.3.1, for each string u in Σ^n we simulate the computation of M with D on $\langle u,\ \alpha_1 \cdots \alpha_{n^{c_2}} \rangle$ and if M with D accepts then u is in A, otherwise u is in $\bar{A}$. Now need to modify this so that all queries are made in one computation step. Notice, as in the proof of Claim 4.3.2, all of the queries can be collected into a list of queries and then made in one computation step.

First, given the advice string, we simulate M on u up to the step where D is queried on some string $w = \langle w_1, w_2, \cdots w_m \rangle$. At this point we save the query string w. When all such queries have been collected then M queries D on all queries at once. Since all query answers are now known then, for each $u \in \Sigma^n$, the simulation of M on u and the advice string, is repeated using the correct query answers. In this way A_n is generated by an oracle Turing machine that makes all queries to D in one computation step. Since this procedure only increases the time of analogous procedure used in the proof of Theorem 4.3.1 by a constant, generating the strings in A_n takes time $DTIME(2^{n^{c_1 c_2 + 1}},\ D,\ tt)$.

Now we can show that $\tau \in K^D_{tt}[m-1,\ 2^{2m}]$. What we need are an encoding of the advice $\alpha_1 \alpha_2 \cdots \alpha_{n^{c_2}}$, from which we can generate all of the n-blocks of τ, and an encoding of the order in which the n-blocks appear in τ.

Refer to the proof of Theorem 4.2.1 for the details of this portion of the proof.

Therefore, for large enough n, τ is generated by a string of length $< m$ in 2^{2m} steps by an oracle Turing machine using D as an oracle. So $\tau \in K^D_{tt}[m-1,\ 2^{2m}]$ contradicting the requirement for constructing A_n. $\square$

Theorem 4.3.2 gives that $EXP^{NP}_{tt} \not\subseteq P^{NP}_{tt}/n^c$ for fixed integer c.

4.4 Considering Sparse Sets

Theorem 4.3.1 implies $EXP^{NP} \not\subseteq NP/n^c$, for any fixed integer c. It is well known that $NP/poly = NP^S$ where S is sparse. We show that it is probably the case that $NP/n^c \subsetneq NP^{S[1]}$. It is easy to see that $NP^S = NP^{S[1]}$.

Proposition 4.4.1 $NP^S = NP^{S[1]}$, *where S is the set of all sparse sets.*

Proof. That $NP^{S[1]} \subseteq NP^S$ is obvious. To see that $NP^S \subseteq NP^{S[1]}$ consider any set $A \in NP^{S_1}$ via a nondeterministic oracle Turing machine M using a fixed sparse oracle S_1. Now we can construct a sparse oracle S_2, such that $S_2 = \{\langle\langle w_{1_1}, \ldots, w_{i_1}\rangle, \ldots, \langle w_{1_m}, \ldots w_{j_m}\rangle\rangle \mid$ for $i, j \geq 0$ and $m \geq 1$, if $\langle w_{1_n}, w_{2_n}, \ldots, w_{i_n}\rangle$ are the stings of length n in $S_1\}$. Now S_2 contains at most one string of any length and that string is polynomial in the length of m. Let the longest query made during any computation path of M on x have length m. So to determine if $x \in A$ we can first guess all of the strings in S_1 up to and including some length m and then query S_2 on this guess. If the guess is correct then we can simulate M on x using the correct query answers otherwise we reject. Since $|m|$ is at most a polynomial in $|x|$ this can be done in polynomial time and $A \in NP^{S_2[1]}$. $\square$

Since is probably true that $NP/n^c \subsetneqq NP/poly$, we conjecture that $NP/n^c \subsetneqq NP^{S[1]}$.

Chapter 5

The Boolean Hierarchy over $NEXP$

Kadin [Kad88] showed that for all constants k, if $P^{NP[k]} = co\text{–}P^{NP[k]}$ then $NP/poly = co-NP/poly$. $NP/poly$ and $co-NP/poly$ are the nonuniform analogies of NP and $co-NP$. Kadin's result combined with a result by Yap [Yap83] shows that for all constants $k \geq 1$, $P^{NP[k]} = co\text{–}P^{NP[k]}$ implies $PH = \Sigma_3^P$. In this chapter we show that if the Boolean hierarchy over $NEXP$ collapses, then $NEXP/poly = co\text{–}NEXP/poly$. Because the classes of the EXP hierarchy are defined in terms of the polynomial hierarchy and not in terms of the exponential hierarchy the techniques which are used to collapse the polynomial hierarchy do not seem to apply to the exponential time hierarchy. Yap's proof that $NP/poly = co\text{–}NP/poly$ implies $PH = \Sigma_3^P$ exploits characteristics of the polynomial hierarchy that do not appear to generalize to the exponential time case. Consequently, the result which we obtain for the exponential time Boolean hierarchy is not as strong as Kadin's result for the polynomial time Boolean hierarchy. The question which pertains to this chapter arises because we would like to generalize both Kadin's result and Yap's result to the exponential time hierarchy. Namely we would like to know:

Does $co-NEXP \subseteq NEXP/poly$ imply that the exponential time hierarchy collapse?

It should be noted that in [Loz92], Lozano observes that a consequence of a theorem presented in his thesis indirectly proves a result which is similar to our result. His theorem implies that if the Boolean hierarchy over $NEXP$ collapses, then $co-NEXP \subseteq NEXP/poly$.

The question motivating this chapter is closely related to the general open question: Does a collapse of lower levels of $EXPH$ or EH imply a collapse of higher levels? In the case of EH, there is an oracle A such that $E^A = NE^A$ but $NE^A \neq \Sigma_2^{E^A}$ [HIS85]. This means that nonrelativizing results will be need to show this type of collapse.

One the other hand, there are exponential hierarchies that in fact do collapse. For example, the strong exponential time hierarchies collapses without assumption. This collapse also implies that $E^{NE} = NE^{NE}$. In [HIS85] it is shown that for any A, there are sparse sets in $NP^A - P^A$ if and only if $E^A \neq NE^A$. Letting A be $K_{\Sigma_1^{EXP}}$, a complete set for NE, we see by the collapse of SEH to P^{NE} that $NP^A = P^A$, so, in fact, $E^{NE} = NE^{NE}$. This alone does not imply that $NE^{NE^{NE}} = NE^{NE}$. As noted by Hartmanis, Immerman and Sewelson, the collapse of the polynomial time hierarchy, under the assumption that $P = NP$, may be a peculiarity of the polynomial time hierarchy and not the case for all time hierarchies.

5.1 The Collapses of $EXPBH$

First we give a complete set for D^E. For brevity, we use $\langle e,\ x,\ t\rangle_i$ to represent the triple $\langle e_i,\ x_i,\ t_i\rangle$.

Lemma 5.1.1 *$K_{\Sigma_1^{EXP}} \wedge \overline{K_{\Sigma_1^{EXP}}}$ is complete for D^E.*

Proof. Let

$$L_1 = \{(\langle e,\, x,\, t\rangle_1,\, \langle e,\, x,\, t\rangle_2) \mid \langle e,\, x,\, t\rangle_1 \in K_{\Sigma_1^{EXP}}\}$$

$$L_2 = \{(\langle e,\, x,\, t\rangle_3,\, \langle e,\, x,\, t\rangle_4) \mid \langle e,\, x,\, t\rangle_4 \in K_{\Sigma_1^{EXP}}\}.$$

and let $z_i = \langle e,\, x,\, t\rangle_i$. Then $L_1 - L_2 = \{(z_1, z_2) - (z_3, z_4) \mid z_1 \in K_{\Sigma_1^{EXP}}$ and $z_4 \in K_{\Sigma_1^{EXP}}\}$ but that is $K_{\Sigma_1^{EXP}} \wedge \overline{K_{\Sigma_1^{EXP}}}$.

For all $L \in D^E$, $L \leq_m^{\mathrm{P}} K_{\Sigma_1^{EXP}} \wedge \overline{K_{\Sigma_1^{EXP}}}$. Let L be some set in D^E. Then for some L_1, L_2 in $NEXP$ we get $L = L_1 \cap \overline{L_2}$. Let M_1, M_2 be the machines accepting L_1, L_2 resp.. The reduction, on input x, produces the pair $(\langle M_1,\, x,\, t\rangle,\, \langle M_2,\, x,\, t\rangle)$. By the definition of D^E and $K_{\Sigma_1^{EXP}} \wedge \overline{K_{\Sigma_1^{EXP}}}$, $x \in L$ if and only if $\langle M_1,\, x,\, t\rangle \in K_{\Sigma_1^{EXP}}$ and $\langle M_2,\, x,\, t\rangle \in \overline{K_{\Sigma_1^{EXP}}}$. □

Following the same argument that Kadin used to show that $D^P = co{-}D^P \Rightarrow NP/poly = co{-}NP/poly$ we show a similar result for $D^E = co{-}D^E$.

Theorem 5.1.1 *If $D^E = co{-}D^E$ then $NEXP/poly = co{-}NEXP/poly$.*

Proof. Suppose that $D^E = co{-}D^E$. Then there is a $\leq_m^{\mathrm{P}}$ reduction from $K_{\Sigma_1^{EXP}} \wedge \overline{K_{\Sigma_1^{EXP}}}$ to $\overline{K_{\Sigma_1^{EXP}}} \vee K_{\Sigma_1^{EXP}}$ since $K_{\Sigma_1^{EXP}} \wedge \overline{K_{\Sigma_1^{EXP}}} \in D^E$ and $\overline{K_{\Sigma_1^{EXP}}} \vee K_{\Sigma_1^{EXP}}$ is $\leq_m^{\mathrm{P}}$ complete for $co{-}D^E$. Then , for all x,

$$x \in K_{\Sigma_1^{EXP}} \wedge \overline{K_{\Sigma_1^{EXP}}} \iff h(x) \in \overline{K_{\Sigma_1^{EXP}}} \vee K_{\Sigma_1^{EXP}}.$$

As x is a pair of tuples we can rewrite this as, for all $(\langle e,\, x,\, t\rangle_1,\, \langle e,\, x,\, t\rangle_2)$,

$\langle e,\, x,\, t\rangle_1 \in K_{\Sigma_1^{EXP}}$ and $\langle e,\, x,\, t\rangle_2 \in \overline{K_{\Sigma_1^{EXP}}} \iff \langle e,\, x,\, t\rangle_3 \in \overline{K_{\Sigma_1^{EXP}}}$ or $\langle e,\, x,\, t\rangle_4 \in K_{\Sigma_1^{EXP}}$,

where $\langle e,\, x,\, t\rangle_3 = \pi_1(h(\langle e,\, x,\, t\rangle_1,\, \langle e,\, x,\, t\rangle_2))$ and $\langle e,\, x,\, t\rangle_4 = \pi_2(h(\langle e,\, x,\, t\rangle_1,\, \langle e,\, x,\, t\rangle_2))$

.

Define a triple $\langle e,\, x,\, t\rangle$ to be *easy* if

$$\langle e,\, x,\, t\rangle \in \overline{K_{\Sigma_1^{EXP}}} \text{ and } \exists y,\ |y| = |\langle e,\, x,\, t\rangle|, \text{ such that } \langle e,\, x,\, t\rangle_2 \in K_{\Sigma_1^{EXP}}$$

where $\langle e,\ x,\ t\rangle_2 = \pi_2(h(y,\ \langle e,\ x,\ t\rangle))$. If all strings in $\overline{K_{\Sigma_1^{EXP}}}$ are easy then the following $NEXP$ machine, N_{easy}, recognizes $co-NEXP$. On input $\langle e,\ x,\ t\rangle$,

1. Guess y with $|y| = |\langle e,\ x,\ t\rangle|$.
2. Compute $\langle e,\ x,\ t\rangle_2 = \pi_2(h(y,\ \langle e,\ x,\ t\rangle))$.
3. Guess an accepting computation path of M_e on x, z.
4. Accept if z is an accepting computation path.

Define a triple H to be *hard* if it is not easy. Let $n = |\langle e,\ x,\ t\rangle|$. So,

$$H \in \overline{K_{\Sigma_1^{EXP}}} \text{ and } \forall y,\ |y| = n,\ \langle e,\ x,\ t\rangle_2 \notin K_{\Sigma_1^{EXP}}$$

where $\langle e,\ x,\ t\rangle_2 = \pi_2(h(y,\ H))$.

$$\langle e,\ x,\ t\rangle_1 \in K_{\Sigma_1^{EXP}} \text{ and } H \in \overline{K_{\Sigma_1^{EXP}}} \iff \langle e,\ x,\ t\rangle_3 \in \overline{K_{\Sigma_1^{EXP}}} \text{ or } \langle e,\ x,\ t\rangle_4 \in K_{\Sigma_1^{EXP}},$$

where $\langle e,\ x,\ t\rangle_3 = \pi_1(h(\langle e,\ x,\ t\rangle_1, H))$ and $\langle e,\ x,\ t\rangle_4 = \pi_2(h(\langle e,\ x,\ t\rangle_1, H))$. As we know $H \in \overline{K_{\Sigma_1^{EXP}}}$ and $\langle e,\ x,\ t\rangle_4 \notin K_{\Sigma_1^{EXP}}$ so for all $\langle e,\ x,\ t\rangle_1$, $|\langle e,\ x,\ t\rangle| = n$,

$$\langle e,\ x,\ t\rangle_1 \in K_{\Sigma_1^{EXP}} \iff \langle e,\ x,\ t\rangle_3 \in \overline{K_{\Sigma_1^{EXP}}}.$$

Negating both sides,

$$\langle e,\ x,\ t\rangle_1 \in \overline{K_{\Sigma_1^{EXP}}} \iff \langle e,\ x,\ t\rangle_3 \in K_{\Sigma_1^{EXP}}.$$

The following $NEXP$ machine, N_{hard}, recognizes triples of length n in *co*–$NEXP$. On input $(\langle e,\ x,\ t\rangle_1,\ H)$, where H is a hard string,

1. If $|\langle e,\ x,\ t\rangle_1| = n$, reject.
2. Compute $\langle e,\ x,\ t\rangle_3 = \pi_2(h(y,\ H))$.
3. Guess an accepting computation path of M_e on x, z.

4. Accept if z is an accepting computation path for $\langle e,\, x,\, t\rangle_3$.

If H is a hard string then N_{hard} accept if and only if $|\langle e,\, x,\, t\rangle_1| = |H|$ and $\langle e,\, x,\, t\rangle_1 \in \overline{K_{\Sigma_1^{EXP}}}$. Clearly, if $\langle e,\, x,\, t\rangle_1 \notin K_{\Sigma_1^{EXP}}$ then $(\langle e,\, x,\, t\rangle_1,\ H) \notin K_{\Sigma_1^{EXP}} \wedge \overline{K_{\Sigma_1^{EXP}}}$. So $\langle e,\, x,\, t\rangle_3 \in K_{\Sigma_1^{EXP}}$ by definition of h and $N_{hard}(\langle e,\, x,\, t\rangle_1,\ H)$ accepts. On the other hand, if $\langle e,\, x,\, t\rangle_1 \in K_{\Sigma_1^{EXP}}$, then $(\langle e,\, x,\, t\rangle_1,\ H) K_{\Sigma_1^{EXP}} \wedge \overline{K_{\Sigma_1^{EXP}}}$. So, either $\langle e,\, x,\, t\rangle_3 \in \overline{K_{\Sigma_1^{EXP}}}$ or $\langle e,\, x,\, t\rangle_2 \in K_{\Sigma_1^{EXP}}$. However, H is a hard string, so $\langle e,\, x,\, t\rangle_2 \notin K_{\Sigma_1^{EXP}}$. Thus $\langle e,\, x,\, t\rangle_3 \in \overline{K_{\Sigma_1^{EXP}}}$ and $N_{hard}(\langle e,\, x,\, t\rangle_1,\ H)$ rejects.

To compute the above algorithm in nondeterministic exponential time all we need is a hard string H, if one exists, or to know that no such string of length n exists. Clearly an advice function can, on input 1^n either provide H or indicate that all stings length n are easy. Hence $co{-}NEXP \subseteq NEXP/poly$ and $NEXP/poly = co{-}NEXP/poly$. □

Chapter 6

On Function Classes

The question which motivates this chapter is:

> Can the structure of function classes be used to show separations between classes of sets?

And more specifically, will an examination of function classes help in answering the following question:

Is $P^{NP[\log]}$ properly contained in EXP?

We can't hope to answer this question with techniques which relativize as there is an oracle A such that $NP^A = EXP^A$ [Dek76]. Surprisingly, using techniques which relativize, we show that the corresponding function classes, $PF^{NP[\log]}$ and $EXPF_{PB}$, are not equal. In this chapter we show that:

For any $i \geq 1$, $PF^{\Sigma_i^P[\log]}$ is properly contained in $EXPF_{PB}$.

This gives evidence that the structure of the polynomial hierarchy is quite different from the structure of the polynomial hierarchy defined via function classes, PHF. Specifically, we believe that the function classes

$$PF^{NP[\log]}, PF^{NP^{NP}[\log]}, PF^{NP^{NP^{NP}}[\log]}, \ldots$$

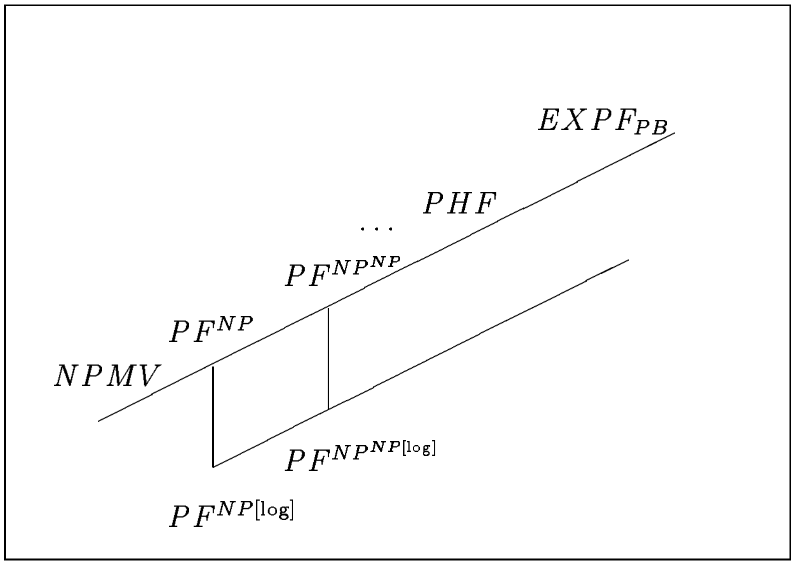

Figure 1: Polynomial time function hierarchies.

are not interleaved with the classes

$$PF^{NP}, PF^{NP^{NP}}, PF^{NP^{NP^{NP}}}, \dots$$

in the same way in which the corresponding classes of sets are. In fact, there is reason to believe that the hierarchy of function classes in which oracle access is restricted to log queries is contained in PF^{NP}. We come to this conclusion as, for $i \geq 1$, $PF^{\Sigma_i^P} \subseteq PF^{\Sigma_{i+1}^P[\log]}$ implies that $P = NP$ and further if $PF^{NP} \subseteq PF^{\Sigma_i^P[\log]}$ for any i then $P = NP$. The proof of these last two statements is implicit in Krentel's proof that if $PF^{NP} \subseteq PF^{NP[\log]}$ then $P = NP$ [Kre88]. Figure 1 shows the known relationships between these function classes.

The class $PF^{NP[\log]}$ has been well studied and was shown to contains the following interesting complete problems [Kre88]:

(i) given a CNF boolean formula,

computing the maximum number of simultaneously satisfiable clauses

(ii) given a graph G

(a) computing the size of the largest clique in G,

(b) computing the chromatic number of G

(c) computing the length of the longest cycle in G.

Also a particularly detailed study of the finer structure of the polynomial-time hierarchy of functions below PF^{NP} was carried out by Beigel [Bei88]. Krentel [Kre88] showed that if $f(n) \leq \frac{1}{2}$ then $PF^{NP[f(n)-1]} \subsetneq PF^{NP[f(n)]}$ unless $P = NP$. Beigel [Bei91] shows that this is true for larger values of $f(n)$ unless certain natural complexity classes, which we expect to differ, coincide.

We begin this chapter by giving some general results about exponentially time-bounded function classes. As we are primarily interested in relating exponentially time-bounded function classes to polynomial time-bounded function classes, the exponential time classes that we consider are restricted to those which contain only partial functions that are polynomial-bounded; i.e., for a function $f(x) = y$, $|y|$ is bounded by some polynomial in $|x|$. Next we study the relationship of PHF to $EXPF_{PB}$ showing that $PF^{\Sigma_i^P} \subsetneq EXPF_{PB}$ for all i. In Section 6.3 we discuss the relationship of the class PF^{NEXP} to function classes which have an exponential time bound. Last, we show that the results of Chapters 3 and 4 are still true, when applied to the corresponding function classes.

6.1 Some Basic Observations

First we need to establish some simple relationships between function classes. For any single-valued function f, *code(f)* is defined by Selman [Sel92] to be the set of all triples (σ, x, k), where $\sigma \in \{0,1\}$, $x \in \Sigma^*$ and k is the binary representation of a number, such that: $(0, x, k) \in code(f)$ if and only if $f(x)$ has a k^{th} bit and $(1, x, k) \in code(f)$ if and only if the k^{th} bit of $f(x)$ is a 1.

Selman has shown that if f is a single-valued function, then $f \in PF^A \Leftrightarrow \mathit{code}(f) \in P^A$ [Sel92]. The proof from right to left follows from the fact that the bits of $f(x)$ can be retrieved by repeatedly computing $(0, x, k) \in \mathit{code}(f)$ and $(1, x, k) \in \mathit{code}(f)$. As $|f(x)| \leq p(|x|)$, for some polynomial p, this procedure can be done in polynomial time.

It follows directly that if f is a single-valued function, then $f \in EXPF^A \Leftrightarrow \mathit{code}(f) \in EXP^A$.

Further, if f is a single-valued function, then $f \in EXPF_{PB}^{A[poly]} \Leftrightarrow \mathit{code}(f) \in EXP^{A[poly]}$. The proof is the same as for $EXPF^A$ and EXP^A: we only need to observe that it only takes a polynomial number of queries to determine a polynomial length output.

The next proposition allows us to infer relationships about function classes based on relationships between classes of sets.

Proposition 6.1.1 (Generalizing Selman [Sel92]) *For any single valued function f and any deterministic time or space-bounded complexity classes C_1, C_2, if $f \in C_1F \Rightarrow \mathit{code}(f) \in C_1$; $\mathit{code}(f) \in C_2 \Rightarrow f \in C_2F$ and $C_1 \subseteq C_2$ then $C_1F \subseteq C_2F$.*

Proof. Let classes C_1 and C_2 be defined as there are above with $f \in C_1F \Rightarrow \mathit{code}(f) \in C_1$; $\mathit{code}(f) \in C_2 \Rightarrow f \in C_2F$ and $C_1 \subseteq C_2$. Consider any partial function $f_1 \in C_1F$. Since $\mathit{code}(f_1) \in C_1$ and $C_1 \subseteq C_2$ then $\mathit{code}(f_1) \in C_2$ so $f_1 \in C_2F$. $\square$

Let C be any of the complexity classes PH, $PSPACE$ or EXP, then $\mathit{code}(f) \in C \Rightarrow f \in CF$, where CF is PHF, $PSPACEF$ or $EXPF_{PB}$. Suppose that for some partial function f and polynomial p, such that $|f(x)| \leq p(|x|)$, $\mathit{code}(f) \in C$ via a Turing machine M. Now in an obvious way we can retrieve the k^{th} bit of $f(x)$, if it exists, by simulating M on the inputs $(0, x, k)$ and $(1, x, k)$. As the length of $f(x)$ is bounded by $p(|x|)$ then to retrieve $f(x)$ we simulate M at most $p(|x|) * 2$ times. This implies that $f \in CF$.

Note that it is not known if $code(f) \in P^{NP[\log]} \Rightarrow f \in PF^{NP[\log]}$. The above argument does not succeed. Using the above argument, more than $\log |x|$ queries would be made to the oracle. Further, since we show that $PF^{NP[\log]} \subsetneqq EXPF_{PB}$ then, if $code(f) \in P^{NP[\log]} \Rightarrow f \in PF^{NP[\log]}$ were true we could conclude that $P^{NP[\log]} \subsetneqq EXP$. As there is an oracle relative to which $P^{NP[\log]}$ and EXP are equal we can conclude that a nonrelativizing proof is needed to show that $code(f) \in P^{NP[\log]} \Rightarrow f \in PF^{NP[\log]}$.

Some of the consequences of Proposition 6.1.1 are that $PHF \subseteq PSPACEF \subseteq EXPF_{PB}$. Since $PH \subseteq PSPACE$ and for any partial function $f \in PF^{\Sigma_i^P}$, for any i, $code(f) \in P^{\Sigma_i^P}$ we conclude that $PHF \subseteq PSPACEF$. Likewise, as $PSPACE \subseteq EXP$ and for all f in $PSPACEF$, $code(f) \in PSPACE$ so $PSPACEF \subseteq EXPF_{PB}$. This last statement is restricted to exponential functions which are polynomial bounded as we assume that functions in $PSPACEF$ are polynomial-bounded.

6.2 Separating $EXPF_{PB}$ From $PF^{\Sigma_i^P[\log]}$

Although it is not known if $P^{NP[\log]}$ is properly contained in EXP we show that $PF^{NP[\log]}$ is properly contained in the class of exponentially time bounded partial functions in which the length of the function values are at most polynomially greater than the length of the input. If this result is improved to show that the length of the function values are bounded by a constant; i.e., $PF_{CB}^{NP[\log]} \subsetneqq EXPF_{CB}$, then this would imply that $P^{NP[\log]}$ is properly contained in EXP.

The idea that Krentel [Kre88] used to show that $PF^{NP[O(\log n)]} = PF^{NP} \Rightarrow P = NP$ will be repeatedly used in this chapter so we will review his argument next. First, Krentel defined the function, MSA (Maximum Satisfying Assignment), over the domain of boolean formulas as follows:

$$\begin{aligned} MSA(\phi(x_1, \ldots, x_n)) &= \{\text{the lexicographically maximum } x_1, \ldots, x_n \in \{0,1\}^n \\ &\qquad \text{that satisfies } \phi, \text{ or } 0 \text{ if } \phi \text{ is unsatisfiable}\} \end{aligned}$$

and showed that it is complete for PF^{NP}. Given this, his proof that $PF^{NP[O(\log n)]} = PF^{NP} \Rightarrow P = NP$ is as follows. Assume that $PF^{NP[O(\log n)]} = PF^{NP}$. As $MSA \in PF^{NP[O(\log n)]}$ there is a PF^{SAT} machine, M, that computes MSA making at most $O(\log n)$ queries. Now to determine if a formula ϕ is satisfiable, simulate $M(\phi)$ for all possible oracle answers. This gives a polynomial number of possible assignments, at least one of which is a satisfying assignment if $\phi \in SAT$.

Proposition 6.2.1 (Generalizing Krentel [Kre88]) *For any class C and oracle set A, if $C \subseteq PF^{A[\log]}$ and $MSA \in C$ then $P = NP$.*

The proof of this follows directly from Krentel's proof that $PF^{NP[O(\log n)]} = PF^{NP} \Rightarrow P = NP$. Note that we are only considering oracle sets which decide set membership. On the other hand, if we consider $A \in NPMV$ then the proof of Proposition 6.2.1 fails.

Lemma 6.2.1 *For $i \geq 1$, $EXPF_{PB} \subseteq PF^{\Sigma_i^P[\log]} \Rightarrow P = NP$.*

Proof. Assume that, for $i \geq 1$, $EXPF_{PB} \subseteq PF^{\Sigma_i^P[\log]}$. As $PHF \subseteq EXPF_{PB}$ then $PF^{\Sigma_i^P} \subseteq EXPF_{PB}$, for all i. Since $PF^{\Sigma_i^P[\log]} \subseteq PF^{\Sigma_i^P}$ then, by assumption, $PF^{\Sigma_i^P[\log]} = PF^{\Sigma_i^P}$. Clearly, as $MSA \in PF^{\Sigma_1^P}$ then, for all $i \geq 1$, $MSA \in PF^{\Sigma_i^P}$. By Proposition 6.2.1 this implies that $P = NP$. □

Proposition 6.2.2 *For $i \geq 1$, $EXPF_{PB} \subseteq PF^{\Sigma_i^P[\log]} \Rightarrow EXP \subseteq P^{\Sigma_i^P[\log]}$.*

Proof. Assume $EXPF_{PB} \subseteq PF^{\Sigma_i^P[\log]}$, for some i. Let $L \in EXP$. This implies that the characteristic function for L is in $EXPF_{PB}$ so, by assumption, the characteristic function for L is in $PF^{\Sigma_i^P[\log]}$. But then if follows that L is in $P^{\Sigma_i^P[\log]}$. □

Theorem 6.2.1 *For all i, $PF^{\Sigma_i^P[\log]} \neq EXPF_{PB}$.*

Proof. We will show that $EXPF_{PB} \subseteq PF^{\Sigma_i^P[\log]} \Rightarrow P = EXP$. Assume $PF^{\Sigma_i^P[\log]} = EXPF_{PB}$, for some i. As $PF^{\Sigma_i^P[\log]} \subseteq PF^{\Sigma_i^P} \subseteq EXPF_{PB}$ then by Lemma 6.2.1,

$$EXPF_{PB} \subseteq PF^{\Sigma_i^P[\log]} \Rightarrow P = NP.$$

Since $P = NP$ then by the collapse of PH, $P = P^{\Sigma_i^P[\log]}$. By Proposition 6.2.2,

$$EXPF_{PB} \subseteq PF^{\Sigma_i^P[\log]} \Rightarrow EXP \subseteq P^{\Sigma_i^P[\log]}.$$

Combining the above two lines we get that $P^{\Sigma_i^P[\log]} = P = EXP$ contradicting the time hierarchy theorem. □

Corollary 6.2.1 $PF^{NP[\log]} \subsetneqq EXPF_{PB}$.

This does not result in PHF being properly contained in $EXPF_{PB}$. Unlike the polynomial hierarchy, it does not appear that $PF^{\Sigma_i^P} \subseteq PF^{\Sigma_{i+1}^P[\log]}$ for any i as this would imply $P = NP$. In fact, it may be that, for every i, $PF^{\Sigma_i^P[\log]}$ is properly contained in PF^{NP}. From this we can conjecture that the polynomial hierarchy over function classes has a very different structure then the polynomial hierarchy over sets.

It is not clear what the relationship is between the function classes $PF^{\Sigma_i^P[\log]}$, $i \geq 1$, and the nondeterministic classes $NPSV$ and $NPMV$. From Selman we know that if either $PF^{NP[\log]} \subseteq NPSV$ or $PF^{NP[\log]} \subseteq NPMV$ then $NP = co\text{–}NP$ [Sel92].

Clearly both of these results also show that, for $i \geq 1$, if either $PF^{\Sigma_i^P[\log]} \subseteq NPSV$ or $PF^{\Sigma_i^P[\log]} \subseteq NPMV$ then $NP = co-NP$ because $PF^{NP[\log]} \subseteq PF^{\Sigma_i^P[\log]}$ for all $i \geq 1$. Examining the reverse containment, Selman showed that if $NPMV \subseteq_c PF^{NP[\log]}$ then $P = NP$ and if $NPSV \subseteq PF^{NP[\log]}$ then $P = UP$. The proof that if $NPMV \subseteq_c PF^{NP[\log]}$ then $P = NP$ can easily be generalized to show that, for $i \geq 1$, if $NPMV \subseteq_c PF^{\Sigma_i^P[\log]}$ then $P = NP$. To see this assume that a $PF^{\Sigma_i^P[\log]}$ machine M can output a satisfying assignment for a formula; i.e., can compute a partial function in $NPMV$. Then a polynomial time machine can simulate M on all possible queries and determine if some computation path ends in a satisfying assignment.

Unfortunately, $code(f)$ can not be used directly to show that $PF^{\Sigma_i^P[\log]} \subsetneq EXPF_{PB}$ implies $P^{\Sigma_i^P[\log]} \subsetneq EXP$. If our result is improved to show that $PF_{CB}^{NP[\log]} \subsetneq EXPF_{CB}$ then we will get the desired consequence. This result would once again not relativize.

Theorem 6.2.1 $PF_{CB}^{NP[\log]} \subsetneq EXPF_{CB} \Rightarrow P^{NP[\log]} \subsetneq EXP$.

Proof. We will show that $P^{NP[\log]} = EXP \Rightarrow PF_{CB}^{NP[\log]} = EXPF_{CB}$. Assume $P^{NP[\log]} = EXP$. We know that for all f in $EXPF_{CB}$, $code(f) \in EXP$ so by assumption $code(f) \in P^{NP[\log]}$. But then using $code(f)$ we can retrieve $f(x)$ with a $PF^{NP[\log]}$ machine. To retrieve one bit of $f(n)$ a $P^{NP[\log]}$ computation is performed at most twice. To retrieve all the bits of $f(n)$ a $P^{NP[\log]}$ computation is performed at most $2 * c$ times, where $|f(n)| = c$. Thus $f \in PF_{CB}^{NP[\log]}$ and $PF_{CB}^{NP[\log]} = EXPF_{CB}$. $\square$

Again using Krentel's basic idea we strengthen the result that $P = PSPACE$ if and only if $PF = PSPACEF$.

Theorem 6.2.2 *For all i, $P \neq PSPACE \Leftrightarrow PF^{\Sigma_i^P[\log]} \neq PSPACEF$.*

Proof. From right to left is trivial. For the converse, we show that if $PSPACEF \subseteq PF^{\Sigma_i^P[\log]}$ then $P = PSPACE$. Assume $PSPACEF \subseteq PF^{\Sigma_i^P[\log]}$. Since $MSA \in PSPACE$, by Proposition 6.2.1 $P = NP$ which gives that $P = PH$. Once again it is easy to see that $PSPACEF \subseteq PF^{\Sigma_i^P[\log]} \Rightarrow PSPACE \subseteq P^{\Sigma_i^P[\log]}$ as the characteristic function for every set in $PSPACE$ is in $PSPACEF$. Combining the collapse of the polynomial hierarchy to P and the fact that $PSPACE \subseteq P^{\Sigma_i^P[\log]}$ we get that $P = PSPACE$. □

6.3 The Relationship of PF^{NEXP} to $EXPHF$

We do not expect that $PF^{NEXP} \subseteq PF^{NEXP[\log]}$ or even $PF^{\Sigma_2^P} \subseteq PF^{NEXP[\log]}$ as if either of these statements is true then by Proposition 6.2.1 $P = NP$. In this section we show that $EXPF_{PB}^{NP[poly]} = PF^{NEXP}$.

By definition, given any partial function $f \in PF^{NEXP}$, for any x in the domain of f, $|f(x)|$ is bounded by some polynomial in $|x|$. So it is reasonable to compare the class PF^{NEXP} only with exponentially time-bounded function classes C, such that, if f is in C then f is polynomially-bounded. In fact we see that the relationship of PF^{NEXP} to $EXPF_{PB}^{NP[poly]}$ is similar to that of the corresponding classes of sets, P^{NEXP} and $EXP^{NP[poly]}$. (See Chapter 3 for a discussion of P^{NEXP}.)

Proposition 6.3.1 $EXPF_{PB}^{NP[poly]} = PF^{NEXP}$.

Proof. We show only that $EXPF_{PB}^{NP[poly]} \subseteq PF^{NEXP}$ the proof of the opposite inclusion follows the same idea. Clearly $f \in EXPF_{PB}^{NP[poly]} \leftrightarrow code(f) \in EXP^{NP[poly]}$ and $f \in PF^{NEXP} \leftrightarrow code(f) \in P^{NEXP}$. Consider a partial function $f_1 \in EXPF_{PB}^{NP[poly]}$. As $code(f_1) \in EXP^{NP[poly]}$ and from Chapter 3 we know

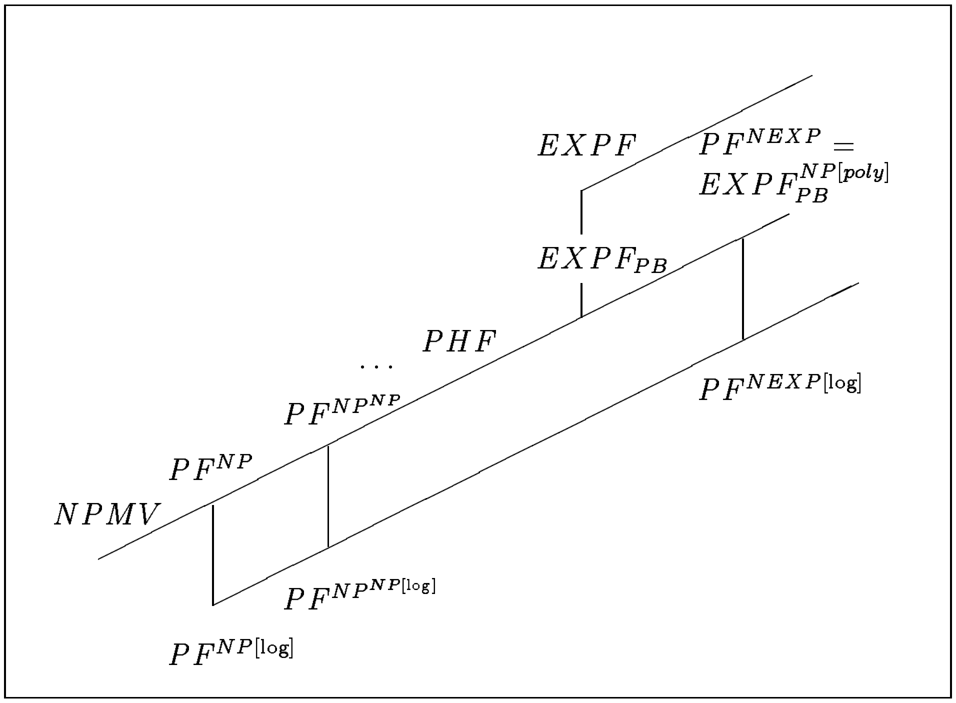

Figure 2: Polynomial time and exponential time function classes.

that $EXP^{NP[poly]} = P^{NEXP}$ then $code(f_1) \in P^{NEXP}$. But since $f_1 \in PF^{NEXP} \leftrightarrow code(f_1) \in P^{NEXP}$ then $f_1 \in PF^{NEXP}$. $\square$

So this gives that

$$EXPF_{PB} \subseteq PF^{NEXP} \subseteq EXPF_{PB}^{NP}.$$

Again, it is not likely that $EXPF_{PB} \subseteq PF^{NEXP[\log]}$ since by by Proposition 6.2.1 this implies $P = NP$.

Figure 2 illustrates the relationships between the function classes that have been discussed in this chapter.

6.4 Separations between Classes of Sets Imply Separations of Function Classes

In this section we show that the results of Chapters 3 and 4 imply similar results for the corresponding classes of functions. As these results follow directly they are stated as propositions.

In Chapter 3 we show that $P^{NP[n^c]} \subsetneq NEXP$ for any integer c.

Proposition 6.4.1 *$PF^{NP[n^c]} \subsetneq NEXPMV_{PB}$ for any integer c.*

Proof. Assume $NEXPMV_{PB} \subseteq_c PF^{NP[n^c]}$ for some integer c. This implies that the characteristic function for every set in $NEXP$ is in $PF^{NP[n^c]}$ an hence every set in $NEXP$ can be computed in $P^{NP[n^c]}$. This contradicts Theorem 3.1.3. $\square$

In Chapter 4 we show that $EXP \not\subseteq DTIME(2^{n^{c_1}})/n^{c_2}$ for any fixed integers c_1 and c_2.

Proposition 6.4.2 *$EXPF \not\subseteq DTIMEF(2^{n^{c_1}})/n^{c_2}$ for any fixed integers c_1 and c_2.*

Proof. Assume $EXPF \subseteq DTIMEF(2^{n^{c_1}})/n^{c_2}$ for some fixed integers c_1 and c_2. But then the characteristic function for every EXP set is in $DTIMEF(2^{n^{c_1}})/n^{c_2}$ an hence every set in EXP can be computed in $DTIME(2^{n^{c_1}})/n^{c_2}$. This contradicts Theorem 4.2.1. $\square$

Chapter 7

Concluding Remarks

We have seen that a closer examination of exponential time classes gives separations between exponential time classes and polynomial time classes, both uniform and nonuniform. It is apparent that these results can be improved and that it may be possible to get improvements by using techniques which relativize.

The question which directly and indirectly motivated much of the work in this thesis is: $P^{NP} \subsetneq NEXP$? This question still remains unanswered. As we showed in Chapter 3, an equivalent question is: $P^{NP} \subsetneq P^{NEXP}$? In particular, since $NP \neq NEXP$, we would assume that $P^{NP} \subsetneq P^{NEXP}$. It may be possible to answer these questions via a proof which uses techniques that relativize. There is currently no proof giving an oracle A such that $P^{NP^A} = NEXP^A$. It would be interesting to show either of these containments or to show an oracle A such that $P^{NP^A} = NEXP^A$ or equivalently $P^{NP^A} = P^{NEXP^A}$.

In Section 3.2, it should be possible to improve the Theorem 3.2.1 from *infinitely often* to an *almost everywhere*, a.e., result. In [ABHH90] Allender, Beigel, Hertrampf and Homer present an a.e. complexity hierarchy for nondeterministic time. Their result may imply the same for alternating machines.

The most interesting question raised in Chapter 5 is: What are the implications of $NEXP/poly = co-NEXP/poly$. Unfortunately, it does not appear that this will be proved with proof techniques which relativize. Gavaldà [Gav92] has bounded the complexity on advice functions for $P/poly$. Can the same be done for $NEXP/poly$?

Chapter 6 begins a discussion on function classes for exponential time. This is an area that has not been closely examined to date and therefore is open to further discoveries.

In Chapter 6 we do not directly consider the relationship between the number of queries that can be made to an oracle and the number of bits that can be output by a transducer but it appears that there may be some general relationship. In the case of $P^{NP[\log]}$ depending on whether we allow polynomial length output or restrict output to a constant number of bits we can show different consequences in PH. On a slightly different track, if PF^{NEXP} is restricted to log queries and log many output bits then does this also alter the type of results that can be stated relating to $EXPH$. A general interesting question is: How does the bound on the number of queries relate to the bound on the length of the output?

Also, as PHF and $EXPHF$ do not appear to contain the same properties as PH and $EXPH$ (for example, $P^{NP} \subseteq P^{NP^{NP}[1]}$ but the same is not true for the corresponding function classes unless $P = NP$), an area to examine is the finer structure of PHF and $EXPHF$.

We have limited our study to partial functions that are computed by time-bounded transducers with oracles in NP or $NEXP$. In [FHOS92] partial functions that are computed in polynomial time with oracles in $NPMV$ are considered. Using this type of oracle slightly different results are achieved with regard to polynomial time function classes. Likewise, the work that was begun in this chapter could be reconsidered using $NPMV$ and $NEXPMV$ as oracle sets. Clearly, this would give different results since $EXPF_{PB} \subseteq PF^{NEXPMV[1]}$. In general, comparing results obtained using both types of oracles is of interest.

Appendix

Complexity classes:

$PSPACE = \bigcup_{c\geq 0} DSPACE(n^c)$

$EXPSPACE = \bigcup_{c\geq 0} DSPACE(2^{n^c})$

$P = \bigcup_{c\geq 0} DTIME(n^c)$

$NP = \bigcup_{c\geq 0} NTIME(n^c)$

$PL = \bigcup_{c\geq 0} DTIME(2^{\log^c n})$

$NPL = \bigcup_{c\geq 0} NTIME(2^{\log^c n})$

$E = \bigcup_{c\geq 0} DTIME(2^{cn})$

$NE = \bigcup_{c\geq 0} NTIME(2^{cn})$

$EXP = \bigcup_{c\geq 0} DTIME(2^{n^c})$

$NEXP = \bigcup_{c\geq 0} NTIME(2^{n^c})$

Function classes, $f : \Sigma^*$ to Σ^*:

$log = \{f \mid f(n) = c \cdot \log_2 n$ for some constant c $\}$

$lin = \{f \mid f(n) = c \cdot n$ for some constant c $\}$

$poly = \{f \mid f(n) = c \cdot n^k$ for some constants c, k $\}$

Restricted oracle queries:

$P^{A[f(n)]}$ = the set of languages computable in P^A that on input x make at most $f(|x|)$ queries to A

$NP^{A[f(n)]}$ = the set of languages computable in NP^A that on input x make at most $f(|x|)$ queries to A

$EXP^{A[f(n)]}$ = the set of languages computable in EXP^A that on input x make at most $f(|x|)$ queries to A

$NEXP^{A[f(n)]}$ = the set of languages computable in $NEXP^A$ that on input x make at most $f(|x|)$ queries to A

P_{tt}^{NP} = the set of languages in P^{NP} in which all queries are written to the query tape before any queries are made

EXP_{tt}^{NP} = the set of languages in EXP^{NP} in which all queries are written to the query tape before any queries are made

Hierarchies

For a function f, let Σ_k^f (Π_k^f) denote the class of languages accepted by a $A\Sigma_k(A\Pi_k)$ alternating Turing machine which runs in time $f(n)$. For a class of functions F, $\Sigma_k^F = \bigcup_{f \in F} \Sigma_k^f$ and $\Pi_k^F = \bigcup_{f \in F} \Pi_k^f$.

$PH = \{$ Let F be polynomial-time computable functions $\}$

$EXPH = \{$ Let $F 2^{p(n)} \mid p$ is a polynomial $\}$

$EH = \{$ Let $F 2^{cn} \mid c$ is a constant $\}$

The i^{th} sigma levels of these hierarchies will be denoted Σ_i^P, Σ_i^{EXP} and Σ_i^E. The pi levels will be denoted Π_i^P, Π_i^{EXP} and Π_i^E. Each of these hierarchies defined inductively can also be represented using the classes usually denoted by Δ_i^F.

$PH = \Delta_i^P = P^{\Sigma_{i-1}^P}$

$EXPH = \Delta_i^{EXP} = EXP^{\Sigma_{i-1}^P}$

$E = \Delta_i^E = E^{\Sigma_{i-1}^P}$

Restricted oracle access in PH:

$\Sigma_i^{P[F]}$ is the set of languages in $NP^{\Sigma_{i-1}^P[F]}$.

$\Pi_i^{P[F]}$ is the set of languages in $co{-}NP^{\Sigma_{i-1}^P[F]}$.

$\Delta_i^{P[F]}$ is the set of languages in $P^{\Sigma_{i-1}^P[F]}$.

$\Sigma_{i,tt}^P$ is the set of languages in $NP_{tt}^{\Sigma_{i-1}^P}$.

$\Pi_{i,tt}^P$ is the set of languages in $co{-}NP_{tt}^{\Sigma_{i-1}^P}$.

$\Delta_{i,tt}^P$ is the set of languages in $P_{tt}^{\Sigma_{i-1}^P}$.

Restricted oracle access in $EXPH$:

$\Sigma_i^{EXP[F]}$ is the set of languages in $NEXP^{\Sigma_{i-1}^P[F]}$.

$\Pi_i^{EXP[F]}$ is the set of languages in $co{-}NEXP^{\Sigma_{i-1}^P[F]}$.

$\Delta_i^{EXP[F]}$ is the set of languages in $EXP^{\Sigma_{i-1}^P[F]}$.

The definitions for EH are analogous to those for $EXPH$.

Probabilistic classes and advice classes

BPP = the class of languages recognized by polynomial time probabilistic Turing machines whose error probability is bounded above by some positive constant $\epsilon < 1/2$.

R = is the class of languages recognized by polynomial time probabilistic Turing machines which have zero error probability for inputs not in the language, and error probability bounded above by some positive constant $\epsilon < 1/2$ for words in the language.

Definition 11 *An* advice *function is a function* $f : \mathbf{N} \rightarrow \Sigma^*$. *Let* C *be a complexity class, and* F *a family of advice functions. The class* C/F *is the collection of all sets* A *such that for some* $B \in C$ *and a function* $f \in F$

$$x \in A \textit{ if and only if } \langle x,\ f(|x|)\rangle \in B.$$

Boolean hierarchy for exponential time, $EXPBH$:

$EXPBH_1 = NEXP$

$EXPBH_{2i} = \{L_1 \bigcap \overline{L_2} \mid L_1 \in EXPBH_{2i-1} \text{ and } L_2 \in NEXP\}$

$EXPBH_{2i+1} = \{L_1 \bigcup L_2 \mid L_1 \in EXPBH_{2i} \text{ and } L_2 \in NEXP\}$

$co{-}EXPBH_i = \{L \mid \overline{L} \in EXPBH_i\}$

$EXPBH = \bigcup_{i \geq 1} EXPBH_i$

Function classes:

$PSPACEF$ = the set of all partial functions which can be computed in polynomial space.

PF = the set of all partial functions which can be computed deterministically by polynomial time-bounded transducer.

$NPMV$ = the set of all partial functions which can be computed nondeterministically by polynomial time-bounded transducer.

$NPSV$ = the set of all $f \in NPMV$ that are single valued.

$EXPF$ = the set of all partial functions which can be computed deterministically by an 2^{n^c} time-bounded transducer where c is a constant.

$NEXPMV$ = the set of all partial functions which can be computed nondeterministically by an 2^{n^c} time-bounded transducer where c is a constant.

$NEXPSV$ = the set of all $f \in NEXPMV$ that are single valued.

A function f is *constant-bounded* if there is a constant c such that, for all x in the domain of f, $|f(x)| \leq c$.

A function f is *polynomial-bounded* if there is a polynomial p such that, for all x in the domain of f, $|f(x)| \leq p(|x|)$.

PF_{CB} = the set of all constant-bounded functions in CF.

$EXPF_{CB}$ = the set of all constant-bounded functions in $EXPF$.

$EXPF_{PB}$ = the set of all polynomial-bounded functions in $EXPF$.

Polynomial hierarchy for function classes:

$$\Delta_0^{PF} = PF$$
$$\Delta_{i+1}^{PF} = PF^{\Sigma_i^P},\ i \geq 0$$

$$PHF = \bigcup_{i \geq 0} \Delta_i^{PF}$$

Exponential hierarchy for function classes:

$$\Delta_0^{EXPF} = EXPF$$
$$\Delta_{i+1}^{EXPF} = EXPF^{\Sigma_i^P},\ i \geq 0$$

$$EXPHF = \bigcup_{i \geq 0} \Delta_i^{EXPF}$$

Bibliography

[ABHH90] E. Allender, R. Beigel, U. Hertrampf, and S. Homer. Almost-everywhere complexity hierarchies for nondeterministic time. *in publication*, 1990.

[AM77] L. M. Adleman and K. Manders. Reducibility, randomness, and intractability. In *Proc. 9th ACM Symp. on Theory of Computing*, pages 151–163, 1977.

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity*, volume I. Springer-Verlag, New York, 1988.

[BDG90] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity*, volume II. Springer-Verlag, New York, 1990.

[Bei88] R. Beigel. NP-hard sets are P-superterse unless R = NP. Technical Report 4, Dept. of Computer Science, John Hopkins University, 1988.

[Bei91] R. Beigel. Bounded queries to SAT and the Boolean hierarchy. *Theoretical Computer Science*, 84(2):199–223, 1991.

[Bei92] R. Beigel. Perceptrons, PP, and the polynomial hierarchy. In *Proc. IEEE Structure in Complexity Theory*, pages 14–19, 1992.

[BFL90] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. In *Proc. 31th IEEE Symp. on Foundations of Computer Sci.*, pages 26–35, 1990.

[BFNW91] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. In *Proc. 6th IEEE Structure in Complexity Theory*, pages 213–219, 1991.

[BG70] R. Book and S. Greibach. Quasi-realtime languages. *Math. Systems Theory*, 4:97–111, 1970.

[BH92] H. Buhrman and S. Homer. Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In *Foundations of Software Technology and Theoretical Computer Science*, 1992.

[BHW91] R. Beigel, L. A. Hemachandra, and G. Wechsung. Probabilistic polynomial time is closed under parity reductions. *Information Processing Letters*, 37:91–94, 1991.

[Boo74a] R. V. Book. Comparing complexity classes. *Journal of Computer and System Sciences*, 9:213–229, 1974.

[Boo74b] R. V. Book. Tally languages and complexity classes. *Information and Control*, 26:186–193, 1974.

[BWX82] R. V. Book, C. Wilson, and M. Xu. Relativizing time, space and time-space. *SIAM Journal on Computing*, 11(3):571–581, 1982.

[CGH$^+$88] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. W. Wagner, and G. Wechsung. The Boolean hierarchy I: Structural properties. *SIAM Journal on Computing*, 6:1232–1252, 1988.

[CGH$^+$89] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. W. Wagner, and G. Wechsung. The Boolean hierarchy II: Structural properties. *SIAM Journal on Computing*, 7:95–111, 1989.

[CH86] J. Cai and L. Hemachandra. The Boolean hierarchy: Hardware over NP. In *Proc. IEEE Structure in Complexity Theory*, pages 105–124, 1986.

[Cha91] R. Chang. *On The Structure of NP Computation Under Boolean Operators*. PhD thesis, Computer Science Department, Cornell University, 1991.

[Chu36] A. Church. An unsolvable problem of elementary number theory. *American Journal of Math.*, 58:345–363, 1936.

[CK90] R. Chang and J. Kadin. The Boolean hierarchy and the polynomial hierarchy: A closer connection. In *Proc. 5th IEEE Struct. Complexity Theory Conf.*, pages 169–178, 1990.

[Coo71] S. A. Cook. The complexity of theorem-proving procedures. In *Proc. 3th ACM Symp. on Theory of Computing*, pages 151–158, 1971.

[Coo73] S. A. Cook. A hierarchy for nondeterministic time complexity. *Journal of Computer and System Sciences*, 7(4):343–353, 1973.

[CS76] A. K. Chandra and L. Stockmeyer. Alternation. In *Proc. 17th IEEE Symp. on Foundations of Computer Sci.*, pages 98–108, 1976.

[Dek76] M. I. Dekhtyar. *On the Relation of Deterministic and Nondeterministic Complexity Classes*, volume 45 of *Math. Foundations of Computer Science, LNCS*, pages 282–287. Springer-Verlag, New York, 1976.

[DT90] J. Díaz and J. Torán. Classes of bounded nondeterminism. *Math. Systems Theory*, 23:21–32, 1990.

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

[EPS87] S. Ephremidis, C. H. Papadimitriou, and M. Sideris. Complexity characterizations of attribute grammar languages. In *Proc. 2nd IEEE Structure in Complexity Theory*, pages 10–13, 1987.

[FHOS92] S. Fenner, S. Homer, M. Ogiwara, and A. Selman. On using oracles that compute values. In *Symposium Theoretical Aspects of Computer Science*, 1992.

[FLZ92] B. Fu, H. Li, and Y. Zhong. Some properties of exponential time complexity classes. In *Proc. IEEE Structure in Complexity Theory*, pages 50–57, 1992.

[FRS88] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. In *Proc. 3rd IEEE Structure in Complexity Theory*, pages 156–161, 1988.

[Fu93] B. Fu. With quasi-linear queries, EXP is not polynomial time Turing reducible to sparse sets. In *Proc. IEEE Structure in Complexity Theory*, pages 185–191, 1993.

[Gan90] K. Ganesan. *Complete Problems, Creative Sets and Isomorphism Conjectures.* PhD thesis, Computer Science Department, Boston University, 1990.

[Gav92] R. Gavaldà. Bounding the complexity of advice functions. In *Proc. 7th IEEE Structure in Complexity Theory*, pages 249–254, 1992.

[GH83] W. I. Gasarch and S. Homer. Relativizations comparing NP and exponential time. *Information and Control*, 58:88–100, 1983.

[Gil77] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6(4):675–695, 1977.

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

[Göd31] K. Gödel. Über formal unentscheidbare Sätze der principia mathematica und verwant der systeme, I. *Monatsheftc Math. Phys.*, 38:173–198, 1931.

[Hel86] H. Heller. On relativized exponential and probabilistic complexity classes. *Information and Control*, 71:231–243, 1986.

[Hem87] L. A. Hemachandra. *Counting in Structural Complexity Theory.* PhD thesis, Computer Science Department, Cornell University, 1987.

[HIS85] J. Hartmanis, N. Immerman, and V. Sewelson. Sparse sets in NP-P: EXPTIME versus NEXPTIME. *Information and Control*, 65:158–181, 1985.

[HLS65] J. Hartmanis, P. Lewis, and R. Stearns. Hierarchies of memory limited computation. In *Proc. 6th IEEE Symp. on Switching Circuit Theory and Logical Design.*, pages 179–190, 1965.

[HS65] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Trans. American Math. Society*, 117:285–306, 1965.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley Pub. Co., Reading, Mass., 1979.

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5):935–938, 1988.

[JOR75] M. Jazayeri, W. F. Ogden, and W. C. Rounds. The intrinsically exponential complexity of the circularity problem for attribute grammars. *Communications of the ACM*, 18:687–706, 1975.

[Kad88] J. Kadin. The polynomial time hierarchy collapses if the Boolean hierarchy collapses. *SIAM Journal on Computing*, 17(6):1263–1282, 1988.

[Kan82] R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55:40–46, 1982.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computation*, pages 85–103. Plenum Press, New York, 1972.

[Kar86] R. M. Karp. Combinatorics, complexity and randomness. *Communications of the ACM*, 29(2):103, Feb. 1986.

[KF80] C. M. R. Kintala and P. C. Fischer. Refining nondeterminism in relativized polynomial-time bounded computations. *SIAM Journal on Computing*, 9:46–53, 1980.

[KL80] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proc. 12th ACM Symposium on Theory of Computing*, pages 302–309, 1980.

[KMR90] S. A. Kurtz, S. R. Mahaney, and J. S. Royer. The structure of complete degrees. In A. Selman, editor, *Complexity Theory Retrospective*, chapter 6, pages 108–146. Springer-Verlag, New York, 1990.

[Ko82] K. Ko. Some observations on the probabilistic algorithms and NP-hard problems. *Information Processing Letters*, 14:39–43, 1982.

[Kre88] M. Krentel. The complexity of optimization problems. *Journal Computer Systems Sci.*, 36:490–509, 1988.

[KSW87] J. Köbler, U. Schöning, and K. W. Wagner. The difference and truth-table hierarchies for NP. *RAIRO*, 21:419–435, 1987.

[Lau83] C. Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17:215–217, 1983.

[Lev73] L. Levin. Universal sorting problems. *Problems Inform. Transmission*, 9:265–266, 1973.

[LM93] J. H. Lutz and E. Mayordomo. Measure, stochasticity, and the density of hard languages. *SIAM Journal on Computing*, 1993. to appear.

[Loz92] A. Lozano. *Computational Complexity versus Structural Simplicity.* PhD thesis, Department of Languages and Information Systems, University of Catalunya, 1992.

[Lut93] J. H. Lutz. The quantitative structure of exponential time. In *Proc. 8th IEEE Structure in Complexity Theory*, pages 158–175, 1993.

[PPR78] W. Paul, E. Prauss, and R. Reischuk. On alternation. In *Proc. 19th IEEE Symp. on Foundations of Computer Sci.*, pages 113–121, 1978.

[PY82] C. H. Papadimitriou and M. Yannakakis. The complexity of facets and some facets of complexity. In *Proc. 14th IEEE Symp. on the Foundations of Computer Sci.*, pages 255–260, 1982.

[PY93] C. H. Papadimitriou and M. Yannakakis. On limited nondeterminism and the complexity of the v-c dimension. In *Proc. 8th IEEE Structure in Complexity Theory*, pages 12–18, 1993.

[Rab59] M. O. Rabin. Speed of computation and classification of recursive sets. In *Third Convention Science Society*, pages 1–2, 1959.

[Rab60] M. O. Rabin. Degree of difficulty of computing a function and a partial order of recursive sets. Technical Report 2, Hebrew University, Jerusalem, 1960.

[Rit63] R. W. Ritchie. Classes of predictably computable functions. *Trans. American Mathematical Society*, 106:139–173, 1963.

[SC78] L. Stockmeyer and A. K. Chandra. Provably difficult combinatorial games. Technical Report RC-6957, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1978.

[Sch85] U. Schöning. *Complexity and Structure*, volume 211 of *Lecture Notes in Computer Science*. Springer-Verlag, New York, 1985.

[Sel92] A. Selman. A taxonomy of complexity classes of functions. *Journal Computer Systems Sci.*, page In press, 1992.

[SFM73] J. I. Seiferas, M. J. Fischer, and A. R. Meyer. Refinements of nondeterministic time and space hierarchies. In *Proc. 14th IEEE Symposium on Switching and Automata Theory*, pages 130–137, 1973.

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Proc. 15th ACM Symp. on Theory of Computing*, pages 330–335, 1983.

[Sit90] M. Sitharam. *Generalized Bounded Query Hierarchies.* PhD thesis, Computer Science Department, University of Wisconsin-Madison, 1990.

[Ste90] R. E. Stearns. Juris hartmanis: The beginnings of computational complexity. In A. Selman, editor, *Complexity Theory Retrospective*, chapter 1. Springer-Verlag, New York, 1990.

[Sto83] L. Stockmeyer. The complexity of approximate counting. In *Proc. 15th ACM Symp. on Theory of Computing*, pages 118–126, 1983.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Inform.*, 26:279–284, 1988.

[TFL91] S. Tang, B. Fu, and T. Liu. Exponential time and subexponential time sets. In *Proc. 6th IEEE Structure in Complexity Theory*, pages 230–237, 1991.

[Tur36] A. M. Turing. Computable numbers with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2):230–265, 1936.

[Wan90] J. Wang. *Polynomial Time Creativity and Its Applications.* PhD thesis, Computer Science Department, Boston University, 1990.

[Wil80] C. B. Wilson. Relativization, reducibilities and the exponential hierarchy. Technical Report 140/80, Dept. of Computer Science, University of Toronto, 1980.

[Wil85] C. B. Wilson. Relativized circuit complexity. *Journal Computer Systems Sci.*, 31:169–181, 1985.

[WW85] K. W. Wagner and G. Wechsung. On the Boolean closure of NP. In *Proc. Conf. Fundament. Computer Theory, Lecture Notes Computer Sci.*, pages 485–493, 1985.

[Yap83] C. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26(3):287–300, 1983.

Biographical Sketch

Sarah Easton Mocas was born on March 1, 1951 in Niagara Falls, New York. She received, with honors, a Bachelor of Arts degree in Computer Science in 1985 from Tufts University. In 1989 she received a Master of Science degree from the College of Computer Science, Northeastern University.