ON THE STRUCTURE OF NP COMPUTATIONS UNDER BOOLEAN OPERATORS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Richard Chang

August 1991

ON THE STRUCTURE OF NP COMPUTATIONS UNDER BOOLEAN OPERATORS

Richard Chang, Ph.D.
Cornell University 1991

This thesis is mainly concerned with the structural complexity of the Boolean Hierarchy. The Boolean Hierarchy is composed of complexity classes constructed using Boolean operators on NP computations. The thesis begins with a description of the role of the Boolean Hierarchy in the classification of the complexity of NP optimization problems. From there, the thesis goes on to motivate the basic definitions and properties of the Boolean Hierarchy. Then, these properties are shown to depend only on the closure of NP under the Boolean operators, AND_2 and OR_2.

A central theme of this thesis is the development of the hard/easy argument which shows intricate connections between the Boolean Hierarchy and the Polynomial Hierarchy. The hard/easy argument shows that the Boolean Hierarchy cannot collapse unless the Polynomial Hierarchy also collapses. The results shown in this regard are improvements over those previously shown by Kadin. Furthermore, it is shown that the hard/easy argument can be adapted for Boolean hierarchies over incomplete NP languages. That is, under the assumption that certain incomplete languages exist, the Boolean hierarchies over those languages must be proper (infinite) hierarchies. Finally, this thesis gives an application of the hard/easy argument to resolve the complexity of a natural problem — the unique satisfiability problem. This last refinement of the hard/easy argument also points out some long-ignored issues in the definition of randomized reductions.

Biographical Sketch

Richard was born in Brussels, Belgium on July 25, 1966. At age 3, he announced to his mother his intention to attend school, and thus started a sinuous 22-year career as a student. After trying Chinese and Italian, Richard finally settled on English as his primary language. However, the Minister of Education in Taiwan greatly disapproved of his choice. Fleeing from such intellectual persecution, Richard completed his High School education in Glastonbury, Connecticut. His college years took him to Potsdam, New York, where he received a Bachelor of Science in Computer Science and Mathematics from Clarkson University in May, 1986. Moving on, Richard enrolled in the Computer Science graduate program at Cornell University. There, he received a Master of Science in Computer Science in May, 1989 and met the love of his life. He married Christine Piatko on July 21, 1991 and completed his Ph.D. in August, 1991.

Acknowledgements

Since this thesis is hardly the fruit of individual labor, I would like to take this opportunity to acknowledge those who have been instrumental in its development. First, I thank my advisor Juris Hartmanis who has given me unwavering guidance and support through the years and who, by his spurrings and enticements, has finally induced me to complete this thesis.

I thank my collaborators Jim Kadin, Desh Ranjan, Pankaj Rohatgi and Suresh Chari. They have made research in complexity theory challenging, interesting and fun (sometimes even funny). Without them, research would become an unbearably lonely endeavor. Special thanks also go to Suresh for proofreading this thesis.

I am grateful to my companions in this quest for a Ph.D. It is the camaraderie of my fellow students that has sustained me through many long nights — whether these nights were plagued by an unfinished paper, a never-ending rubber of bridge or that extra half-hour of ice-time. Together we have discovered just why doing a Ph.D. is "90 percent psychology."

I am especially grateful to my wife Christine Piatko, who has been my best friend, confidante and constant companion. I thank her for sharing her life with me and for putting up with this crazy, hectic summer.

I am indebted to my parents for all the trouble they have gone through to put me in school and for instilling in me a proper appreciation for education.

Finally, I acknowledge the generous financial support from the Cornell Sage Graduate Fellowship, the IBM Graduate Fellowship, and NSF Research Grants DCR-85-20597 and CCR-88-23053.

Table of Contents

1 Introduction **1**

2 Groundwork **5**
2.1 The Boolean Hierarchy Defined 6
2.2 Bounded Query Classes Defined 8
2.3 The Hard/Easy Argument 10
2.4 Nonuniformity, Polynomial Advice and Sparse Sets 16
2.5 Summary 18

3 Building Boolean Hierarchies **20**
3.1 Some Building Blocks 22
3.2 Languages Which Do 23
3.3 Characterizations of Complete Languages 26
3.4 Languages Which Don't 28
3.5 AND_2 and OR_2 and Hierarchies 30
3.6 Summary 36

4 Collapsing the Boolean Hierarchy **37**
4.1 If the Boolean Hierarchy Collapses 38
4.2 Extensions 53
4.3 Summary 54

5 Incomplete Sets **56**
5.1 High and Low Sets 56
5.2 Bounded Queries to Incomplete Sets 58
5.3 Proof of Lemma 5.2 64
5.4 Summary 68

6 Unique Satisfiability and Randomized Reductions **70**
6.1 An Historical Account 71
6.2 Anomalous Behavior 75
6.3 Threshold Behavior 76
6.4 Examples of Threshold Behavior 76
6.5 Threshold Behavior in D^P and co-D^P 77
6.6 Merlin-Arthur-Merlin Games 86
6.7 Summary 91

Bibliography **92**

List of Figures

2.1 The Boolean Hierarchy and the Query Hierarchies. 11

3.1 New labelled graphs A and B. 25

5.1 Let A be an incomplete set in NP $-$ $\widehat{\text{low}_3}$. 61

6.1 USAT and related complexity classes. 72

Chapter 1

Introduction

The goal of computational complexity theory is to measure the difficulty of problems using computation as a benchmark. In this paradigm, a problem is considered more difficult if a Turing machine requires more resources to solve the problem. The field has evolved from the basic study of space and time as resources [HS65,SHL65] into a rich discipline covering such diverse resource as randomization, nondeterminism, alternation, parallelism and oracle queries, to name a few. Throughout this development the P =? NP question — whether nondeterminism is required if one wishes to solve some seemingly intractable problems in polynomial time — has remained the central issue in complexity theory. In this thesis, we examine some structural aspects of the complexity of NP computations. We begin by explaining the need for structural complexity theory and devote the remainder of the chapter to an outline of this thesis.

The P =? NP question is of preeminent interest in complexity theory because the NP-complete languages capture the complexity of natural problems such as Boolean Satisfiability (SAT), the Traveling Salesman Problem (TSP) and Graph Colorability. The P =? NP question addresses only one aspect of the complexity of the NP-complete languages — their deterministic time complexity. However, there are many aspects of the complexity of the NP-complete problems which cannot be described in these terms. For example, the following question may arise when one needs to find the optimal traveling salesman tour.

> Given a graph G and a tour T, is T an optimal tour?

This problem is not complete for NP, but for co-NP, the set of languages whose

complements are in NP. The difference between NP and co-NP is quite substantial. For example, given a graph, one can conceivably find in polynomial time (using, say, a stretch of the imagination, a heuristic and some lucky guesses) a coloring for the graph or even a traveling salesman tour. This endeavor is greatly assisted by the fact that given a coloring or a tour, one can easily check that the coloring does not assign adjacent vertices the same color and that the tour actually visits the all the vertices. However, there is no obvious way to verify that the coloring or the tour is *optimal*. Our belief that there is, in fact, no proof of polynomial length — not even using heuristics and lucky guesses — which demonstrates the optimality of a coloring or tour is modeled precisely by the assumption that NP $\neq$ co-NP. If P $=?$ NP is the first question one must ask about NP, then surely NP $=?$ co-NP is the second.

Please note that both NP and co-NP have the same deterministic time complexity, because deterministic time classes are trivially closed under complementation. So, deterministic time does not fully capture the computational complexity of the NP-complete problems. To accomplish this goal, we need to investigate the *structural* complexity of the NP-complete languages.

Many complexity classes have been defined to formalize the NP optimization problems — problems such as finding the shortest traveling salesman tour, finding the lexically largest satisfying assignment and computing a coloring of a graph using the least number of colors [BJY91]. For example, let $\mathrm{PF}^{\mathrm{NP}[f(n)]}$ denote the class of functions computed by polynomial time Turing machines which ask at most $f(n)$ queries to an NP oracle on inputs of length n. Krentel [Kre88] showed that the function classes $\mathrm{PF}^{\mathrm{NP}[\log n]}$ and $\mathrm{PF}^{\mathrm{NP}[n]}$ correspond to the NP optimization problems over a solution space of polynomial and exponential size, respectively. Moreover, he showed that if $\mathrm{PF}^{\mathrm{NP}[\log n]} = \mathrm{PF}^{\mathrm{NP}[n]}$, then P = NP. For the analogous classes of languages, Kadin [Kad88] showed that for all constants k, if $\mathrm{P}^{\mathrm{NP}[k]} = \mathrm{P}^{\mathrm{NP}[k+1]}$ then NP/*poly* = co-NP/*poly*. (NP/*poly* and co-NP/*poly* are the nonuniform analogues of NP and co-NP.) In these formalizations, each query corresponds roughly to one step in the optimization process. These theorems show that queries to NP oracles cannot be reduced unless deciding membership in NP languages is much easier than we believe. That is, each step in the optimization process is as hard as recognizing an NP-complete language.

Please note again that, aside from polynomial factors, the classes $\mathrm{PF}^{\mathrm{NP}[1]}$,

$\mathrm{PF}^{\mathrm{NP}[k]}$, $\mathrm{PF}^{\mathrm{NP}[\log n]}$ and $\mathrm{PF}^{\mathrm{NP}[n]}$ all lie in the same deterministic time class. However, the theorems of Krentel show that the gaps between these classes are at least as wide as the gap between P and NP. Similarly, the results of Kadin show that the gaps between the bounded query language classes are as large as that between NP/*poly* and co-NP/*poly*. Again, we conclude that classifying these problems in terms of deterministic time is simply inadequate.

In the proof of his theorem, Kadin did not work with the bounded query classes directly. Instead, he worked with a related hierarchy called the Boolean Hierarchy. The Boolean Hierarchy (BH) is essentially a hierarchy of nested differences of NP languages. To prove his main theorem, Kadin pioneered the use of the hard/easy argument which showed that if the Boolean Hierarchy collapses, then NP/*poly* = co-NP/*poly*. In Chapter 2, we will discuss the definition of the Boolean Hierarchy in full detail. We will also review the hard/easy argument because this proof technique is used throughout this thesis.

In Chapter 3, we will examine the properties of the Boolean Hierarchy in greater depth. We show that the closure of NP under the Boolean operators AND_2 and OR_2 is the essential property that gives the Boolean Hierarchy is particular characteristics. The theorems presented in this chapter allows us to work with Boolean hierarchies built on top of arbitrary complexity classes.

In Chapter 4, we show that if the Boolean Hierarchy collapses to level k, then the Polynomial Hierarchy collapses to $\mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}})$, where $\mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}})$ is the k^{th} level of the Boolean hierarchy over Σ_2^{P}. This is an improvement over previously known results [Kad88,Wag89]. This result is significant in two ways. First, the theorem says that a deeper collapse of the Boolean Hierarchy implies a deeper collapse of the Polynomial Hierarchy. Also, this result points to some previously unexplored connections between the Boolean and query hierarchies of Δ_2^{P} and Δ_3^{P}. Namely,

- $\mathrm{BH}_k = \text{co-}\mathrm{BH}_k \Longrightarrow \mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}}) = \text{co-}\mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}})$.
- $\mathrm{P}^{\mathrm{NP}\|[\mathrm{k}]} = \mathrm{P}^{\mathrm{NP}\|[\mathrm{k}+1]} \Longrightarrow \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}\|[k+1]} = \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}\|[k+2]}$.

So far, the theorems we have mentioned show that if NP/*poly* $\neq$ co-NP/*poly*, then the NP-complete languages have the *additional query property*. That is, each additional query to an NP-complete language provides enough extra computational resource to recognize new languages. Presumably, the NP-complete languages have

the additional query property because they have a high-degree of complexity. In Chapter 5, we show that the additional query property is not an exclusive property of the NP-complete languages. In fact, we show that any language in NP $-$ $\widehat{\text{low}_3}$ has the additional query property.

Finally, in Chapter 6, we present an application of the hard/easy argument which determines the structural complexity of the unique satisfiability language (USAT). USAT is the set of Boolean formulas with exactly one satisfying assignment and was known to be complete for D^{P} under randomized reductions. However, the meaning of this sort of completeness was not known. The theorems in this chapter show that USAT $\notin$ co-D^{P} unless NP/*poly* = co-NP/*poly*. Moreover, USAT has this property *because* it is complete for D^{P} under randomized reductions. Hence, these theorems also provide an explanation of the meaning of completeness under randomized reductions.

Chapter 2

Groundwork

In this chapter, we examine the definition of the Boolean Hierarchy and describe its essential features. We take this opportunity to review the basic properties of the Boolean Hierarchy, the bounded query hierarchies, sparse sets and polynomial advice functions. Also, we look at the hard/easy argument in the simplest setting, so the reader might become familiar with this important proof technique.

We assume that the reader is familiar with Turing machines; the complexity classes NP and co-NP; the polynomial time hierarchy (PH); and the NP-complete language SAT. For the uninitiated reader, any standard textbook on automata theory and complexity theory would serve as an excellent reference [HU79,LP81, BDG88,BDG90].

In addition to the standard definition of the Turing machine, which includes an effective enumeration of polynomial time deterministic and nondeterministic Turing machines, we will assume the existence of an efficient compiler which translates high level descriptions of algorithms into equivalent Turing machine indices[1]. This complier gives us all of the usual trappings of Turing machines, such as universal machines, effective composition and the s-m-n theorem. Also, all of the polynomial time Turing machines we will consider are clocked, so a simple examination of the transition tables will reveal upper bounds on the running times of the machines. Finally, we assume that all running times are at least linear and are monotonically increasing.

[1]Detractors from this point of view are invited to enroll in a course in compiler construction.

2.1 The Boolean Hierarchy Defined

When the Boolean Hierarchy was first defined, it was envisioned as a collection of "hardware circuits" over NP. These circuits are composed of the standard AND, OR, NOT gates and an additional black box which determines whether a given Boolean formula is satisfiable [CH86]. Alternatively, the Boolean Hierarchy has been defined as the Hausdorff closure of NP sets under the logical operators AND, OR, NOT [WW85]. (We will say more about these Boolean connectives in Chapter 3.) The first years of research in the Boolean Hierarchy showed that these definitions are equivalent. In fact, the Boolean Hierarchy was shown to be robust under many other definitions. In the ensuing years, the results obtained about the Boolean Hierarchy have shown that these are, if not the *correct* definitions, then certainly most convenient definitions.

To appreciate the definition of the Boolean Hierarchy, we will start at the bottom level. The first level of the Boolean Hierarchy, denoted BH_1, is simply the class NP. Similarly, co-BH_1 is co-NP. Note that NP and co-NP are closed under effective intersections and unions. However, if we join an NP language and a co-NP language using intersection, then the resulting language may not be in either NP or co-NP. In fact, the class of all languages formed by such an intersection is $\mathrm{D^P}$, a class introduced by Papadimitriou and Yannakakis to study the complexity of facets of polytopes.

Definition:

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

These two classes have complete languages under many-one reductions. Since SAT is the canonical complete language for NP, the canonical complete languages for $\mathrm{D^P}$ and co-$\mathrm{D^P}$ also involve satisfiability. $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ and $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$ are complete under $\leq^{\mathrm{P}}_{\mathrm{m}}$-reductions for $\mathrm{D^P}$ and co-$\mathrm{D^P}$ respectively.

Definition:

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

Clearly, $\mathrm{D^P}$ contains both NP and co-NP. However, $\mathrm{D^P}$ is closed under effective intersections, but not known to be closed under unions. Thus, if we take the union of a $\mathrm{D^P}$ language and an NP language, the resulting language may not be in either $\mathrm{D^P}$ or co-$\mathrm{D^P}$. In this manner, alternating between intersections with co-NP languages and unions with NP languages, we can construct the levels of the Boolean Hierarchy.

Definition: We write BH_k and co-BH_k for the k^{th} levels of the Boolean Hierarchy, defined as follows [CGH$^+$88]:

$$\begin{aligned}
\mathrm{BH}_1 &= \mathrm{NP}, \\
\mathrm{BH}_{2k} &= \{\ L_1 \cap \overline{L_2} \mid L_1 \in \mathrm{BH}_{2k-1} \text{ and } L_2 \in \mathrm{NP}\ \}, \\
\mathrm{BH}_{2k+1} &= \{\ L_1 \cup L_2 \mid L_1 \in \mathrm{BH}_{2k} \text{ and } L_2 \in \mathrm{NP}\ \}, \\
\text{co-}\mathrm{BH}_k &= \{\ L \mid \overline{L} \in \mathrm{BH}_k\ \}.
\end{aligned}$$

The Boolean Hierarchy thus formed is an interlacing hierarchy of complementary languages and has the upward collapse property, much like the more familiar Kleene and Polynomial Hierarchies. That is,

$$\mathrm{BH}_k \cup \text{co-}\mathrm{BH}_k \subseteq \mathrm{BH}_{k+1} \cap \text{co-}\mathrm{BH}_{k+1}$$

and

$$\mathrm{BH}_k = \text{co-}\mathrm{BH}_k \Longrightarrow \forall j \geq 1,\ \mathrm{BH}_k = \mathrm{BH}_{k+j}.$$

Also, the Boolean Hierarchy as a single complexity class is simply the union of BH_k (i.e., $\mathrm{BH} = \bigcup_{k\geq 1} \mathrm{BH}_k$). Finally, we note that the structure of the Boolean Hierarchy can be mimicked to produce the canonical complete languages for each level of the hierarchy [CGH$^+$88].

Definition: We write BL_k for the canonical complete language for BH_k and co-BL_k for the complete language for co-BH_k:

$$\begin{aligned}
\mathrm{BL}_1 &= \mathrm{SAT}, \\
\mathrm{BL}_{2k} &= \{\langle x_1, \ldots, x_{2k}\rangle \mid \langle x_1, \ldots, x_{2k-1}\rangle \in \mathrm{BL}_{2k-1} \text{ and } x_{2k} \in \overline{\mathrm{SAT}}\}, \\
\mathrm{BL}_{2k+1} &= \{\langle x_1, \ldots, x_{2k+1}\rangle \mid \langle x_1, \ldots, x_{2k}\rangle \in \mathrm{BL}_{2k} \text{ or } x_{2k+1} \in \mathrm{SAT}\},
\end{aligned}$$

$$\begin{aligned}
\text{co-BL}_1 &= \overline{\text{SAT}}, \\
\text{co-BL}_{2k} &= \{\langle x_1, \ldots, x_{2k}\rangle \mid \langle x_1, \ldots, x_{2k-1}\rangle \in \text{co-BL}_{2k-1} \text{ or } x_{2k} \in \text{SAT}\}, \\
\text{co-BL}_{2k+1} &= \{\langle x_1, \ldots, x_{2k+1}\rangle \mid \langle x_1, \ldots, x_{2k}\rangle \in \text{co-BL}_{2k} \text{ and } x_{2k+1} \in \overline{\text{SAT}}\}.
\end{aligned}$$

Of course, $\text{BH}_2 = \text{D}^{\text{P}}$ and $\text{BL}_2 = \text{SAT} \wedge \overline{\text{SAT}}$. We will use these terms interchangeably. An alternative way to generalize the class D^{P} is to consider D^{P} as the class of *differences* of NP languages. That is,

$$\text{D}^{\text{P}} \equiv \{\ L_1 - L_2 \mid L_1, L_2 \in \text{NP}\ \}$$

Then, we can generalize D^{P} by continuing with nested differences. The hierarchy constructed this way is called the Difference Hierarchy.

Definition: We write DIFF_k for the k^{th} level of the Difference Hierarchy:

$$\begin{aligned}
\text{DIFF}_1 &= \text{NP}, \\
\text{DIFF}_{k+1} &= \{\ L_1 - L_2 \mid L_1 \in \text{NP} \text{ and } L_2 \in \text{DIFF}_k\ \}
\end{aligned}$$

The language $\text{ODD}_k(\text{SAT})$ is $\leq^{\text{P}}_{\text{m}}$-complete for level k of the Difference Hierarchy. $\text{ODD}_k(\text{SAT})$ consists of all k-tuples of Boolean formulas of which an odd number are satisfiable (see Chapter 3). It turns out that [CGH+88]

$$\forall k,\ \text{BL}_k \equiv^{\text{P}}_{\text{m}} \text{ODD}_k(\text{SAT}).$$

That is, $\text{DIFF}_k = \text{BH}_k$. This equivalence is important because the proofs for some theorems require the structure of BL_k and for others, the structure of $\text{ODD}_k(\text{SAT})$. Since BL_k and $\text{ODD}_k(\text{SAT})$ are both sets of tuples, we will need the following notational devices to specify the portions of a k-tuple.

Notation: We will write π_j for the j^{th} projection function, and $\pi_{(i,j)}$ for the function that selects the i^{th} through j^{th} elements of a k-tuple. For example,

$$\pi_j(x_1, \ldots, x_k) = x_j$$
$$\pi_{(1,j)}(x_1, \ldots, x_k) = \langle x_1, \ldots, x_j\rangle.$$

2.2 Bounded Query Classes Defined

The hierarchy of bounded query classes is inextricably tied to the Boolean Hierarchy. We start by defining what we mean by bounded queries.

Notation: For any language A and function $f(n)$, we write $\mathrm{P}^{A[f(n)]}$ for the set of languages recognized by polynomial time Turing machines that ask at most $f(n)$ queries to the oracle A on inputs of length n. Also, we write $\mathrm{PF}^{A[f(n)]}$ for the corresponding class of functions.

The bounded query classes of interest to us are the classes $\mathrm{P}^{\mathrm{SAT}[k]}$, where k is a constant, and $\mathrm{P}^{\mathrm{SAT}[O(\log n)]}$. At this point, we should make it very clear that we will be working exclusively with bounded query *language* classes. In the literature, there is a related line of research on the bounded query *function* classes. However, these classes are only tangentially related to the Boolean Hierarchy. For example, it is known that $\mathrm{PF}^{\mathrm{SAT}[O(\log n)]} \neq \mathrm{PF}^{\mathrm{SAT}[poly]}$ unless P = NP [Kre88], but no such conditional separation is known for the corresponding language classes. In fact, the $\mathrm{P}^{\mathrm{SAT}[O(\log n)]}$ versus $\mathrm{P}^{\mathrm{SAT}}$ question is perhaps the most obvious and persistent open question in this area.

In general, we would like to know whether queries, as resource bounds, behave like the resource bounds space and time. In particular, when the number of queries is bounded by a constant, we would like to know if each additional query gives Turing machines enough computational power to recognize more languages:

$$\mathrm{P}^{\mathrm{SAT}[1]} \subsetneq \mathrm{P}^{\mathrm{SAT}[2]} \subsetneq \cdots \subsetneq \mathrm{P}^{\mathrm{SAT}[k]} \subsetneq \mathrm{P}^{\mathrm{SAT}[k+1]} \cdots$$

In other words, we ask: is the Query Hierarchy, $\mathrm{QH} = \bigcup_{k \geq 1} \mathrm{P}^{\mathrm{SAT}[k]}$, an infinite hierarchy?

To answer this question, we turn to the parallel query classes. Even though a $\mathrm{P}^{\mathrm{SAT}[k]}$ machine is restricted to asking only k queries, the queries asked by the machine may depend on the answers to previous queries. So, the entire oracle computation tree may have as many as $2^k - 1$ distinct queries. These queries are called *adaptive* or *serial* queries. In the *non-adaptive* or *parallel* query model, the machine is allowed to look at the input string, do some computation and ask the oracle all the queries at once. The oracle then gives the machine the answers to all the queries in one step, say as a bit vector. After this query phase, no more queries are allowed.

Notation: For any language A and function $f(n)$, we write $\mathrm{P}^{A\|[f(n)]}$ for the set of languages recognized by polynomial time Turing machines which ask at most $f(n)$ queries in *parallel* on inputs of length n.

The parallel Query Hierarchy, $\mathrm{QH}_{\parallel} = \bigcup_{k\geq 1} \mathrm{P}^{\mathrm{SAT}\parallel[k]}$, is simply a finer division of the serial Query Hierarchy, because $\mathrm{P}^{\mathrm{SAT}[k]} = \mathrm{P}^{\mathrm{SAT}\parallel[2^k-1]}$ [Bei87]. So, QH is infinite if and only if $\mathrm{QH}_{\parallel}$ is infinite. Moreover, $\mathrm{QH}_{\parallel}$ was shown to be intertwined with the Boolean Hierarchy [Bei87,KSW87]:

$$\forall k \geq 0,\ \mathrm{P}^{\mathrm{SAT}\parallel[k]} \subseteq \mathrm{BH}_{k+1} \subseteq \mathrm{P}^{\mathrm{SAT}\parallel[k+1]}.$$

Thus, all three hierarchies, QH, $\mathrm{QH}_{\parallel}$ and BH, rise or fall together. It has been shown that these hierarchies do not collapse unless the Polynomial Hierarchy collapses [Kad88]. The proof uses the hard/easy argument which we review in the next section. Before we go on, we summarize the basic properties of these hierarchies.

- BL_k is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for BH_k [CGH$^+$88].
- $\mathrm{DIFF}_k = \mathrm{BH}_k$ and $\mathrm{ODD}_k(\mathrm{SAT}) \equiv^{\mathrm{P}}_{\mathrm{m}} \mathrm{BL}_k$ [CGH$^+$88].
- $\mathrm{BH}_k \cup \text{co-}\mathrm{BH}_k \subseteq \mathrm{BH}_{k+1} \cap \text{co-}\mathrm{BH}_{k+1}$ [CGH$^+$88].
- $\mathrm{EVEN}_k(\mathrm{SAT}) \oplus \mathrm{ODD}_k(\mathrm{SAT})$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for $\mathrm{P}^{\mathrm{SAT}\parallel[k]}$ [WW85,Bei87].
- $\mathrm{BH}_k \cup \text{co-}\mathrm{BH}_k \subseteq \mathrm{P}^{\mathrm{SAT}\parallel[k]} \subseteq \mathrm{BH}_{k+1} \cap \text{co-}\mathrm{BH}_{k+1}$ [KSW87,Bei87].
- For all $k \geq 0$, $\mathrm{P}^{\mathrm{SAT}[k]} = \mathrm{P}^{\mathrm{SAT}\parallel[2^k-1]}$ [Bei87].
- $\mathrm{BH}_k = \text{co-}\mathrm{BH}_k \Longrightarrow \forall j \geq 1,\ \mathrm{BH}_k = \mathrm{BH}_{k+j}$ [CGH$^+$88].
- $\mathrm{P}^{\mathrm{SAT}\parallel[k]} = \mathrm{P}^{\mathrm{SAT}\parallel[k+1]} \Longrightarrow \forall j \geq 1,\ \mathrm{P}^{\mathrm{SAT}\parallel[k]} = \mathrm{P}^{\mathrm{SAT}\parallel[k+j]}$.
- $\mathrm{P}^{\mathrm{SAT}[k]} = \mathrm{P}^{\mathrm{SAT}[k+1]} \Longrightarrow \forall j \geq 1,\ \mathrm{P}^{\mathrm{SAT}[k]} = \mathrm{P}^{\mathrm{SAT}[k+j]}$.
- $\mathrm{P}^{\mathrm{SAT}\parallel[n^{O(1)}]} = \mathrm{P}^{\mathrm{SAT}[O(\log n)]}$ [Hem87].

2.3 The Hard/Easy Argument

In this section, we present the hard/easy argument in the simplest setting. The argument is used throughout this thesis, so it would be helpful for the reader to be familiar with the proof technique.

The hard/easy argument was first used by Kadin to show that $\mathrm{D}^{\mathrm{P}} = \text{co-}\mathrm{D}^{\mathrm{P}}$ implies the Polynomial Hierarchy collapses [Kad87,Kad88], although some hints at this technique can be found in recursive function theory [Soa87, p. 58].

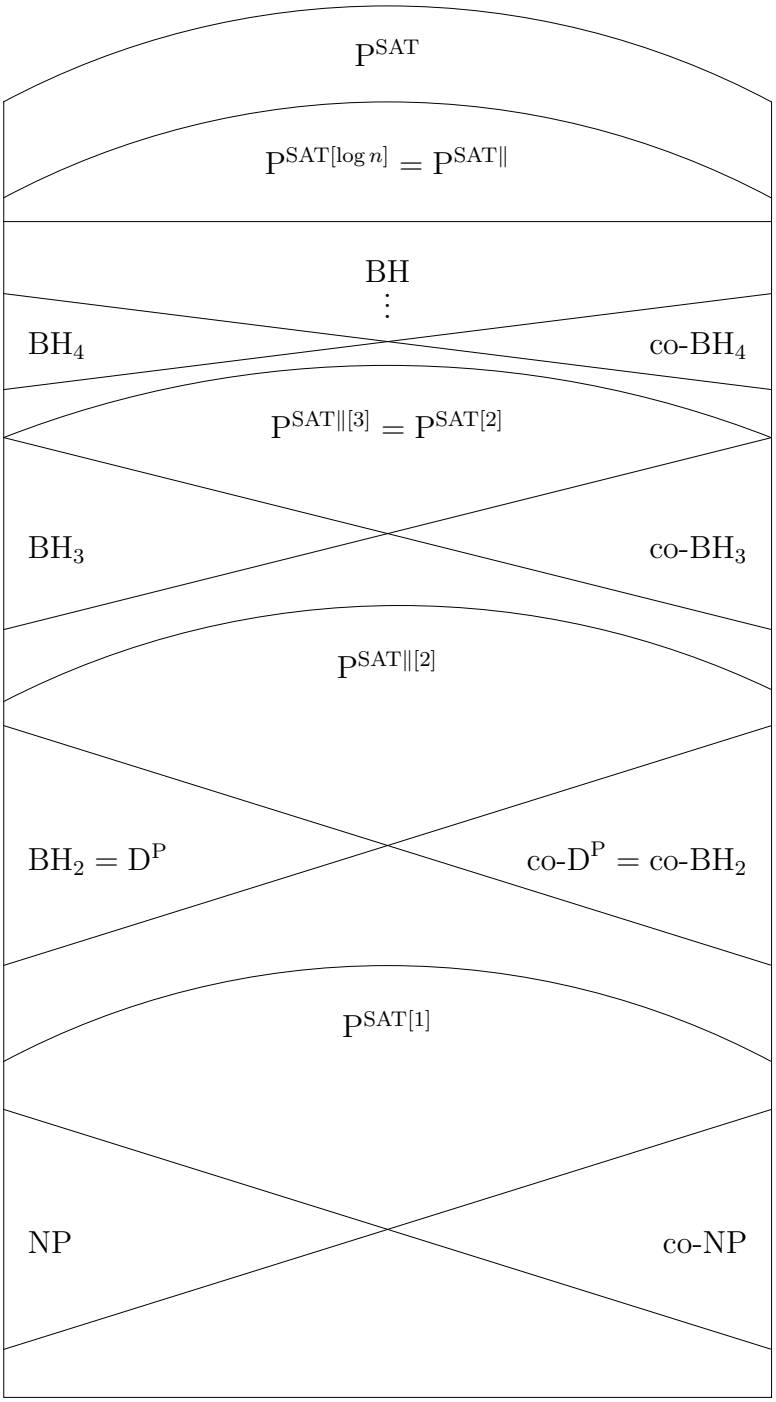

Figure 2.1: The Boolean Hierarchy and the Query Hierarchies.

Theorem 2.1. [Kadin]
If $\mathrm{D^P} = \text{co-}\mathrm{D^P}$ then $\mathrm{NP}/poly = \text{co-NP}/poly$ and $\mathrm{PH} \subseteq \Delta_3^{\mathrm{P}}$.

Proof: Suppose that $\mathrm{D^P} = \text{co-}\mathrm{D^P}$. Then, there must be a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ to $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$, since $\mathrm{SAT} \wedge \overline{\mathrm{SAT}} \in \mathrm{D^P}$ and $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$ is $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for co-$\mathrm{D^P}$. Call this $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction h. Then, for all x,

$$x \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}} \iff h(x) \in \overline{\mathrm{SAT}} \vee \mathrm{SAT}.$$

However, both $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ and $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$ are sets of tuples. So, we can restate this condition as: for all x_1 and x_2,

$$x_1 \in \mathrm{SAT} \text{ and } x_2 \in \overline{\mathrm{SAT}} \iff u_1 \in \overline{\mathrm{SAT}} \text{ or } u_2 \in \mathrm{SAT}, \tag{2.1}$$

where $u_1 = \pi_1(h(x1, x2))$ and $u_2 = \pi_2(h(x1, x2))$.

At this point, Kadin made the critical observation that the structure of the right side of Equation 2.1 is drastically different from the structure of the left side. To satisfy the right side, we only need to determine if either $u_1 \in \overline{\mathrm{SAT}}$ or $u_2 \in \mathrm{SAT}$. To satisfy the left side, we need to check *both* conditions. This observation is critical because a proof that $u_2 \in \mathrm{SAT}$ is sufficient to show that $x_2 \in \overline{\mathrm{SAT}}$. If the unsatisfiability of all formulas can be proven this way, then an NP machine could recognize $\overline{\mathrm{SAT}}$ and NP would equal co-NP. So, let us suspend our disbelief for the moment and assume that unsatisfiability can be checked this easily.

First, let us define our assumption rigorously. We call a string x *easy* if

$$x \in \overline{\mathrm{SAT}} \text{ and } \exists y,\ |y| = |x|, \text{ such that } u_2 \in \mathrm{SAT} \text{ where } u_2 = \pi_2(h(y, x)).$$

If a string x is easy, then there is a polynomial length witness to the unsatisfiability of x. The witness consists of y and the satisfying assignment for u_2. If all the strings in $\overline{\mathrm{SAT}}$ are easy, then the NP machine N_{easy} described below can recognize $\overline{\mathrm{SAT}}$. On input x, N_{easy} does the following:

1. Guess y with $|y| = |x|$.
2. Compute $u_2 = \pi_2(h(y, x))$.
3. Guess an assignment string a.
4. Accept if a is a satisfying assignment of u_2.

We claim that $N_{easy}(x)$ accepts if and only if $x \in \overline{\text{SAT}}$. To see this, observe that if $N_{easy}(x)$ accepts then $u_2 \in \text{SAT}$. Then, by the definition of the reduction h, x must be in $\overline{\text{SAT}}$. On the other hand, if $x \in \overline{\text{SAT}}$, then by assumption x is also easy. So, there must be some y which satisfies the definition of easy. Thus, $N_{easy}(x)$ will find such a y and accept.

What if our outrageous assumption does not hold and some string in $\overline{\text{SAT}}$ is not easy? Let H be such a *hard* string and let $n = |H|$. Then, by definition

$$H \in \overline{\text{SAT}} \text{ and } \forall x,\ |x| = n,\ u_2 \notin \text{SAT} \text{ where } u_2 = \pi_2(h(x, H)).$$

Now, the conditions of equation 2.1 can be simplified. Using only the equation, we know that for all x, $|x| = n$,

$$x \in \text{SAT} \text{ and } H \in \overline{\text{SAT}} \iff u_1 \in \overline{\text{SAT}} \text{ or } u_2 \in \text{SAT},$$

where $u_1 = \pi_1(h(x, H))$ and $u_2 = \pi_2(h(x, H))$. However, we already know that $H \in \overline{\text{SAT}}$ and that $u_2 \notin \overline{\text{SAT}}$, so for all x, $|x| = n$,

$$x \in \text{SAT} \iff u_1 \in \overline{\text{SAT}}. \tag{2.2}$$

By negating both sides of Equation 2.2, we can conclude that

$$x \in \overline{\text{SAT}} \iff u_1 \in \text{SAT}.$$

Now, we can construct an NP machine to recognize unsatisfiable formulas of length n. On input (x, H), N_{hard} does the following

1. If $|x| \neq |H|$, reject.
2. Compute $u_1 = \pi_1(h(x, H))$.
3. Guess an assignment string a.
4. Accept if a is a satisfying assingment of u_1.

Assuming that H is indeed a hard string, we claim that $N_{hard}(x, H)$ accepts if and only if $|x| = |H|$ and $x \in \overline{\text{SAT}}$. Clearly, if $x \notin \text{SAT}$, $(x, H) \notin \text{SAT} \wedge \overline{\text{SAT}}$. So, $u_1 \in \text{SAT}$ by definition of h and $N_{hard}(x, H)$ accepts. On the other hand, if $x \in \text{SAT}$, then $(x, H) \in \text{SAT} \wedge \overline{\text{SAT}}$. So, either $u_1 \in \overline{\text{SAT}}$ or $u_2 \in \text{SAT}$. However, H is a hard string, so $u_2 \notin \text{SAT}$. Thus, $u_1 \in \overline{\text{SAT}}$ and $N_{hard}(x, H)$ rejects.

The construction of N_{easy} and N_{hard} puts us in a win-win situation. If all the strings in $\overline{\text{SAT}}$ of length n are easy, then we can use N_{easy} to recognize $\overline{\text{SAT}}$. On the other hand, if there is a hard string of length n in $\overline{\text{SAT}}$, then we use N_{hard}. There are only two obstacles in our way. First, we need to know which of the two conditions hold for each length. Second, we need to get hold of a hard string if one exists. Unfortunately, checking whether there is a hard string is inherently a two-quantifier ($\exists\forall$) question, so an NP machine cannot answer the question directly—it needs the help of an advice function.

Definition: [KL80,KL82]

Let f be a polynomially bounded function (i.e., $\exists k,\ \forall x,\ |f(x)| \leq |x|^k + k$), and let $\mathcal{C}$ be any class of languages, then a language L is an element of $\mathcal{C}/f$ if there exists a language $A \in \mathcal{C}$ such that

$$x \in L \iff (x, f(1^{|x|})) \in A.$$

Here, f is called the advice function and $f(|x|)$ the advice string. Note that the advice string depends only on the length of x and not on x itself. We say that a language L is in $\mathcal{C}/poly$ if $L \in \mathcal{C}/f$ for some polynomially bounded function f. In general, for any class $\mathcal{F}$ of polynomially bounded functions, $L \in \mathcal{C}/\mathcal{F}$ if $L \in \mathcal{C}/f$ for some $f \in \mathcal{F}$.

Now let f be an advice function which provides a hard string of length n on input 1^n or says that all strings in $\overline{\text{SAT}}$ of length n are easy. (For each length n, there may be several hard strings. In this case, f provides the lexicographically largest one.) From the preceding argument, it is clear that $\overline{\text{SAT}} \in \text{NP}/f$, because an NP machine can run either N_{easy} or N_{hard} depending on the advice. Therefore, co-NP $\subseteq$ NP/*poly* and NP/*poly* = co-NP/*poly*. Then, by a theorem due to Yap [Yap83], PH $\subseteq \Sigma_3^{\text{P}}$. Moreover, f is computable in Δ_3^{P}, so PH $\subseteq \Delta_3^{\text{P}}$. □

The hard/easy argument generalizes to higher levels of the Boolean Hierarchy in a straightforward manner. Since the Boolean Hierarchy is intertwined with QH and $\text{QH}_{\|}$, we have immediate corollaries about the Query Hierarchies as well.

Theorem 2.2. [Kadin]

If $\text{BH}_k = \text{co-BH}_k$ then NP/*poly* = co-NP/*poly* and PH $\subseteq \Delta_3^{\text{P}}$.

Corollary 2.3.
If $\mathrm{P}^{\mathrm{SAT}\|[k]} = \mathrm{P}^{\mathrm{SAT}\|[k+1]}$ then $\mathrm{NP}/\mathit{poly} = \text{co-NP}/\mathit{poly}$ and $\mathrm{PH} \subseteq \Delta_3^{\mathrm{P}}$.

Corollary 2.4.
If $\mathrm{P}^{\mathrm{SAT}[k]} = \mathrm{P}^{\mathrm{SAT}[k+1]}$ then $\mathrm{NP}/\mathit{poly} = \text{co-NP}/\mathit{poly}$ and $\mathrm{PH} \subseteq \Delta_3^{\mathrm{P}}$.

These theorems illustrate two important features of the Boolean Hierarchy. First, recall that in 1987, when these theorems were proven, complexity theorists were busy collapsing hierarchies [Har87a]. Although hierarchies such as the Strong Exponential Hierarchy and the alternating hierarchy over logspace were believed to be infinite, they were shown to collapse using a common census technique [Hem87, Imm88, Kad89, LJK87, SW87, Sze88]. So, there was a strong suspicion that the Boolean Hierarchy would also collapse; that is, until Kadin showed that if the Boolean Hierarchy were to collapse then PH would collapse as well — and nobody knows how to collapse the Polynomial Hierarchy. This theorem lent much respect to the fledgling hierarchy.

Second, these theorems show that the Boolean Hierarchy have a *downward* collapse property with the help of an advice function. That is, if we assume that $\mathrm{BH}_k = \text{co-BH}_k$, then we know that

$$\mathrm{BH}_1/\mathit{poly} = \text{co-BH}_1/\mathit{poly}.$$

In contrast, not much is known about Σ_1^{P} versus Π_1^{P} under the assumption that $\Sigma_2^{\mathrm{P}} = \Pi_2^{\mathrm{P}}$. (Note that $\mathrm{NP} = \mathrm{BH}_1 = \Sigma_1^{\mathrm{P}}$ by definition.) This downward collapse property is a consequence of our ability to use the hard/easy argument to decompose the reduction from BL_k to co-BL_k into a reduction from BL_{k-1} to co-BL_{k-1}. As a result, whether the Boolean Hierarchy is infinite depends only on what happens at the first level — whether NP = co-NP — and on the power of polynomial advice functions.

Technical properties and and historical prespectives aside, what do these theorems really mean? Intuitively, we do not believe that PH collapses or that $\mathrm{NP}/\mathit{poly}$ would be equal to co-$\mathrm{NP}/\mathit{poly}$. Hence, we do not believe that the Boolean Hierarchy collapses. However, many "beliefs" once held with high esteem by complexity theorists have turned out to be false. For example, the relativization principle stated that theorems which have contradictory relativizations cannot be solved by standard techniques like diagonalization and simulation [BGS75]. This principle

has been attacked from several directions [Koz80,Har85,Cha90]. Also, the recent work on the complexity of IP suggests that "non-standard" techniques are not beyond our reach [LFKN90,Sha90]. Another example is the Random Oracle Hypothesis [BG81] which states that theorems which hold for almost all oracle worlds would also hold in the real world. This hypothesis has also been refuted [Kur82, Har85,HCRR90,CGH90].

The relativization principle and the Random Oracle Hypothesis were notorious because of the vagueness of their statements. Many writers have found it difficult to state the conjectures succinctly, even though the conjectures reflect their own opinions. As a result, these conjectures were difficult to support, and even more difficult to refute. In contrast, when we use the working hypothesis that NP/*poly* $\neq$ co-NP/*poly*, our assumptions can be expressed succinctly and exactly. In the next section, we explain why we believe that NP/*poly* would be different from co-NP/*poly* and we examine the consequences should NP/*poly* be equal to co-NP/*poly*.

2.4 Nonuniformity, Polynomial Advice and Sparse Sets

In Chapter 1, we examined the structural differences between NP and co-NP and we mentioned that NP/*poly* and co-NP/*poly* are the nonuniform analogues of NP and co-NP. However, we have not explained what we mean by "nonuniform".

A language L in a nonuniform complexity class is not recognized by a *single* (i.e., uniform) Turing machine. Instead, a sequence of machines $D_1, D_2, D_3, \ldots$ accepts L in the following sense:

$$\forall n,\ \forall x, |x| = n,\ D_n(x) \text{ accepts } \iff x \in L.$$

That is, the n^{th} machine in the sequence is only responsible for recognizing the strings of length n. We make no assumptions about the complexity of generating the sequence $D_1, D_2, D_3, \ldots$, so the sequence may even be uncomputable. However, we do insist that the machines do not grow too large. That is, there must exist a constant k such that $|D_n| \leq n^k + k$.

The obvious question to ask about these nonuniform classes is whether they are more powerful than their uniform counterparts. For example, can nonuni-

form polynomial time Turing machines recognize NP-complete languages? We do not have an absolute answer to this question, but the result of many years of research suggest that nonuniformity does not provide much additional computational power. To explain this statement we need to establish some equivalences between nonuniformity, polynomial advice and sparse sets.

When universal machines exist, the nonuniform complexity classes correspond exactly to the languages recognized by machines with polynomial advice. For example, the languages recognized by nonuniform polynomial time Turing machines is exactly P/*poly*. To see this equivalence, simply note that the polynomial advice provided to a P/*poly* computation on an input of length n is simply the machine D_n. The P/*poly* machine can then simulate D_n using the universal Turing machine. Conversely, a P/*poly* language can be recognized nonuniformly, because the nonuniform machine D_n can have the advice encoded in its transition table without increasing the size of the machine by more than a polynomial. This argument holds for all of the familiar classes like NP, co-NP, Σ_k^{P}, Π_k^{P} and PSPACE. So, instead of asking about the power of nonuniformity, we ask how much information a polynomial advice function can provide.

In a separate development, the complexity of sparse sets was studied.

Definition: For any language L, $L^{\leq n}$ is the set of strings in L of length less than or equal to n. Similarly, we write $L^{=n}$ for the set of strings in L of length n. A set L is *sparse* if there exists a constant k such that $\|L^{\leq n}\| \leq n^k + k$.

The complexity of sparse sets was first studied in the context of the Berman-Hartmanis Conjecture [BH77] which claimed that all languages $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for NP are polynomially isomorphic[2]. If the conjecture holds, then sparse sets cannot be $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for NP. So, when Mahaney showed that sparse sets cannot be NP-hard unless P = NP [Mah82], he lent much support to the belief that there is simply not enough room in a sparse set to encode NP-hard information. Similar theorems have been shown for Turing reductions and truth-table reductions as well [Mah82,Lon82,Har87b,BK88,Kad89,OW90,HL91].

The concepts of sparse sets and polynomial advice are equivalent in the follow-

[2] A and B are polynomially isomorphic if there exists a bijective deterministic polynomial time reduction from A to B whose inverse is also computable in deterministic polynomial time.

ing way:

$$L \in \mathrm{P}/poly \iff \exists S, \text{ a sparse set, s.t. } L \in \mathrm{P}^S$$

Similar relationships hold for NP, co-NP, Σ_k^{P}, Π_k^{P}, PSPACE, etc. We briefly summarize some results concerning NP/*poly*.

- $\overline{\mathrm{SAT}} \in \mathrm{NP} \iff \mathrm{NP}/poly = \text{co-NP}/poly$ [Yap83].
- $\mathrm{NP}/poly = \text{co-NP}/poly \Longrightarrow \mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}$ [Yap83].
- $\Sigma_k^{\mathrm{P}}/poly = \Pi_k^{\mathrm{P}}/poly \Longrightarrow \mathrm{PH} \subseteq \Sigma_{k+2}^{\mathrm{P}}/poly$ [Yap83].
- $\mathrm{BP}\cdot\mathrm{NP} \subseteq \mathrm{NP}/poly$ [Sch89].

The main thrust of this considerable body of research show that polynomial advice (and sparse oracles) provide only a meager amount of information. Moreover, these bits of information are not enough to bridge the structural gaps between determinism and nondeterminism or between nondeterminism and co-nondeterminism. Most of the theorems in this thesis can be interpreted as conditional separations. That is, assuming that there is no polynomially long proof for unsatisfiability — not even in a nonuniform setting — then we can show that certain complexity classes are distinct.

Whether $\mathrm{NP}/poly \neq \text{co-NP}/poly$ is a reasonable assumption is another point to consider. Yap's theorem tells us that if NP/*poly* were equal to co-NP/*poly*, then the Polynomial Hierarchy would collapse. It is the prevailing opinion at this time that PH does not collapse. So, if one subscribes to this view, one would also believe that $\mathrm{NP}/poly \neq \text{co-NP}/poly$. However, if one does not, then we would argue that the connections between the Boolean Hierarchy and the Polynomial Hierarchy are interesting on their own.

2.5 Summary

We have reviewed the basic properties of the Boolean Hierarchy and the serial and parallel Query Hierarchies. We reviewed the hard/easy argument and showed how a collapse at the second level of the Boolean Hierarchy would collapse the Polynomial Hierarchy. We gave some motivation for what collapsing the Polynomial Hierarchy means. The following chapters build on these basic results. We will show that the

hard/easy argument is quite robust and can be modified in many ways to prove interesting properties about the Boolean Hierarchy.

Chapter 3

Building Boolean Hierarchies[1]

In this chapter we consider the existence of polynomial time Boolean combining functions for languages. We say that a language L has a binary AND function, i.e. an AND_2 function, if there is a polynomial time function f such that for all strings x and y, $f(x,y) \in L$ if and only if $x \in L$ and $y \in L$. Similarly, we say that a language L has a binary OR function, an OR_2 function, if there is a polynomial time function g such that for all strings x and y, $g(x,y) \in L$ if and only if $x \in L$ or $y \in L$. In addition, a language may have "any-ary" Boolean functions (AND_ω and OR_ω) — polynomial time functions f and g such that for all n and strings $x_1, \cdots, x_n$, $f(x_1, \cdots, x_n) \in L$ if and only if $x_1, \cdots, x_n$ are all in L, and $g(x_1, \cdots, x_n) \in L$ if and only if at least one of $x_1, \cdots, x_n$ is in L.

The existence of these functions is intimately tied to questions about polynomial time reducibilities and structural properties of languages and complexity classes. Also, we will need these characterizations in the succeeding chapters. Our initial motivation for considering these functions was the observation that all NP-complete languages have AND_ω and OR_ω functions. In fact any language that is $\leq_{\text{m}}^{\text{P}}$-complete for even relativized versions of complexity classes such as P, NP, PSPACE have any-ary Boolean functions by virtue of the fact that such robust complexity classes are represented by machine models that can be run on the different strings in the input tuple. We will show that languages that are $\leq_{\text{m}}^{\text{P}}$-complete for D^{P} have AND_ω but do not have OR_2 unless the Polynomial Hierarchy collapses. Complete languages for the higher levels of the Boolean Hierarchy do not have either AND_2

[1]The results presented in this chapter were discovered jointly with J. Kadin and have appeared in [CK90b].

or OR_2 unless the Polynomial Hierarchy collapses.

These Boolean functions are related to polynomial time conjunctive and disjunctive reducibilities (defined by Ladner, Lynch, and Selman [LLS75]) and to closure of complexity classes under union and intersection. Let m-1(L) be the class of languages $\leq^{\mathrm{P}}_{\mathrm{m}}$- reducible to a language L. It is easy to show that:

$$\begin{aligned} L \text{ has } \mathrm{AND}_2 \quad &\Longleftrightarrow \quad \text{m-1}(L) \text{ is closed under intersection} \\ &\Longleftrightarrow \quad \text{m-1}(L) \text{ is closed under } \leq^{\mathrm{P}}_{2-\mathrm{c}}, \end{aligned}$$

$$L \text{ has } \mathrm{AND}_\omega \quad \Longleftrightarrow \quad \text{m-1}(L) \text{ is closed under } \leq^{\mathrm{P}}_{\mathrm{c}}$$

(the conjunctive reductions, $\leq^{\mathrm{P}}_{2\text{-c}}$ and $\leq^{\mathrm{P}}_{\mathrm{c}}$, are defined in [LLS75]). Similarly, OR is related to disjunctive reducibilities and union. Hence by looking at these concepts in terms of Boolean functions for languages, we are simply thinking of them more as structural properties of languages than as structural properties of complexity classes. An advantage of this approach is that it becomes convenient to study interesting languages such as Graph Isomorphism and USAT (the set of Boolean formulas that have exactly one satisfying assignment) that are not known to be $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for any standard classes.

This chapter is organized in the following manner. Section 3.1 presents definitions and preliminary concepts. In Section 3.2 we discuss some languages that have AND and OR functions. Most notably, we show that Graph Isomorphism does have any-ary AND and OR functions even though it is not known to be NP-complete. In Section 3.3 we characterize the $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete languages of D^{P} and $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ in terms of AND and OR functions. In Section 3.4 we use the above characterizations to show that the complete languages of the different levels of the Boolean Hierarchy and the Query Hierarchies do not have AND and OR functions unless the Boolean Hierarchy collapses (which in turn implies that the Polynomial Hierarchy collapses).

Finally, in Section 3.5, we observe that the existence of AND and OR functions for languages is a condition that makes many proof techniques work. For instance the mind-change technique used by Beigel to show that $\mathrm{P}^{\mathrm{SAT}\|[2^k-1]} = \mathrm{P}^{\mathrm{SAT}[k]}$ [Bei87] works for any set A that has binary AND and OR functions. Similarly, most of the theorems concerning the basic structure and normal forms of the Boolean Hierarchy and the intertwining of the Boolean and Query Hierarchies depend only

on the fact that SAT has AND_2 and OR_2. The results of this section have been proven independently by Bertoni, Bruschi, Joseph, Sitharam, and Young [BBJ$^+$89].

3.1 Some Building Blocks

Although our intuitive notion of AND and OR consists of polynomial time functions that operate on strings, it is convenient to define AND_2, AND_ω, OR_2, OR_ω as sets:

Definition: For any set A, we define the sets

$$\begin{aligned}
\mathrm{AND}_2(A) &= \{\ \langle x, y\rangle \mid x \in A \text{ and } y \in A\ \} \\
\mathrm{OR}_2(A) &= \{\ \langle x, y\rangle \mid x \in A \text{ or } y \in A\ \} \\
\mathrm{AND}_k(A) &= \{\ \langle x_1, \ldots, x_k\rangle \mid \forall i,\ 1 \le i \le k, x_i \in A\ \} \\
\mathrm{OR}_k(A) &= \{\ \langle x_1, \ldots, x_k\rangle \mid \exists i,\ 1 \le i \le k, x_i \in A\ \} \\
\mathrm{AND}_\omega(A) &= \bigcup_{i=1}^{\infty} \mathrm{AND}_i(A) \\
\mathrm{OR}_\omega(A) &= \bigcup_{i=1}^{\infty} \mathrm{OR}_i(A).
\end{aligned}$$

If $\mathrm{AND}_2(A) \le^{\mathrm{P}}_{\mathrm{m}} A$ or $\mathrm{AND}_\omega(A) \le^{\mathrm{P}}_{\mathrm{m}} A$, then we say that A *has* AND_2 or AND_ω respectively. If $\mathrm{OR}_2(A) \le^{\mathrm{P}}_{\mathrm{m}} A$ or $\mathrm{OR}_\omega(A) \le^{\mathrm{P}}_{\mathrm{m}} A$, then we say that A *has* OR_2 or OR_ω respectively. Some obvious facts about $\mathrm{AND}_2(A)$ and $\mathrm{OR}_2(A)$ are:

- $\mathrm{AND}_2(A) \le^{\mathrm{P}}_{\mathrm{m}} A \iff \mathrm{OR}_2(\overline{A}) \le^{\mathrm{P}}_{\mathrm{m}} \overline{A}$.
- if $A \equiv^{\mathrm{P}}_{\mathrm{m}} B$ then $\mathrm{AND}_2(A) \le^{\mathrm{P}}_{\mathrm{m}} A \iff \mathrm{AND}_2(B) \le^{\mathrm{P}}_{\mathrm{m}} B$.
- if $A \equiv^{\mathrm{P}}_{\mathrm{m}} B$ then $\mathrm{OR}_2(A) \le^{\mathrm{P}}_{\mathrm{m}} A \iff \mathrm{OR}_2(B) \le^{\mathrm{P}}_{\mathrm{m}} B$.

These facts hold for $\mathrm{AND}_\omega(A)$ and $\mathrm{OR}_\omega(A)$ as well.

Note that if $\mathrm{AND}_2(A) \le^{\mathrm{P}}_{\mathrm{m}} A$, then for all k, $\mathrm{AND}_k(A) \le^{\mathrm{P}}_{\mathrm{m}} A$. It is possible that $\mathrm{AND}_2(A) \le^{\mathrm{P}}_{\mathrm{m}} A$, but $\mathrm{AND}_\omega(A) \not\le^{\mathrm{P}}_{\mathrm{m}} A$. However if $\mathrm{AND}_2(A) \le^{\mathrm{P}}_{\mathrm{m}} A$ by a *linear time* reduction, then $\mathrm{AND}_\omega(A) \le^{\mathrm{P}}_{\mathrm{m}} A$.

Lemma 3.1. If $\mathrm{AND}_2(A) \le^{\mathrm{P}}_{\mathrm{m}} A$ by a linear time reduction, then $\mathrm{AND}_\omega(A) \le^{\mathrm{P}}_{\mathrm{m}} A$. Similarly, if $\mathrm{OR}_2(A) \le^{\mathrm{P}}_{\mathrm{m}} A$ by a linear time reduction, then $\mathrm{OR}_\omega(A) \le^{\mathrm{P}}_{\mathrm{m}} A$.

Proof: Let f be a linear time reduction from $\mathrm{AND}_2(A)$ to A. For any r, we can take a tuple $\langle x_1, \ldots, x_r\rangle$ and apply f pairwise to $f(x_1, x_2)\ f(x_3, x_4) \cdots f(x_{r-1}, x_r)$. Then

we can apply f pairwise to the outputs of the first applications of f. Repeating this process until we have a single string gives us a tree of applications of f. The height of the tree is $\log r$. If n is the total length of the tuple, $r \leq n$, and so the total running time is bounded by $c^{\log n} n$ for some constant c. This is polynomial in n. □

3.2 Languages Which Do

In this section, we present some familiar languages which are known to have AND_ω and OR_ω.

Lemma 3.2.

1. SAT has AND_ω and OR_ω.
2. $\text{SAT} \wedge \overline{\text{SAT}}$ has AND_ω.

Proof:

1. Given n formulas $\langle f_1, \ldots, f_n \rangle$,

$$\langle f_1, \ldots, f_n \rangle \in \text{AND}_\omega(\text{SAT}) \iff f_1 \wedge \ldots \wedge f_n \in \text{SAT}$$
$$\langle f_1, \ldots, f_n \rangle \in \text{OR}_\omega(\text{SAT}) \iff f_1 \vee \ldots \vee f_n \in \text{SAT}.$$

2. Given n tuples $\langle (f_1, g_1), \ldots, (f_n, g_n) \rangle$,

$$\langle (f_1, g_1), \ldots, (f_n, g_n) \rangle \in \text{AND}_\omega(\text{SAT} \wedge \overline{\text{SAT}})$$
$$\iff (f_1 \wedge \ldots \wedge f_n, g_1 \vee \ldots \vee g_n) \in \text{SAT} \wedge \overline{\text{SAT}}.$$

□

Lemma 3.2 also implies that all NP-complete languages have AND_ω and OR_ω. In fact any language that is $\leq_{\text{m}}^{\text{P}}$-complete for any relativized version of NP, P, or PSPACE has AND_ω and OR_ω. In addition, all languages in P also have AND_ω and OR_ω. One question we ask is whether any of the incomplete languages in $\text{NP} - \text{P}$ have these Boolean functions. In our next theorem, we show that Graph Isomorphism, a natural language that is probably not $\leq_{\text{m}}^{\text{P}}$-complete for NP [GS86, GMW86,BHZ87,Sch88], does have AND_ω and OR_ω. At this time, Primes is the

only other candidate for a natural incomplete language in NP − P, and whether Primes has AND_2 or OR_2 remains an open question.

Definition: $\text{GI} \stackrel{\text{def}}{=} \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$.

The Labelled Graph Isomorphism problem is the problem of recognizing whether two graphs with labelled vertices are isomorphic by an isomorphism that preserves the labels.

Definition:

$$\text{LGI} \stackrel{\text{def}}{=} \{\ \langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs with labelled nodes, and the isomorphism preserves labels }\}.$$

We will show that LGI has AND_ω and OR_ω functions. The existence of AND_ω and OR_ω functions for GI follows from the fact that $\text{LGI} \equiv_{\text{m}}^{\text{P}} \text{GI}$ [Hof79].

Lemma 3.3. LGI has AND_ω and OR_ω.

Proof: Without loss of generality we can assume that graphs are represented as adjacency matrices paired with a table mapping vertices to integer labels.

We define an AND_ω function for LGI as follows. Given r pairs of graphs $\langle (G_1, H_1), \ldots, (G_r, H_r)\rangle$, we preprocess each G_i and H_i by:

1. for each G_i and H_i, add a new vertex and make it adjacent to every old vertex of the original graph.

2. define r new labels. For each i, label the new vertex of G_i and H_i with new label i.

Then let G be the disjoint union of all the altered G_i's (i.e. put them all together in one graph with no extra edges), and let H be the disjoint union of all the altered H_i's.

If for all i, the original G_i is isomorphic to the original H_i by a label preserving mapping, then G is isomorphic to H by mapping each G_i to H_i and the new vertex of G_i to the new vertex of H_i. Clearly, this is an isomorphism from G to H that preserves the labelling. If G is isomorphic (label preserving) to H, then the isomorphism must map the unique vertex in G with new label i (the new vertex

added to G_i) to the unique vertex in H with new label i (the new vertex added to H_i). This induces an isomorphism between G_i and H_i (for all i).

To show that LGI has OR_ω, we first show that LGI has OR_2. Then, we note that the reduction can be done in linear time, which implies that LGI has OR_ω.

Given two pairs of labelled graphs, (G_1, H_1) and (G_2, H_2), we preprocess the graphs as described above (adding 2 new labels). Define a new labelled graph A containing all four graphs G_1, H_1, G_2, and H_2 as subgraphs with 2 new edges added connecting the new vertices of G_1 and G_2 and the new vertices of H_1 and H_2. A new graph B is constructed similarly except the new edges connect G_1 with H_2 and H_1 with G_2 (see Figure 3.1).

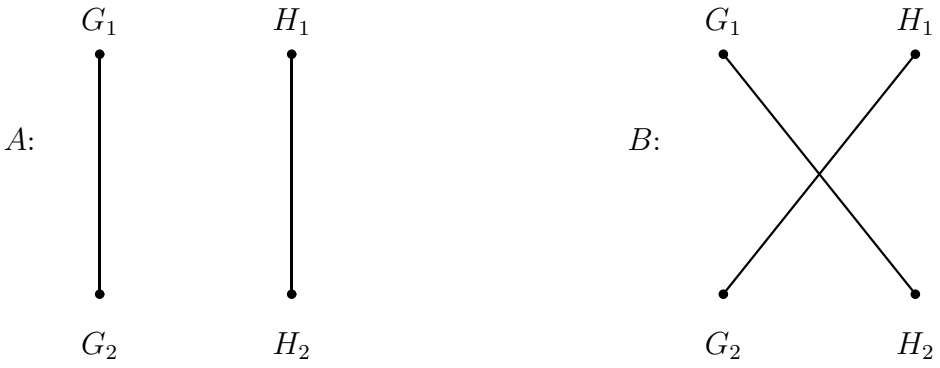

Figure 3.1: New labelled graphs A and B.

Suppose G_1 and H_1 are isomorphic by a label preserving mapping. Then, A and B are isomorphic by mapping G_1 in A to H_1 in B, H_1 to G_1, G_2 to G_2 and H_2 to H_2. Symmetrically, if G_2 and H_2 are isomorphic, then A and B are also isomorphic.

If A and B are isomorphic and G_1 is not isomorphic to H_1, then the new vertex of G_1 in A must be mapped to the new vertex of G_1 in B. Thus, the new vertex of G_2 in A must be mapped to the new vertex of H_2 in B. This induces an isomorphism between G_2 and H_2.

To see that the reduction from OR_2(LGI) to LGI is linear time, note that we only doubled the size of the input and added only 2 new labels. Lemma 3.1 then implies that LGI has OR_ω. □

Corollary 3.4. GI has AND_ω and OR_ω.

3.3 Characterizations of Complete Languages

In this section, we show that the complete languages for D^{P} and $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ can be characterized using AND_2 and OR_ω. These two characterizations are very similar. The only difference is that $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ complete sets have OR_ω and D^{P} complete sets do not. Since $\mathrm{D}^{\mathrm{P}} \neq \mathrm{P}^{\mathrm{NP}[O(\log n)]}$ unless PH collapses, D^{P} complete sets probably do not have OR_ω.

Lemma 3.5. If $\mathrm{SAT} \leq^{\mathrm{P}}_{\mathrm{m}} A$, $\overline{\mathrm{SAT}} \leq^{\mathrm{P}}_{\mathrm{m}} A$ and A has AND_2, then A is D^{P} hard under $\leq^{\mathrm{P}}_{\mathrm{m}}$ reductions.

Proof: $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for D^{P}. Clearly, $\mathrm{SAT} \wedge \overline{\mathrm{SAT}} \leq^{\mathrm{P}}_{\mathrm{m}} A$. So, A must be D^{P} hard. □

Theorem 3.6. Any language A is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for D^{P} if and only if

1. $A \in \mathrm{D}^{\mathrm{P}}$.
2. $\mathrm{SAT}, \overline{\mathrm{SAT}} \leq^{\mathrm{P}}_{\mathrm{m}} A$.
3. A has AND_2.

Now, we show that if a D^{P}-hard set has OR_2 or OR_ω, then it is also hard for higher levels of the parallel Query Hierarchy.

Theorem 3.7. Let A be any set that is $\leq^{\mathrm{P}}_{\mathrm{m}}$-hard for D^{P}.

1. If A has OR_2, then for all k, A is $\leq^{\mathrm{P}}_{\mathrm{m}}$-hard for $\mathrm{P}^{\mathrm{SAT}[k]}$.
2. If A has OR_ω, then A is $\leq^{\mathrm{P}}_{\mathrm{m}}$-hard for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$.

Proof: Let C be any set in $\mathrm{P}^{\mathrm{SAT}[k]}$. To determine if $x \in C$, consider the query tree of the $\mathrm{P}^{\mathrm{SAT}[k]}$ computation. (The query tree is a full binary tree where the internal nodes are labelled by the oracle queries. The two subtrees below the node represent the computations that follow oracle reply. One branch assumes the oracle replied "yes", the other "no".) The query tree has height k and 2^k leaves. Only one path in the tree (from root to leaf) is the correct path, and $x \in C$ if and only if this path ends in an accepting configuration.

Now we show that a D^{P} computation can determine if a given path is the correct path. Let $p_1, \ldots, p_i$ be the queries on the path assumed to be answered "yes", and

$q_1, \ldots, q_j$ be the queries assumed to be answered "no". Then, the path is correct if and only if $p_1 \wedge \ldots \wedge p_i \in \mathrm{SAT}$ and $q_1 \vee \ldots \vee q_j \in \overline{\mathrm{SAT}}$. This is exactly a $\mathrm{D^P}$ computation.

Since the query tree is of constant depth, it is possible to generate the entire tree in polynomial time and write down all the paths that end in an accepting configuration. Note that $x \in C$ if and only if one of these accepting paths is the correct path. Let r be the number of accepting paths. We can use r separate $\mathrm{D^P}$ computations to check if any of the accepting paths is the correct path; and since we assume that A is $\mathrm{D^P}$-hard and has OR_2, we can use one computation of A instead of the r $\mathrm{D^P}$ computations.

The proof of the second case is similar. The only difference is that the query tree has logarithmic depth and polynomial size. □

Corollary 3.8. If $\mathrm{SAT} \leq^{\mathrm{P}}_{\mathrm{m}} A$, $\overline{\mathrm{SAT}} \leq^{\mathrm{P}}_{\mathrm{m}} A$ and A has AND_2 and OR_ω, then A is $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ hard under $\leq^{\mathrm{P}}_{\mathrm{m}}$-reductions.

Proof: By Lemma 3.5 A is a $\mathrm{D^P}$-hard set. By the preceding theorem, A is also hard for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. □

Theorem 3.9. A set A is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ if and only if

1. $A \in \mathrm{P}^{\mathrm{NP}[O(\log n)]}$.
2. $\mathrm{SAT}, \overline{\mathrm{SAT}} \leq^{\mathrm{P}}_{\mathrm{m}} A$.
3. A has AND_2.
4. A has OR_ω.

Notice that the only difference in the characterizations of $\mathrm{D^P}$ complete sets and $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ complete sets (Theorems 3.6 and 3.9) is that $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ complete sets have OR_ω. So, we have the following corollary.

Corollary 3.10. If $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ has OR_ω, then $\mathrm{P}^{\mathrm{NP}[O(\log n)]} \subseteq \mathrm{D^P}$.

Since $\mathrm{P}^{\mathrm{NP}[O(\log n)]} \subseteq \mathrm{D^P}$ implies that PH collapses, Corollary 3.10 is evidence that $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ does not have OR_ω. In fact, $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ probably does not have OR_2, either.

Corollary 3.11. If $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ has OR_2, then $\mathrm{P}^{\mathrm{NP}[O(\log n)]} \subseteq \mathrm{D^P}$.

Proof: If SAT$\wedge\overline{\text{SAT}}$ has OR_2, then by Theorem 3.7, SAT$\wedge\overline{\text{SAT}}$ is hard for $\text{P}^{\text{SAT}[2]}$. However, the complement of SAT$\wedge\overline{\text{SAT}}$ is $\overline{\text{SAT}}\vee$SAT which is in $\text{P}^{\text{SAT}[2]}$. So, SAT$\wedge\overline{\text{SAT}} \equiv^{\text{P}}_{\text{m}} \overline{\text{SAT}}\vee$SAT. Since SAT$\wedge\overline{\text{SAT}}$ has AND_ω by Lemma 3.2, we know that $\overline{\text{SAT}}\vee$SAT has OR_ω by DeMorgan's Law. Thus, SAT$\wedge\overline{\text{SAT}}$ must also have OR_ω since the two sets are $\leq^{\text{P}}_{\text{m}}$-equivalent. Then, by Corollary 3.10 $\text{P}^{\text{NP}[O(\log n)]} \subseteq \text{D}^{\text{P}}$. □

Corollary 3.12. If $\text{D}^{\text{P}} = \text{co-D}^{\text{P}}$ then PH collapses and $\text{P}^{\text{NP}[O(\log n)]} \subseteq \text{D}^{\text{P}}$.

Proof: If $\text{D}^{\text{P}} = \text{co-D}^{\text{P}}$, then $\overline{\text{SAT}}\vee$SAT is $\leq^{\text{P}}_{\text{m}}$-complete for D^{P}. So, $\overline{\text{SAT}}\vee$SAT has AND_2 by Theorem 3.6. Then, by DeMorgan's Law, SAT$\wedge\overline{\text{SAT}}$ has OR_2 which implies $\text{P}^{\text{NP}[O(\log n)]} \subseteq \text{D}^{\text{P}}$ by Theorem 3.11. The Polynomial Hierarchy collapses by Theorem 2.1 due to Kadin. □

It was observed by Kadin [Kad89] that the collapse of the Boolean Hierarchy at level 2 ($\text{D}^{\text{P}} = \text{co-D}^{\text{P}}$) immediately implies $\text{P}^{\text{NP}[O(\log n)]} \subseteq \text{D}^{\text{P}}$. However, we cannot push the same theorem through for the collapse of the Boolean Hierarchy at levels 3 or higher. We can explain this phenomenon using AND_2 and OR_2. Observe that in the proof above we relied on the fact that SAT$\wedge\overline{\text{SAT}}$ has AND_2. We will show in the next section that the complete languages for the levels of the Boolean Hierarchy cannot have AND_2 or OR_2, unless PH collapses.

3.4 Languages Which Don't

In this section, we show that the complete languages for the levels of the Boolean Hierarchy and Query Hierarchies probably do not have AND_2 or OR_2. In the following theorems, keep in mind that $\text{BH}_k \subseteq \text{BH} \subseteq \text{P}^{\text{NP}[O(\log n)]}$. Also, recall that if $\text{BH} \subseteq \text{BH}_k$, $\text{QH}_{\|} \subseteq \text{P}^{\text{SAT}\|[k]}$, $\text{QH} \subseteq \text{P}^{\text{SAT}[k]}$, or $\text{P}^{\text{NP}[O(\log n)]} \subseteq \text{BH}_k$, then the Polynomial Hierarchy collapses. Of course, Theorems 3.13 and 3.14 apply to any set $\leq^{\text{P}}_{\text{m}}$-complete for some level of the Boolean Hierarchy.

Theorem 3.13. For $k \geq 2$,

1. BL_k has $\text{OR}_2 \iff \text{BH} \subseteq \text{BH}_k$.
2. BL_k has $\text{OR}_\omega \iff \text{P}^{\text{NP}[O(\log n)]} \subseteq \text{BH}_k$.

Proof:

1. ($\Rightarrow$) For $k \geq 2$, BL_k is D^{P}-hard, so by Theorem 3.7, BL_k has OR_2 implies that it is hard for $\mathrm{P}^{\mathrm{SAT}[k]}$. However, co-$\mathrm{BL}_k \in \mathrm{P}^{\mathrm{SAT}[k]}$, so co-$\mathrm{BL}_k \leq^{\mathrm{P}}_{\mathrm{m}} \mathrm{BL}_k$. Thus, BH_k is closed under complementation and $\mathrm{BH} \subseteq \mathrm{BH}_k$.

   ($\Leftarrow$) If $\mathrm{BH} \subseteq \mathrm{BH}_k$ then $\mathrm{P}^{\mathrm{SAT}\|[2k]} \subseteq \mathrm{BH}_k$, since $\mathrm{P}^{\mathrm{SAT}\|[2k]} \subseteq \mathrm{BH}_{2k+1}$. Clearly, $\mathrm{OR}_2(\mathrm{BL}_k) \in \mathrm{P}^{\mathrm{SAT}\|[2k]}$, so BL_k has OR_2.

2. ($\Rightarrow$) As above, BL_k is D^{P}-hard. By Theorem 3.9, BL_k has OR_ω implies that BL_k is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. So, $\mathrm{P}^{\mathrm{NP}[O(\log n)]} \subseteq \mathrm{BH}_k$.

   ($\Leftarrow$) $\mathrm{OR}_\omega(\mathrm{BL}_k) \in \mathrm{P}^{\mathrm{SAT}\|} = \mathrm{P}^{\mathrm{NP}[O(\log n)]}$. So, $\mathrm{P}^{\mathrm{NP}[O(\log n)]} \subseteq \mathrm{BH}_k$ implies that BL_k has OR_ω. □

Theorem 3.14. For $k \geq 3$,

1. BL_k has $\mathrm{AND}_2 \iff \mathrm{BH} \subseteq \mathrm{BH}_k$.
2. BL_k has $\mathrm{AND}_\omega \iff \mathrm{P}^{\mathrm{NP}[O(\log n)]} \subseteq \mathrm{BH}_k$.

Proof: Under these assumptions, DeMorgan's Law implies that co-BL_k has OR_2 or OR_ω. Then, the proof proceeds as above, because for $k \geq 3$, co-BL_k is D^{P}-hard. □

If we believe that the Polynomial Hierarchy does not collapse, then the preceding theorems tell us that as we go up the Boolean Hierarchy, the complete languages lose the Boolean connective functions. At the first level, $\mathrm{BL}_1 = \mathrm{SAT}$, so BL_1 has both AND_ω and OR_ω. At the second level, BL_2 is D^{P}complete, so BL_2 has AND_ω but does not have OR_2. From the third level up, BL_k has neither AND_2 nor OR_2. Thus, when we walk up the Boolean Hierarchy from level one, the complete languages lose "pieces" of robustness.

The $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete languages for the different levels of the Query Hierarchy and the parallel Query Hierarchy (relative to SAT) also do not have AND_2 or OR_2 unless PH collapses.

Theorem 3.15. For any $k \geq 1$, if C is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for $\mathrm{P}^{\mathrm{SAT}\|[k]}$, then

C has AND_2 or $\mathrm{OR}_2 \Longrightarrow \mathrm{BH} \subseteq \mathrm{P}^{\mathrm{SAT}\|[k]}$.

Proof: First, note that since $\mathrm{P}^{\mathrm{SAT}\|[k]}$ is closed under complementation, $\overline{C} \equiv^{\mathrm{P}}_{\mathrm{m}} C$. Therefore C has AND_2 if and only if C has OR_2. We will show that if C has either Boolean function, then $\mathrm{BH}_{k+1} \subseteq \mathrm{P}^{\mathrm{SAT}\|[k]}$ which implies the collapse of BH (and of PH). If k is odd, then the $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete language for BH_{k+1} is

$$\mathrm{BL}_{k+1} = \{\ \langle x_1, \ldots, x_{k+1}\rangle \mid \langle x_1, \ldots, x_k\rangle \in \mathrm{BL}_k \text{ and } x_{k+1} \in \overline{\mathrm{SAT}}\ \}.$$

Since $\mathrm{BL}_k \in \mathrm{P}^{\mathrm{SAT}\|[k]}$, $\mathrm{BL}_k \leq^{\mathrm{P}}_{\mathrm{m}} C$. Also, $\overline{\mathrm{SAT}} \leq^{\mathrm{P}}_{\mathrm{m}} C$. So, using the AND_2 function for C, we can combine these two reductions and obtain a reduction from BL_{k+1} to C. Hence, $\mathrm{BH}_{k+1} \subseteq \mathrm{P}^{\mathrm{SAT}\|[k]}$.

If k is even, then the $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete language for BH_{k+1} is

$$\mathrm{BL}_{k+1} = \{\ \langle x_1, \ldots, x_{k+1}\rangle \mid \langle x_1, \ldots, x_k\rangle \in \mathrm{BL}_k \text{ or } x_{k+1} \in \mathrm{SAT}\ \}.$$

In this case, we use the OR_2 function to combine the reductions from BL_k to C and from SAT to C into a reduction from BL_{k+1} to C. Again, we conclude that $\mathrm{BH}_{k+1} \subseteq \mathrm{P}^{\mathrm{SAT}\|[k]}$. □

For another example, in Chapter 6 we will show that USAT, the set of uniquely satisfiable formulas, has AND_ω but does not have OR_ω unless PH collapses.

3.5 AND_2 and OR_2 and Hierarchies[2]

In Chapter 2, we discussed the following basic structural properties of the Boolean and Query Hierarchies:

1. *Normal forms for* BH_k: there are many different but equivalent ways to define the levels of the Boolean Hierarchy [CGH+88].

2. *Complete languages*:

   (a) BL_k is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for BH_k.

   (b) $\mathrm{ODD}_k(\mathrm{SAT})$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for BH_k [Bei87,WW85]. ($\mathrm{ODD}_k(\mathrm{SAT})$ is defined below.)

   (c) $\mathrm{EVEN}_k(\mathrm{SAT}) \oplus \mathrm{ODD}_k(\mathrm{SAT})$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for $\mathrm{P}^{\mathrm{SAT}\|[k]}$ [Bei87,WW85].

[2]The results of this section have been proven independently by Bertoni, Bruschi, Joseph, Sitharam, and Young [BBJ+89].

3. *Basic containments and intertwining* [Bei87,KSW87]:

$$\mathrm{BH}_k \cup \text{co-BH}_k \subseteq \mathrm{P}^{\mathrm{SAT}\|[k]} \subseteq \mathrm{BH}_{k+1} \cap \text{co-BH}_{k+1}.$$

4. *Intertwining of the Query Hierarchies* [Bei87]:

$$\mathrm{P}^{\mathrm{SAT}\|[k]} = \mathrm{P}^{\mathrm{SAT}[2^k-1]}.$$

5. *Upward collapse*:

$$\begin{aligned} \mathrm{BH}_k = \text{co-BH}_k &\implies \mathrm{BH} \subseteq \mathrm{BH}_k \\ \mathrm{P}^{\mathrm{SAT}\|[k]} = \mathrm{P}^{\mathrm{SAT}\|[k+1]} &\implies \mathrm{QH}_{\|} \subseteq \mathrm{P}^{\mathrm{SAT}\|[k]} \\ \mathrm{P}^{\mathrm{SAT}[k]} = \mathrm{P}^{\mathrm{SAT}[k+1]} &\implies \mathrm{QH} \subseteq \mathrm{P}^{\mathrm{SAT}[k]}. \end{aligned}$$

Definition: For any language A, $\mathrm{ODD}_k(A)$ and $\mathrm{EVEN}_k(A)$ are sets:

$$\begin{aligned} \mathrm{ODD}_k(A) &\stackrel{\text{def}}{=} \{\ \langle x_1, \ldots, x_k \rangle \mid \|\{x_1, \ldots, x_k\} \cap A\| \text{ is odd } \} \\ \mathrm{EVEN}_k(A) &\stackrel{\text{def}}{=} \{\ \langle x_1, \ldots, x_k \rangle \mid \|\{x_1, \ldots, x_k\} \cap A\| \text{ is even } \}. \end{aligned}$$

In this section, we show that all the above properties follow simply from the fact that SAT (or any NP-complete set) has AND_2 and OR_2. That is, they do not depend on the fact that SAT is NP-complete or even in NP. To talk about these properties, we need to define the Boolean hierarchy relative to A.

Definition:

$$\begin{aligned} \mathrm{BH}_1(A) &= \{\ L \mid L \leq^{\mathrm{P}}_{\mathrm{m}} A\ \}, \\ \mathrm{BH}_{2k}(A) &= \{\ L_1 \cap \overline{L_2} \mid L_1 \in \mathrm{BH}_{2k-1}(A) \text{ and } L_2 \leq^{\mathrm{P}}_{\mathrm{m}} A\ \}, \\ \mathrm{BH}_{2k+1}(A) &= \{\ L_1 \cup L_2 \mid L_1 \in \mathrm{BH}_{2k}(A) \text{ and } L_2 \leq^{\mathrm{P}}_{\mathrm{m}} A\ \}, \\ \text{co-BH}_k(A) &= \{\ L \mid \overline{L} \in \mathrm{BH}_k(A)\ \}, \\ \mathrm{BH}(A) &= \bigcup_{k=1}^{\infty} \mathrm{BH}_k(A). \end{aligned}$$

Definition: The $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete languages for $\mathrm{BH}_k(A)$ and co-$\mathrm{BH}_k(A)$ are:

$$\begin{aligned}
\mathrm{BL}_1(A) &= A, \\
\mathrm{BL}_{2k}(A) &= \{\ \langle \vec{x}_1, x_2 \rangle \mid \vec{x}_1 \in \mathrm{BL}_{2k-1}(A) \text{ and } x_2 \in \overline{A}\ \}, \\
\mathrm{BL}_{2k+1}(A) &= \{\ \langle \vec{x}_1, x_2 \rangle \mid \vec{x}_1 \in \mathrm{BL}_{2k}(A) \text{ or } x_2 \in A\ \}, \\
\\
\text{co-}\mathrm{BL}_1(A) &= \overline{A}, \\
\text{co-}\mathrm{BL}_{2k}(A) &= \{\ \langle \vec{x}_1, x_2 \rangle \mid \vec{x}_1 \in \text{co-}\mathrm{BL}_{2k-1}(A) \text{ or } x_2 \in A\ \}, \\
\text{co-}\mathrm{BL}_{2k+1}(A) &= \{\ \langle \vec{x}_1, x_2 \rangle \mid \vec{x}_1 \in \text{co-}\mathrm{BL}_{2k}(A) \text{ and } x_2 \in \overline{A}\ \}.
\end{aligned}$$

Note that $\mathrm{BL}_2(\mathrm{SAT}) = \mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ and $\mathrm{BL}_3(\mathrm{SAT}) = (\mathrm{SAT} \wedge \overline{\mathrm{SAT}}) \vee \mathrm{SAT}$. In the general case, $\mathrm{BL}_k(A)$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for $\mathrm{BH}_k(A)$ and co-$\mathrm{BL}_k(A)$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for co-$\mathrm{BH}_k(A)$. However, in the general case, the Boolean hierarchy over A may not be intertwined with the parallel query hierarchy over A. We only know that $\mathrm{BH}_k(A) \cup \text{co-}\mathrm{BH}_k(A) \subseteq \mathrm{P}^{A\|[k]}$.

First, we will leave it to the reader to show that if A has AND_2 and OR_2, then the various normal forms for BH_k hold for the different levels of $\mathrm{BH}(A)$ (see [CH86]). For example, a language L is in $\mathrm{BH}_k(A)$ iff there exist languages $L_1, \ldots, L_k \in$ m-1(A) such that

$$L = D(L_1, \ldots, L_k) \stackrel{\text{def}}{=} L_1 - (L_2 - (L_3 - (\cdots - L_k))).$$

Theorem 3.16.

If A has AND_2 and OR_2, then $\mathrm{ODD}_k(A)$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for $\mathrm{BH}_k(A)$.

Proof: First, we show that $\mathrm{ODD}_k(A)$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-hard for $\mathrm{BH}_k(A)$. If $L \in \mathrm{BH}_k(A)$, then $L = D(L_1, \ldots, L_k)$ for $L_1, \ldots, L_k \in$ m-1(A). Let $L'_i \stackrel{\text{def}}{=} \bigcap_{j \leq i} L_j$. Then, each $L'_i \in$ m-1(A), since m-1(A) is closed under intersection. Also, $L'_1 \supseteq L'_2 \supseteq \cdots \supseteq L'_k$. One can prove by induction that $D(L_1, \ldots, L_k) = D(L'_1, \ldots, L'_k)$. Clearly then, $x \in L$ iff the number of sets $L'_1, \ldots, L'_k$ that contains x is odd. Since each L'_i is in m-1(A), given a string x, we can reduce x to $\langle q_1, \ldots, q_k \rangle$ such that $x \in L'_i$ iff $q_i \in A$. Then, $x \in L$ iff $\langle q_1, \ldots, q_k \rangle \in \mathrm{ODD}_k(A)$. Therefore, $\mathrm{ODD}_k(A)$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-hard for $\mathrm{BH}_k(A)$.

To see that $\mathrm{ODD}_k(A) \in \mathrm{BH}_k(A)$, for each $i \leq k$, consider the set

$$L_i \stackrel{\text{def}}{=} \{\ \langle x_1, \ldots, x_k \rangle \mid \ \| \{x_1, \ldots, x_k\} \cap A \| \ \ \geq i\ \}.$$

$L_i \in$ m-1(A) since given $\langle x_1, \ldots, x_k \rangle$, we can generate all $s = \binom{k}{i}$ subsets of i strings and use the reduction from $\mathrm{OR}_s(\mathrm{AND}_i(A))$ to A to generate a string that is in A iff all i strings in one of the subsets are in A. Then, $\mathrm{ODD}_k(A) = D(L_1, \ldots, L_k)$ and $\mathrm{ODD}_k(A) \in \mathrm{BH}_k(A)$. □

The most difficult proof in this section is the argument that the language $\mathrm{EVEN}_k(A) \oplus \mathrm{ODD}_k(A)$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for $\mathrm{P}^{A\|[k]}$. The argument shows that the mind-change technique [Bei87] works whenever A has AND_2 and OR_2.

Theorem 3.17. If A has AND_2 and OR_2, then $\mathrm{EVEN}_k(A) \oplus \mathrm{ODD}_k(A)$ is complete for $\mathrm{P}^{A\|[k]}$ under $\leq^{\mathrm{P}}_{\mathrm{m}}$-reductions.

Proof: Both $\mathrm{EVEN}_k(A)$ and $\mathrm{ODD}_k(A)$ are obviously in $\mathrm{P}^{A\|[k]}$, so we need to show that every set in $\mathrm{P}^{A\|[k]}$ can be reduced to $\mathrm{EVEN}_k(A) \oplus \mathrm{ODD}_k(A)$.

Let L be a language in $\mathrm{P}^{A\|[k]}$ computed by a polynomial time oracle Turing machine M which asks at most k parallel queries. With each $M^A(x)$ computation, we associate *truth tables* that have the following syntax:

q_1	q_2	$\ldots$	q_k	result
0	0	$\ldots$	0	acc
0	0	$\ldots$	1	rej
$\vdots$	$\vdots$		$\vdots$	$\vdots$
1	1	$\ldots$	1	rej

The idea is that $q_1, \ldots, q_k$ are the strings queried by $M^A(x)$. Each row of the truth table records whether the computation accepts or rejects assuming that the answer to the queries are as listed in the row. (A "1" in column q_i means we are assuming that $q_i \in A$ and a "0", $q_i \notin A$). The full truth table has 2^k lines, but we consider truth tables with fewer lines, as well. In particular, we call a truth table *valid* if

1. The first row of the truth table is all 0's.

2. If a "1" appears in column q_i of row j, then for all rows below row j a "1" appears in column q_i. (Think of each row as the set of q_i's assumed to be in A represented as a bit vector. This condition implies that the rows are monotonic under the set containment relation.)

3. If a "1" appears in column q_i of any row, then q_i is in fact an element of A.

N.B. valid truth tables have between 1 and $k+1$ rows. The following is an example of a valid truth table.

q_1	q_2	q_3	q_4	q_5	q_6	q_7	result
0	0	0	0	0	0	0	rej
0	0	1	0	0	0	0	acc
0	0	1	0	1	0	0	acc
0	0	1	0	1	1	0	rej
0	0	1	0	1	1	0	acc
1	0	1	0	1	1	0	rej

For each valid truth table T, we associate a number m_T—the mind changes of T—which is the number of times the result column changes from accept to reject or from reject to acc. In the example above, m_T is 4. Since valid truth tables have between 1 and $k+1$ rows, $0 \leq m_T \leq k$. Now we define the set of valid truth tables labelled with the number of mind changes.

$$\mathcal{T} = \{\ \langle T, x, s\rangle \mid T \text{ is a valid truth table for } M^A(x) \text{ and } m_T = s\ \}.$$

Claim: $\mathcal{T} \leq_{\mathrm{m}}^{\mathrm{P}} A$. On input $\langle T, x, s\rangle$, a polynomial time machine can simulate $M^A(x)$ to determine which strings, $q_1, \ldots, q_k$, will be queried. Then, the machine can easily check the syntax of T to see if it meets conditions 1 and 2 in the definition of a valid truth table and to see if the number of mind changes is indeed equal to s. If any of these conditions is not met, $\langle T, x, s\rangle$ is reduced to a known string in $\overline{A}$. Otherwise, T is a valid truth table if and only if all the q_i's with a 1 in the last row are actually in A. This last condition can be reduced to A via the reduction from $\mathrm{AND}_k(A)$ to A.

Now define

$$\mathcal{ET}_r = \{\ x \mid \exists T, s \text{ such that } s \geq r \text{ and } \langle T, x, s\rangle \in \mathcal{T}\ \}.$$

Claim: $\mathcal{ET}_r \leq_{\mathrm{m}}^{\mathrm{P}} A$. Since k is constant, for fixed queries $q_1, \ldots, q_k$, there is only a constant number of truth tables with k columns. So simply generate all truth tables T and all values of s between r and k. Since $x \in \mathcal{ET}_r$ if and only if $\langle T, x, s\rangle \in \mathcal{T}$ for one of the (T, s) pairs generated and since $\mathcal{T} \leq_{\mathrm{m}}^{\mathrm{P}} A$, $\mathcal{ET}_r$ can be reduced to A using the OR_2 function for A to combine all the reductions from $\mathcal{T}$ to A into one reduction.

We use the reduction from $\mathcal{ET}_r$ to A to produce a reduction from L to $\mathrm{EVEN}_k(A) \oplus \mathrm{ODD}_k(A)$. This reduction will simply print out a k-tuple $\langle z_1, \ldots, z_k\rangle$

and an extra bit to indicate if the reduction is to $\mathrm{EVEN}_k(A)$ or $\mathrm{ODD}_k(A)$. Each z_i has the property that $z_i \in A$ iff $x \in \mathcal{ET}_i$ iff there exists a valid truth table that makes i mind changes. Let t be the maximum number of mind changes. Then, $z_1, \ldots, z_t \in A$ and $z_{t+1}, \ldots, z_k \notin A$. So, the parity of the number of z_i's in A is the same as the parity of t.

Now, we claim that the parity of the maximum number of mind changes is enough to determine if $M^A(x)$ accepted. First, note that there must exist a valid truth table with the maximum number of mind changes which has, in its last row, the actual oracle replies. (I.e., there is a 1 in column q_i of the last row if and only if $q_i \in A$.) To see this, simply confirm that adding the row of actual oracle replies to the bottom of a valid truth table results in another valid truth table. Thus, $M^A(x)$ accepts iff the result in the last row of this truth table is "accept".

Second, consider this valid truth table that makes t mind changes. Note that if t is odd (even) then the result in the last row is the opposite of (the same as) the result in the first row. Suppose the result in the first row is "accept", then $x \in L$ iff t is even iff $\langle z_1, \ldots, z_k \rangle \in \mathrm{EVEN}_k(A)$. Similarly, if the result in the first row is "reject", then $x \in L$ iff $\langle z_1, \ldots, z_k \rangle \in \mathrm{ODD}_k(A)$. Since the result in the first row can be computed in polynomial time, a polynomial time function can reduce L to $\mathrm{EVEN}_k(A) \oplus \mathrm{ODD}_k(A)$. □

Theorem 3.18. If A has AND_2 and OR_2, then

$$\mathrm{BH}_k(A) \cup \text{co-}\mathrm{BH}_k(A) \subseteq \mathrm{P}^{A\|[k]} \subseteq \mathrm{BH}_{k+1}(A) \cap \text{co-}\mathrm{BH}_{k+1}(A).$$

Proof: The first containment follows from the definitions of the various classes. The second containment holds because $\mathrm{EVEN}_k(A) \oplus \mathrm{ODD}_k(A)$ many-one reduces to $\mathrm{ODD}_{k+1}(A)$. □

Theorem 3.19. If a set A has AND_2 and OR_2, then $\mathrm{P}^{A[k]} = \mathrm{P}^{A\|[2^k-1]}$.

Proof: Since $\mathrm{P}^{A[k]} \subseteq \mathrm{P}^{A\|[2^k-1]}$ for any A, we only need to show $\mathrm{P}^{A\|[2^k-1]} \subseteq \mathrm{P}^{A[k]}$. By Theorem 3.17, it suffices to show that $\mathrm{ODD}_{2^k-1}(A)$, the complete language for $\mathrm{P}^{A\|[2^k-1]}$, is contained in $\mathrm{P}^{A[k]}$.

The main idea of this proof is to use k serial queries to do binary search over $2^k - 1$ strings to determine how many are elements of A. To determine if at least r strings are in A, generate all $s = \binom{2^k-1}{r}$ subsets of the queries with r elements.

Then, use the reduction from $\mathrm{OR}_s(\mathrm{AND}_r(A))$ to A to determine if one of the subsets contains only strings in A. If so, at least r of the query strings are elements of A. Since it takes exactly k steps of binary search to search over $2^k - 1$ elements, $\mathrm{P}^{A[k]} = \mathrm{P}^{A\|[2^k-1]}$. □

With Theorem 3.19 in place, we can prove the upward collapse property for the parallel query hierarchy over A.

Theorem 3.20. If A has AND_2 and OR_2 and $\mathrm{P}^{A\|[k]} = \mathrm{P}^{A\|[k+1]}$, then for all $j > k$, $\mathrm{P}^{A\|[k]} = \mathrm{P}^{A\|[j]}$.

Proof: It suffices to show that under these assumptions, $\mathrm{P}^{A\|[k+1]} = \mathrm{P}^{A\|[k+2]}$. Since A has AND_2 and OR_2, by Theorem 3.17, $\mathrm{EVEN}_{k+1}(A) \oplus \mathrm{ODD}_{k+1}(A)$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for $\mathrm{P}^{A\|[k+1]}$. However, $\mathrm{P}^{A\|[k]} = \mathrm{P}^{A\|[k+1]}$, so there is some $\mathrm{P}^{A\|[k]}$ computation for $\mathrm{EVEN}_{k+1}(A) \oplus \mathrm{ODD}_{k+1}(A)$. Thus, we can construct a $\mathrm{P}^{A\|[k+1]}$ computation to accept $\mathrm{EVEN}_{k+2}(A) \oplus \mathrm{ODD}_{k+2}(A)$. We use k queries to compute the parity of the first $k+1$ strings and one more query to determine if the last string is in A. Since $\mathrm{EVEN}_{k+2}(A) \oplus \mathrm{ODD}_{k+2}(A)$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for $\mathrm{P}^{A\|[k+2]}$, $\mathrm{P}^{A\|[k+1]} = \mathrm{P}^{A\|[k+2]}$. □

The upward collapse properties of the $\mathrm{BH}(A)$ and $\mathrm{QH}(A)$ follow from Theorem 3.20.

3.6 Summary

We have seen how the AND_2 and OR_2 functions interact with the Boolean and Query Hierarchies. These interactions will be useful in the succeeding chapters of this thesis. For example, we have shown that the AND_2 and OR_2 functions can be used to characterize the complete languages for D^{P} and for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. These characterizations are used in Chapter 6 when we determine the structural complexity of the unique satisfiability problem. We have also shown that when we build Boolean hierarchies over languages which have AND_2 and OR_2, all of the basic structural properties of the Boolean Hierarchy still hold. In Chapter 4, we will use this fact when we work with the Boolean hierarchy over $\mathrm{NP}^{\mathrm{NP}}$. Moreover, in Chapter 5, we will examine the difficulties encountered when we work with Boolean hierarchies built over languages not known to have AND_2 and OR_2.

Chapter 4

Collapsing the Boolean Hierarchy

In this chapter, we present a closer analysis of the relationship between the Boolean Hierarchy and the Polynomial Hierarchy. In Section 2.3, we showed that the collapse of the Boolean Hierarchy implies a collapse of the Polynomial Hierarchy. In this chapter, we show how the hard/easy argument can be refined to give a deeper collapse of the Polynomial Hierarchy. The theorems also demonstrate a linkage between the Boolean Hierarchy over NP and the Boolean hierarchy over $\mathrm{NP}^{\mathrm{NP}}$:

$$\mathrm{BH}_k = \text{co-}\mathrm{BH}_k \Longrightarrow \mathrm{PH} \subseteq \mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}}).$$

Recall that in the original hard/easy argument we constructed an NP machine to recognize $\overline{\mathrm{SAT}}$ using an advice function which provided the lexically largest hard string. Then, we argued that since the advice function can be computed in Δ_3^{P}, PH collapses to Δ_3^{P}. In the refinement of the hard/easy argument we do not need all of the information provided by this advice function — a smaller amount of information allows an $\mathrm{NP}^{\mathrm{NP}}$ machine to recognize the complete language for Σ_3^{P}. Moreover, since the smaller amount of information is easier to compute, we can obtain a deeper collapse of PH.

For example, for the case where $\mathrm{BH}_2 = \text{co-}\mathrm{BH}_2$, this information is essentially tally set

$$T \stackrel{\mathrm{def}}{=} \{\ 1^m \mid \exists \text{ a hard string of length } m\ \}.$$

First, we show that $\Sigma_3^{\mathrm{P}} \subseteq \mathrm{NP}^{T \oplus \mathrm{NP}}$. Since the set of hard strings is in co-NP, if we tell an $\mathrm{NP}^{\mathrm{NP}}$ machine that there is a hard string of length m, it can guess a hard string and verify with one query that it is hard. With that hard string, the

$\mathrm{NP}^{\mathrm{NP}}$ machine can produce an NP algorithm that recognizes $\overline{\mathrm{SAT}}^{=m}$. If we tell an $\mathrm{NP}^{\mathrm{NP}}$ machine that there are no hard strings of length m, then it knows that the "easy" NP algorithm recognizes all of $\overline{\mathrm{SAT}}^{=m}$. In either case, the $\mathrm{NP}^{\mathrm{NP}}$ machine can use an NP algorithm for $\overline{\mathrm{SAT}}^{=m}$ to remove one level of oracle queries from a Σ_3^{P} machine, and therefore recognize any Σ_3^{P} language.

Using this refinement, we can demonstrate a collapse of PH down to $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[2]}$. Since an $\mathrm{NP}^{\mathrm{NP}}$ machine can guess and verify hard strings, $T \in \mathrm{NP}^{\mathrm{NP}}$. This implies that a $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ machine can tell with one query if there are any hard strings of a given length. Since this is exactly what an $\mathrm{NP}^{\mathrm{NP}}$ machine needs to recognize a Σ_3^{P} language, the $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ machine can pass the information in one more $\mathrm{NP}^{\mathrm{NP}}$ query and therefore recognize a Σ_3^{P} language with only two queries. Hence $\mathrm{BH}_2 = \text{co-BH}_2$ implies $\Sigma_3^{\mathrm{P}} \subseteq \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[2]}$, the class of languages recognizable in deterministic polynomial time with two queries to $\mathrm{NP}^{\mathrm{NP}}$.

With more work, we can show that Σ_3^{P} is actually contained in the second level of Boolean hierarchy over $\mathrm{NP}^{\mathrm{NP}}$.

4.1 If the Boolean Hierarchy Collapses[1]

We can generalize the analysis presented above to higher levels of the BH by replacing the concept of hard strings with the concept of hard sequences of strings. Just as an individual hard string could be used with the reduction from BH_2 to co-BH_2 to induce a reduction from $\overline{\mathrm{SAT}}$ to SAT, a hard sequence is a j-tuple that can be used with a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from BH_k to co-BH_k to define a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from BH_{k-j} to co-BH_{k-j}.

Before we go on to give the complete definition of a hard sequence, we need some notational devices to handle parts of tuples, reversals of tuples, and sets of tuples. These devices greatly simplify the presentation of the proofs, so we ask the reader to bear with us.

[1] The results presented in this chapter were discovered jointly with J. Kadin and have appeared in [CK90a].

Notation: We will write π_j for the j^{th} projection function, and $\pi_{(i,j)}$ for the function that selects the i^{th} through j^{th} elements of any tuple. For example,

$$\pi_j(x_1, \ldots, x_r) = x_j$$
$$\pi_{(1,j)}(x_1, \ldots, x_r) = \langle x_1, \ldots, x_j \rangle.$$

Notation: Let $\langle x_1, \ldots, x_r \rangle$ be any r-tuple. When the individual strings in the tuple are not significant, we will substitute $\vec{x}$ for $\langle x_1, \ldots, x_r \rangle$. Also, we write $\vec{x}^R$ for $\langle x_r, \ldots, x_1 \rangle$, the reversal of the tuple.

Notation: We abuse the standard notation and use $\{0,1\}^{\leq m \times r}$ to denote the set of r-tuples of strings with length less than or equal to m. That is, $\vec{y} \in \{0,1\}^{\leq m \times r}$, if $\vec{y} = \langle y_1, \ldots, y_r \rangle$ and for $1 \leq i \leq r$, $|y_i| \leq m$.

Now we are ready to give the definition of hard sequences. This definition may seem clumsy at first, but actually sharpens our proofs. We will give some motivation for the definition below.

Definition: Suppose $\mathrm{BL}_k \leq_{\mathrm{m}}^{\mathrm{P}} \text{co-BL}_k$ via some polynomial time function h. Then, we call $\langle 1^m, \vec{x} \rangle = \langle 1^m, x_1, \ldots, x_j \rangle$ a *hard sequence* with respect to h if $j = 0$ or if all of the following hold:

1. $1 \leq j \leq k-1$.
2. $|x_j| \leq m$.
3. $x_j \in \overline{\mathrm{SAT}}$.
4. $\langle 1^m, x_1, \ldots, x_{j-1} \rangle$ is a hard sequence with respect to h.
5. $\forall \vec{y} = \langle y_1, \ldots, y_\ell \rangle \in \{0,1\}^{\leq m \times \ell}$, $\pi_{\ell+1} \circ h(\vec{y}, \vec{x}^R) \in \overline{\mathrm{SAT}}$, where $\ell = k - j$.

If $\langle 1^m, x_1, \ldots, x_j \rangle$ is a hard sequence, then we refer to j as the *order* of the sequence and say that it is a hard sequence for length m. Also, we will call a hard sequence *maximal* if it cannot be extended to a hard sequence of higher order. Finally, we say that j is the maximum order for length m, if there is a hard sequence of order j for length m and there is no hard sequence of order $j+1$ for length m.

To understand the definition of hard sequences one must recall the role of hard strings in Section 2.3. Hard strings, as advice, allow an NP machine to recognize

$\overline{\text{SAT}}$. In Section 2.3, a hard string of length m allows an NP machine to recognize strings in $\overline{\text{SAT}}$ of length *exactly* m. In this chapter, we want a single advice string to help an NP machine to recognize strings in $\overline{\text{SAT}}$ of length *less than or equal to* m. Thus, a hard sequence for length m consists of strings in $\{0,1\}^{\leq m}$. An additional 1^m is placed before the tuple, so it is easy to determine that the sequence is for length m. Moreover, we need the 1^m to enforce a lower bound on the length of a hard sequence. (Otherwise, m may be exponential in the size of the hard sequence.) Conditions 3 and 5 simply mimic the original definition of a hard string (q.v. Section 2.3). Finally, Condition 4 in the definition allows us to prove theorems inductively.

Definition: L_{u_2} and L_{u_3} are the canonical $\leq^{\text{P}}_{\text{m}}$-complete languages for Σ^{P}_2 and Σ^{P}_3 respectively:

$$L_{u_2} \stackrel{\text{def}}{=} \{\ (N_j, x, 1^i) \mid N_j^{\text{SAT}}(x) \text{ accepts in } \leq |x| + i \text{ steps }\},$$

$$L_{u_3} \stackrel{\text{def}}{=} \{\ (N_j, x, 1^i) \mid N_j^{L_{u_2}}(x) \text{ accepts in } \leq |x| + i \text{ steps }\}.$$

Our proof that $\text{BH} \subseteq \text{BH}_k$ implies $\text{PH} \subseteq \text{BH}_k(\text{NP}^{\text{NP}})$ is rather involved. All of our lemmas and theorems start with the assumption that $\text{BH}_k = \text{co-BH}_k$, or in other words, that there exists a function h that is a $\leq^{\text{P}}_{\text{m}}$-reduction from BL_k to co-BL_k. First we show that a hard sequence of order j for length m does indeed induce a reduction from BL_{k-j} to co-BL_{k-j} (Lemma 4.1). Then we show that a maximal hard sequence for length m induces a polynomial time reduction from $\overline{\text{SAT}}^{\leq m}$ to SAT (Lemma 4.2). In Lemma 4.3 we argue that given a maximal hard sequence, an NP machine can recognize an initial segment of L_{u_2}, the canonical complete language for Σ^{P}_2. Lemma 4.4 takes this a step further by showing that given the maximum order of hard sequences for a length, an NP^{NP} machine can recognize an initial segment of L_{u_3}, the canonical complete language for Σ^{P}_3. We then define the set T which encodes the orders of hard sequences for each length, and we show that $T \in \text{NP}^{\text{NP}}$ (Lemma 4.5). We put all this analysis together in Theorem 4.6 and show that $\text{BH}_k = \text{co-BH}_k$ implies $\text{PH} \subseteq \text{P}^{\text{NP}^{\text{NP}[k]}}$.

Moving toward the Δ^{P}_3 Boolean hierarchy, we prove that an NP machine can recognize if there is a hard sequence of order j for length m if it is given a hard sequence for a polynomially longer length (Lemma 4.7). In Lemma 4.8 we show that the maximum order of hard sequences for a length is enough information to

permit an $\mathrm{NP}^{\mathrm{NP}}$ machine to recognize when a string *is not in* L_{u_3}; that is, the $\mathrm{NP}^{\mathrm{NP}}$ machine can recognize an initial segment of a complete language for Π_3^{P}. Finally, this gives us the machinery to prove our main theorem.

We start by showing that a hard sequence of order j for length m induces a reduction from BL_{k-j} to co-BL_{k-j} for tuples of strings up to length m.

Lemma 4.1. Suppose $\mathrm{BL}_k \leq_{\mathrm{m}}^{\mathrm{P}} \text{co-BL}_k$ via some polynomial time function h and $\langle 1^m, \vec{x}\rangle = \langle 1^m, x_1, \ldots, x_j\rangle$ is a hard sequence with respect to h. Then for all $\vec{y} = \langle y_1, \ldots, y_\ell\rangle \in \{0,1\}^{\leq m\times\ell}$ (where $\ell = k - j$)

$$\vec{y} \in \mathrm{BL}_\ell \iff \pi_{(1,\ell)}{}^{\circ}h(\vec{y}, \vec{x}^R) \in \text{co-BL}_\ell.$$

Proof: (by induction on j)

Induction Hypothesis $P(j)$: If $\langle 1^m, \vec{x}\rangle = \langle 1^m, x_1, \ldots, x_j\rangle$ is a hard sequence, then for all $\vec{y} = \langle y_1, \ldots, y_{k-j}\rangle \in \{0,1\}^{\leq m\times(k-j)}$

$$\vec{y} \in \mathrm{BL}_{k-j} \iff \pi_{(1,k-j)}{}^{\circ}h(\vec{y}, \vec{x}^R) \in \text{co-BL}_{k-j}.$$

Base Case $P(0)$: In this case, $\vec{x}$ is the empty sequence, so $\ell = k$. By the hypothesis of the lemma, h reduces BL_k to co-BL_k, so

$$\vec{y} \in \mathrm{BL}_k \iff h(\vec{y}) \in \text{co-BL}_k.$$

However, $\pi_{(1,k)}{}^{\circ}h(\vec{y}) = h(\vec{y})$, so

$$\vec{y} \in \mathrm{BL}_k \iff \pi_{(1,k)}{}^{\circ}h(\vec{y}) \in \text{co-BL}_k.$$

Induction Case $P(j+1)$: Suppose that the induction hypothesis $P(j)$ holds. Let $\ell = k - j$. Let $\langle 1^m, \vec{x}\rangle = \langle 1^m, x_1, \ldots, x_{j+1}\rangle$ be a hard sequence. Then, by definition, $\langle 1^m, \vec{x}\,'\rangle = \langle 1^m, x_1, \ldots, x_j\rangle$ is also a hard sequence. By $P(j)$, for all $\vec{y} = \langle y_1, \ldots, y_{\ell-1}\rangle \in \{0,1\}^{\leq m\times(\ell-1)}$

$$\langle \vec{y}, x_{j+1}\rangle \in \mathrm{BL}_\ell \iff \pi_{(1,\ell)}{}^{\circ}h(\vec{y}, x_{j+1}, \vec{x}\,'^R) \in \text{co-BL}_\ell.$$

If ℓ is even, then by the definitions of BL_ℓ and co-BL_ℓ

$$\begin{aligned} &\vec{y} \in \mathrm{BL}_{\ell-1} \text{ and } x_{j+1} \in \overline{\mathrm{SAT}} \\ &\quad\iff \pi_{(1,\ell-1)}{}^{\circ}h(\vec{y}, \vec{x}^R) \in \text{co-BL}_{\ell-1} \text{ or } \pi_\ell{}^{\circ}h(\vec{y}, \vec{x}^R) \in \mathrm{SAT}. \end{aligned} \tag{4.1}$$

If ℓ is odd, then by the definitions of BL_ℓ and co-BL_ℓ

$$\begin{aligned} &\vec{y} \in \mathrm{BL}_{\ell-1} \ \text{ or } \ x_{j+1} \in \mathrm{SAT} \\ &\quad \iff \pi_{(1,\ell-1)} \circ h(\vec{y}, \vec{x}^R) \in \text{co-BL}_{\ell-1} \text{ and } \pi_\ell \circ h(\vec{y}, \vec{x}^R) \in \overline{\mathrm{SAT}}, \end{aligned} \tag{4.2}$$

or (by negating both sides of the iff),

$$\begin{aligned} &\vec{y} \notin \mathrm{BL}_{\ell-1} \ \text{ and } \ x_{j+1} \in \overline{\mathrm{SAT}} \\ &\quad \iff \pi_{(1,\ell-1)} \circ h(\vec{y}, \vec{x}^R) \notin \text{co-BL}_{\ell-1} \text{ or } \pi_\ell \circ h(\vec{y}, \vec{x}^R) \in \mathrm{SAT}. \end{aligned} \tag{4.3}$$

Since $\langle 1^m, x_1, \ldots, x_{j+1} \rangle$ is a hard sequence, from parts 3 and 5 of the definition of a hard sequence we know that $x_{j+1} \in \overline{\mathrm{SAT}}$ and $\pi_\ell \circ h(\vec{y}, \vec{x}^R) \in \overline{\mathrm{SAT}}$. Therefore in equations (4.1) and (4.3), the second conjunct on the left side is true and the second disjunct on the right side is false. Hence

$$\vec{y} \in \mathrm{BL}_{\ell-1} \iff \pi_{(1,\ell-1)} \circ h(\vec{y}, \vec{x}^R) \in \text{co-BL}_{\ell-1}.$$

Then replacing $k - j$ for ℓ, we have

$$\begin{aligned} &\langle y_1, \ldots, y_{k-(j+1)} \rangle \in \mathrm{BL}_{k-(j+1)} \\ &\quad \iff \pi_{(1,k-(j+1))} \circ h(\langle y_1, \ldots, y_{k-(j+1)}, x_{j+1}, \ldots, x_1 \rangle) \in \text{co-BL}_{k-(j+1)}. \end{aligned}$$

So, we have established the induction hypothesis $P(j+1)$. □

Lemma 4.2 shows that a maximal hard sequence for length m induces a polynomial time reduction from $\overline{\mathrm{SAT}}^{\leq m}$ to SAT, or in other words, given a maximal hard sequence for length m, an NP machine can recognize $\overline{\mathrm{SAT}}^{\leq m}$.

Lemma 4.2. Suppose $\mathrm{BL}_k \leq_{\mathrm{m}}^{\mathrm{P}} \text{co-BL}_k$ via some polynomial time function h and $\langle 1^m, \vec{x} \rangle = \langle 1^m, x_1, \ldots, x_j \rangle$ is a maximal hard sequence with respect to h. Then for all $v \in \{0,1\}^{\leq m}$

$$v \in \overline{\mathrm{SAT}} \iff \exists\, \vec{y} = \langle y_1, \ldots, y_{\ell-1} \rangle \in \{0,1\}^{\leq m \times (\ell-1)},\ \pi_\ell \circ h(\vec{y}, v, \vec{x}^R) \in \mathrm{SAT}$$

(where $\ell = k - j$).

Proof:
If $j = k - 1$ ($\vec{y}$ is the empty sequence), then by Lemma 4.1 for all $v \in \{0,1\}^{\leq m}$

$$v \in \mathrm{BL}_1 \iff \pi_1 \circ h(y, \vec{x}^R) \in \text{co-BL}_1.$$

However, $\mathrm{BL}_1 = \mathrm{SAT}$ and co-$\mathrm{BL}_1 = \overline{\mathrm{SAT}}$. So, we have

$$v \in \mathrm{SAT} \iff \pi_1 \circ h(v, \vec{x}^R) \in \overline{\mathrm{SAT}}.$$

or (by negating both sides of the iff)

$$v \in \overline{\text{SAT}} \iff \pi_1 \circ h(v, \vec{x}^R) \in \text{SAT}.$$

Thus, the lemma holds when $j = k - 1$ (i.e. when $\vec{y}$ is the empty sequence).

Consider the case when $j < k - 1$.
($\Rightarrow$) Suppose $v \in \overline{\text{SAT}}$. Since $\langle 1^m, x_1, \ldots, x_j \rangle$ is maximal, $\langle 1^m, x_1, \ldots, x_j, v \rangle$ is not a hard sequence. However, $j + 1 \leq k - 1$, $|v| \leq m$, $v \in \overline{\text{SAT}}$, and $\langle 1^m, x_1, \ldots, x_j \rangle$ is a hard sequence. So, $\langle 1^m, x_1, \ldots, x_j, v \rangle$ must fail to be a hard sequence by failing to satisfy condition 5 of the definition of hard sequences. Thus,

$$\exists\ \vec{y} = \langle y_1, \ldots, y_{\ell-1} \rangle \in \{0,1\}^{\leq m \times (\ell-1)},\ \pi_\ell \circ h(\vec{y}, v, \vec{x}^R) \in \text{SAT}.$$

($\Leftarrow$) Suppose that for some $\vec{y} = \langle y_1, \ldots, y_{\ell-1} \rangle \in \{0,1\}^{\leq m \times (\ell-1)}$

$$\pi_\ell \circ h(\vec{y}, v, \vec{x}^R) \in \text{SAT}.$$

Since $\langle 1^m, \vec{x} \rangle$ is a hard sequence for length m, by Lemma 4.1

$$\langle \vec{y}, v \rangle \in \text{BL}_\ell \iff \pi_{(1,\ell)} \circ h(\vec{y}, v, \vec{x}^R) \in \text{co-BL}_\ell.$$

If ℓ is even, then by the definitions of BL_ℓ and co-BL_ℓ

$$\begin{aligned} &\vec{y} \in \text{BL}_{\ell-1} \text{ and } v \in \overline{\text{SAT}} \\ &\quad \iff \pi_{(1,\ell-1)} \circ h(\vec{y}, v, \vec{x}^R) \in \text{co-BL}_{\ell-1} \text{ or } \pi_\ell \circ h(\vec{y}, v, \vec{x}^R) \in \text{SAT}. \end{aligned} \tag{4.4}$$

If ℓ is odd, then by the definitions of BL_ℓ and co-BL_ℓ

$$\begin{aligned} &\vec{y} \in \text{BL}_{\ell-1} \text{ or } v \in \text{SAT} \\ &\quad \iff \pi_{(1,\ell-1)} \circ h(\vec{y}, v, \vec{x}^R) \in \text{co-BL}_{\ell-1} \text{ and } \pi_\ell \circ h(\vec{y}, v, \vec{x}^R) \in \overline{\text{SAT}}, \end{aligned} \tag{4.5}$$

or (by negating both sides of the iff)

$$\begin{aligned} &\vec{y} \notin \text{BL}_{\ell-1} \text{ and } v \in \overline{\text{SAT}} \\ &\quad \iff \pi_{(1,\ell-1)} \circ h(\vec{y}, v, \vec{x}^R) \notin \text{co-BL}_{\ell-1} \text{ or } \pi_\ell \circ h(\vec{y}, v, \vec{x}^R) \in \text{SAT}. \end{aligned} \tag{4.6}$$

In either case, we already know by hypothesis that $\pi_\ell \circ h(\vec{y}, v, \vec{x}^R) \in \text{SAT}$, so the right sides of the iff in equations (4.4) and (4.6) are satisfied. Therefore, the left sides of equations (4.4) and (4.6) must also be satisfied, and we have $v \in \overline{\text{SAT}}$.
□

Lemma 4.2 essentially states that a maximal hard sequence produces a way to witness that a formula is unsatisfiable. Hence, given a maximal hard sequence,

an NP machine can guess these witnesses and verify that formulas up to a certain length are unsatisfiable. But if an NP machine can verify that formulas are unsatisfiable, it can simulate an $\mathrm{NP}^{\mathrm{NP}}$ computation by guessing the answer to each NP query and verifying that its guess is correct. We use this idea to prove Lemma 4.3 which states that given a maximal hard sequence, an NP machine can recognize an initial segment of L_{u_2}, the canonical complete language for Σ_2^{P}.

Lemma 4.3. Suppose h is a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from BL_k to co-BL_k. Then, there exists an NP machine N_{σ_2} and a polynomial p_{σ_2} such that if $m \geq p_{\sigma_2}(|w|)$ and $\langle 1^m, \vec{x}\rangle = \langle 1^m, x_1, \ldots, x_j\rangle$ is a maximal hard sequence w.r.t. h, then

$$w \in L_{u_2} \iff N_{\sigma_2}(w, \langle 1^m, \vec{x}\rangle) \text{ accepts.}$$

Proof:

Let $L_{u_2} = L(N_{u_2}^{\mathrm{SAT}})$. Define $p_{\sigma_2}(n)$ to be the upper bound on the running time of N_{u_2} on inputs of length n. Obviously, $N_{u_2}(w)$ queries only strings of length $\leq m$, since $m \geq p_{\sigma_2}(|w|)$. On input $(w, \langle 1^m, \vec{x}\rangle)$, N_{σ_2} does the following:

1. Simulate N_{u_2} step by step until N_{u_2} makes an oracle query to SAT.

2. When N_{u_2} queries "$v \in \mathrm{SAT}$?", branch into two computations. One guesses that $v \in \mathrm{SAT}$, the other guesses that $v \in \overline{\mathrm{SAT}}$.

3. The branch that guesses $v \in \mathrm{SAT}$, will guess a satisfying assignment for v. If none are found, all computations along this branch terminate. If a satisfying assignment is found, then the guess that $v \in \mathrm{SAT}$ is correct and the simulation continues.

4. The branch that guesses $v \in \overline{\mathrm{SAT}}$ will use the maximal hard sequence to find a witness for $v \in \overline{\mathrm{SAT}}$. I.e., it guesses a sequence $\vec{y} = \langle y_1, \ldots, y_{\ell-1}\rangle \in \{0,1\}^{\leq m\times(\ell-1)}$, computes $F = \pi_\ell \circ h(\vec{y}, v, \vec{x}^R)$, and guesses a satisfying assignment for F. If none are found, all computations along this branch terminate. Otherwise, the guess that $v \in \overline{\mathrm{SAT}}$ is correct and the simulation continues.

By Lemma 4.2, if $\langle 1^m, \vec{x} \rangle$ is a maximal hard sequence, then for each query v, $v \in \overline{\text{SAT}}$ if and only if some computation in step 4 finds a satisfiable F. So, in the simulation of the oracle query, all the computations along one branch will terminate and some computations in the other branch will continue. Thus, the simulation continues if and only if the guesses for the oracle answers (either $v \in \text{SAT}$ or $v \in \overline{\text{SAT}}$) are verified, and hence

$$w \in L(N_{u_2}^{\text{SAT}}) \iff N_{\sigma_2}(w, \langle 1^m, \vec{x} \rangle) \text{ accepts.}$$

□

Taking the spirit of Lemma 4.3 one step further, we show that, with the help of a maximal hard sequence, an NP^{NP} machine can simulate a Σ_3^{P} machine. In addition, an NP^{NP} machine can guess and verify hard sequences, so it does not need to be given a maximal hard sequence, all it really needs is the *maximum order*. Therefore there exists an NP^{NP} machine which, given the maximum order of hard sequences for a length, can recognize initial segments of L_{u_3}, the complete language for Σ_3^{P}.

Lemma 4.4. Suppose h is a $\leq_{\text{m}}^{\text{P}}$-reduction from BL_k to co-BL_k. There exists an NP^{SAT} machine N_{σ_3} and a polynomial p_{σ_3} such that for any $m \geq p_{\sigma_3}(|w|)$, if j is the maximum order for length m w.r.t. h, then

$$w \in L_{u_3} \iff N_{\sigma_3}^{\text{SAT}}(w, j, 1^m) \text{ accepts.}$$

Furthermore, if j is greater than the maximum order for length m w.r.t. h,

$$\forall w \;\; N_{\sigma_3}^{\text{SAT}}(w, j, 1^m) \text{ rejects.}$$

Proof:

Let $L_{u_3} = L(N_{u_3}^{L_{u_2}})$, where $L_{u_2} = L(N_{u_2}^{\text{SAT}})$ is the canonical complete language for Σ_2^{P}. Let $r(n)$ be a polynomial upper bound on the running time of N_{u_3} on inputs of length n. Clearly, $N_{u_3}(w)$ will only query strings of length $\leq r(n)$, where $n = |w|$. Apply Lemma 4.3 to obtain N_{σ_2} and the polynomial p_{σ_2}. Let $p_{\sigma_3}(n) \stackrel{\text{def}}{=} p_{\sigma_2}(r(n))$.

The critical observation to make here is that the set of hard sequences is in co-NP. (This is obvious from the definition of hard sequences.) So, given j, the maximum order for length $m \geq p_{\sigma_3}(n)$, an NP^{NP} machine can guess a sequence of j strings $\vec{x} = \langle x_1, \ldots, x_j \rangle \in \{0,1\}^{\leq m \times j}$ and ask the NP oracle if $\langle 1^m, \vec{x} \rangle$ forms

a hard sequence. If $\langle 1^m, \vec{x} \rangle$ does form a hard sequence, then it must also be a maximal sequence since it is of maximum order. Now, $N_{\sigma_3}^{\mathrm{SAT}}(w, j, 1^m)$ can simulate $N_{u_3}^{L_{u_2}}(w)$ step by step, and when N_{u_3} queries "$y \in L_{u_2}$?", N_{σ_3} will ask "$(y, \langle 1^m, \vec{x} \rangle) \in L(N_{\sigma_2})$". By Lemma 4.3, the two queries will return with the same answers, so

$$w \in L(N_{u_3}^{L_{u_2}}) \iff N_{\sigma_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ accepts.}$$

Note that when N_{σ_3} guesses the hard sequence $\langle 1^m, \vec{x} \rangle$, several computation paths of the NP machine may survive, because there may be many hard sequences of maximum order. However, uniqueness is not important here because any maximal hard sequence will work for N_{σ_2}. So, all the computation branches that manage to guess a hard sequence of maximum order will have the same acceptance behavior. Furthermore, if j is greater than the maximum order, then none of the computation paths survive because there are no hard sequences of order j for length m. Thus, in this case, $N_{\sigma_3}(w, j, 1^m)$ will reject. □

We have shown that maximal hard sequences and maximum orders expand the computational power of nondeterministic machines. We define the set T to be the set of strings encoding the orders of hard sequences for each length.

Definition: Suppose h is a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from BL_k to co-BL_k. We define an associated set T by

$$T \stackrel{\mathrm{def}}{=} \{\ (1^m, j) \mid \exists x_1, \ldots, x_j \in \{0,1\}^{\leq m}, \\ \text{s.t. } \langle 1^m, x_1, \ldots, x_j \rangle \text{ is a hard sequence. } \}$$

Note that since the set of hard sequences is in co-NP, T itself is in $\mathrm{NP}^{\mathrm{NP}}$. This gives us the following lemma.

Lemma 4.5. Suppose h is a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from BL_k to co-BL_k, then the set T defined above is in $\mathrm{NP}^{\mathrm{NP}}$.

Since $T \in \mathrm{NP}^{\mathrm{NP}}$, a $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ machine can compute the order of the maximum hard sequence for length m with $k-1$ queries. The $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ machine can then pass this number to the $\mathrm{NP}^{\mathrm{NP}}$ machine $N_{\sigma_3}^{\mathrm{SAT}}$ of Lemma 4.4 to recognize L_{u_3}, the complete language for Σ_3^{P}. Therefore if the BH collapses to its k^{th} level, Lemmas 4.4 and 4.5 imply that the PH collapses to $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}[k]}}$. This collapse of the polynomial time hierarchy implied by the collapse of the BH is lower than previously known.

Theorem 4.6. Suppose h is a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from BL_k to co-BL_k. Then there exists a $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ machine which accepts L_{u_3} with only k queries to the $\mathrm{NP}^{\mathrm{NP}}$oracle. That is, the polynomial hierarchy collapses to $\mathrm{P}^{(\mathrm{NP}^{\mathrm{NP}})[k]}$.

Proof:

By Lemma 4.4, there exists N_{σ_3} and p_{σ_3} such that if j is the maximum order for length m and $m \geq p_{\sigma_3}(|w|)$, then

$$w \in L_{u_3} \iff N_{\sigma_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ accepts.}$$

Using the fact that T is in $\mathrm{NP}^{\mathrm{NP}}$ (Lemma 4.5), a $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ machine can determine if $(1^m, \ell)$ is in T by asking the oracle. Doing this for all values of ℓ between 1 and $k-1$, it can determine the maximum ℓ such that $(1^m, \ell)$ is in T. This maximum ℓ, call it j, is of course the maximum order for length m. Then with one final query, the $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ machine asks if

$$N_{\sigma_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ accepts.}$$

If the oracle answers "yes", the machine accepts. Otherwise, it rejects. □

Note that we could make Theorem 4.6 stronger by using binary search instead of linear search to find the maximum order. However, we will push the collapse even further in Theorem 4.9, so our inquiry will follow a new direction.

The following lemma states that an NP machine can recognize whether there is a hard sequence of order j for length m if it is given a maximal hard sequence for a longer length.

Lemma 4.7. Suppose h is a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from BL_k to co-BL_k. There exists an NP machine N_t and a polynomial p_t such that if $\langle 1^{m_2}, \vec{x} \rangle$ is a maximal hard sequence w.r.t. h and $m_2 \geq p_t(m_1 + k)$, then

$$(1^{m_1}, j_1) \in T \iff N_t((1^{m_1}, j_1), \langle 1^{m_2}, \vec{x} \rangle) \text{ accepts.}$$

Proof:

Use Lemmas 4.3 and 4.5. □

In Lemma 4.4, we showed that, with the help of the maximum order, an $\mathrm{NP}^{\mathrm{NP}}$ machine can recognize a complete language for Σ_3^{P}. In the next lemma we show that with the help of the maximum order, an $\mathrm{NP}^{\mathrm{NP}}$ machine can also recognize a complete language for Π_3^{P} (i.e. recognize when a string is *not* in L_{u_3}).

Lemma 4.8. Suppose h is a $\leq^{\mathrm{P}}_{\mathrm{m}}$-reduction from BL_k to co-BL_k. Let L_{u_3} be the canonical complete language for Σ^{P}_3. There exists an $\mathrm{NP}^{\mathrm{SAT}}$ machine N_{π_3} and a polynomial p_{π_3} such that for any $m \geq p_{\pi_3}(|w|)$, if j is the maximum order for length m w.r.t. h, then

$$w \in \overline{L_{u_3}} \iff N^{\mathrm{SAT}}_{\pi_3}(w, j, 1^m) \text{ accepts.}$$

Furthermore, if j is greater than the maximum order for length m w.r.t. h,

$$\forall w \;\; N^{\mathrm{SAT}}_{\pi_3}(w, j, 1^m) \text{ rejects.}$$

Proof:

Let $L_{u_3} = L(N^{L_{u_2}}_{u_3})$ where L_{u_2} is the canonical complete language for Σ^{P}_2. By Lemma 4.4, there exist N_{σ_3} and p_{σ_3} such that if j_1 is the maximum order for length m_1, and $m_1 \geq p_{\sigma_3}(|w|)$, then

$$w \in L_{u_3} \iff N^{\mathrm{SAT}}_{\sigma_3}(w, j_1, 1^{m_1}) \text{ accepts.}$$

The language accepted by $N^{\mathrm{SAT}}_{\sigma_3}$ is in Σ^{P}_2, so we can reduce it to L_{u_2} via some polynomial time function g. Let $r(n)$ be an upper bound on the running time of g. Using Lemma 4.3, there exist N_{σ_2} and p_{σ_2}, such that if $\langle 1^{m_2}, \vec{y}\rangle$ is a maximal hard sequence and $m_2 \geq p_{\sigma_2}(r(|w| + k + m_1))$, then

$$N^{\mathrm{SAT}}_{\sigma_3}(w, j_1, 1^{m_1}) \text{ accepts} \iff N_{\sigma_2}(g(w, j_1, 1^{m_1}), \langle 1^{m_2}, \vec{y}\rangle) \text{ accepts.}$$

Let N_s be the NP machine that runs the reduction g and then simulates N_{σ_2}, i.e.,

$$N_s(w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\rangle) \text{ accepts.} \iff N_{\sigma_2}(g(w, j_1, 1^{m_1}), \langle 1^{m_2}, \vec{y}\rangle) \text{ accepts.}$$

Let $p_s \stackrel{\mathrm{def}}{=} p_{\sigma_2} \circ r$. Now, if $m_1 \geq p_{\sigma_3}(|w|)$, $m_2 \geq p_s(|w| + k + m_1)$, j_1 is the maximum order for length m_1 and $\langle 1^{m_2}, \vec{y}\rangle$ is a maximal hard sequence, then

$$\begin{aligned} w \in L_{u_3} &\iff N^{\mathrm{SAT}}_{\sigma_3}(w, j_1, 1^{m_1}) \text{ accepts} \\ &\iff N_s(w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\rangle) \text{ accepts.} \end{aligned} \tag{4.7}$$

We are trying to show that there exists a machine $N^{\mathrm{SAT}}_{\pi_3}$ that accepts $(w, j, 1^m)$ if $w \notin L_{u_3}$ when m is big enough in relation to $|w|$ and j is the maximum order of the hard sequences for length m. The $N^{\mathrm{SAT}}_{\pi_3}$ that we have in mind will map

$$(w, j, 1^m) \longmapsto (w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\rangle)$$

and accept iff $N_s(w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\rangle)$ rejects (iff $w \notin L_{u_3}$). $N^{\mathrm{SAT}}_{\pi_3}$ can tell if $N_s(w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\rangle)$ rejects with one query to SAT.

The difficulty in mapping $(w, j, 1^m) \longmapsto (w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\rangle)$ lies in the fact that $N^{\mathrm{SAT}}_{\pi_3}$ is given j, the maximum order of hard sequences for one length m, and it must compute the maximum orders of two other lengths, m_1 and m_2. We will define p_{π_3} so that if $m \geq p_{\pi_3}(|w|)$, then m will be bigger enough than both m_1 and m_2 so that we can apply Lemma 4.7 to compute j_1 and j_2.

Let $p_{\pi_3}(n) \stackrel{\mathrm{def}}{=} p_t(p_s(n + k + p_{\sigma_3}(n)) + k)$, i.e. $p_{\pi_3}(n) = p_t(m_2 + k)$ where $m_2 = p_s(n + k + m_1)$ and $m_1 = p_{\sigma_3}(n)$ (p_t is the polynomial bound from Lemma 4.7). $N^{\mathrm{SAT}}_{\pi_3}(w, j, 1^m)$ will do the following. (We will annotate the program with a description of what $N^{\mathrm{SAT}}_{\pi_3}(w, j, 1^m)$ accomplishes when j is the maximum order.)

1. Reject if $m < p_{\pi_3}(|w|)$.

2. Guess j strings $x_1, \ldots, x_j \in \{0,1\}^{\leq m}$ and confirm that $\langle 1^m, x_1, \ldots, x_j\rangle$ is a hard sequence by asking the SAT oracle. (Recall that checking if a given tuple forms a hard sequence is a co-NP question.) If j is the maximum order and $\langle 1^m, \vec{x}\,\rangle$ is a hard sequence, then $\langle 1^m, \vec{x}\,\rangle$ is a maximal hard sequence, too.

3. Let $n = |w|$. Compute $m_1 = p_{\sigma_3}(n)$ and $m_2 = p_s(n + k + m_1)$.

4. For $\ell = 0$ to $k - 1$ ask SAT if $N_t((1^{m_1}, \ell), \langle 1^m, \vec{x}\,\rangle)$ accepts. Let j_1 be the maximum ℓ where $N_t((1^{m_1}, \ell), \langle 1^m, \vec{x}\,\rangle)$ does accept. Note that $m = p_t(m_2 + k) \geq p_t(m_1 + k)$ so j_1 is the maximum order for length m_1 (by Lemma 4.7) if j is the maximum order for length m.

5. For $\ell = 0$ to $k - 1$ ask SAT if $N_t((1^{m_2}, \ell), \langle 1^m, \vec{x}\,\rangle)$ accepts. Let j_2 be the maximum ℓ where $N_t((1^{m_2}, \ell), \langle 1^m, \vec{x}\,\rangle)$ does accept. As in step 4, j_2 is the maximum order for length m_2 if j is the maximum order for length m.

6. Guess j_2 strings $y_1, \ldots, y_{j_2} \in \{0,1\}^{\leq m_2}$ and confirm that $\langle 1^{m_2}, y_1, \ldots, y_{j_2}\rangle$ is a hard sequence (using one query to SAT). Note that if $\langle 1^{m_2}, \vec{y}\,\rangle$ is a hard sequence and j_2 is the maximum order, then $\langle 1^{m_2}, \vec{y}\,\rangle$ is also a maximal hard sequence.

7. Ask SAT if $N_s(w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\,\rangle)$ accepts. If SAT returns "no", then $N^{\mathrm{SAT}}_{\pi_3}$ accepts. Note that by the preceding discussion, if j is the maximum order for length m, then j_1 is the maximum order for length m_1, and $\langle 1^{m_2}, \vec{y}\,\rangle$ is a

maximal hard sequence. Also, $m_1 = p_{\sigma_3}(|w|)$ and $m_2 = p_s(|w| + k + m_1)$, so by equation 4.7

$$w \in L_{u_3} \iff N_s(w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y} \rangle) \text{ accepts.}$$

Note that if j is the maximum order for length m and $m \geq p_{\pi_3}(|w|)$, then

$$w \in \overline{L_{u_3}} \iff N_{\pi_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ accepts.}$$

First of all, $N_{\pi_3}^{\mathrm{SAT}}$ accepts in step 7 only. So, if $w \in L_{u_3}$, all computation paths of $N_{\pi_3}^{\mathrm{SAT}}$ reject—even those that reach step 7, because SAT would answer "yes" in step 7. On the other hand, if $w \in \overline{L_{u_3}}$ then some computation path will reach step 7, get "no" from the SAT oracle and accept.

Finally, we note that if j is greater than the maximum order for length m, then no computation path will survive step 2. Thus, in this case $N_{\pi_3}^{\mathrm{SAT}}(w, j, 1^m)$ rejects. □

Now we are ready to prove our main theorem. This theorem demonstrates a close linkage between the collapse of the Boolean Hierarchy and the polynomial time hierarchy.

Theorem 4.9. Suppose h is a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from BL_k to co-BL_k. Let L_{u_3} be the canonical complete language for Σ_3^{P}. Then there exist languages $B_1, \ldots, B_k \in \mathrm{NP}^{\mathrm{NP}}$ such that

$$L_{u_3} = B_1 - (B_2 - (B_3 - (\cdots - B_k))).$$

That is, $\Sigma_3^{\mathrm{P}} \subseteq \mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}})$, and therefore $\mathrm{PH} \subseteq \mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}})$.

Proof:

First, recall that in Lemmas 4.4 and 4.8 it was shown that $N_{\sigma_3}^{\mathrm{SAT}}$ and $N_{\pi_3}^{\mathrm{SAT}}$ accepted L_{u_3} and $\overline{L_{u_3}}$ (respectively) with the help of the maximum order for a large enough length (and they reject if the number given for the maximum order is too large). Let w be any string. Let $m = \max(p_{\sigma_3}(|w|), p_{\pi_3}(|w|))$; then m is large enough so that if j is the maximum order for length m,

$$N_{\sigma_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ accepts} \iff w \in L_{u_3},$$

$$N_{\pi_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ accepts} \iff w \notin L_{u_3}.$$

We will define the $\mathrm{NP}^{\mathrm{NP}}$ languages $B_1, \ldots, B_k$ to be the strings accepted by $\mathrm{NP}^{\mathrm{NP}}$ machines that try to guess j, the maximum order for length m, and then run N_{σ_3} and N_{π_3}. These $\mathrm{NP}^{\mathrm{NP}}$ machines cannot verify when they have guessed the true maximum order, instead they will base their acceptance behavior on whether they can determine that an earlier machine in the sequence may have been fooled by an incorrect guess for j. This successive approximation scheme converges to the language L_{u_3} within k steps.

Definition: For $1 \leq \ell \leq k$ the language B_ℓ is the set of strings w with the property that there exist $j_1, \ldots, j_\ell$ such that

1. $0 \leq j_1 < j_2 < \ldots < j_\ell \leq k-1$.
2. for all odd d, $1 \leq d \leq \ell$, $N_{\sigma_3}^{\mathrm{SAT}}(w, j_d, 1^m)$ accepts.
3. for all even d, $1 \leq d \leq \ell$, $N_{\pi_3}^{\mathrm{SAT}}(w, j_d, 1^m)$ accepts.

Clearly, B_ℓ is in $\mathrm{NP}^{\mathrm{NP}}$, since an $\mathrm{NP}^{\mathrm{NP}}$ machine can guess $j_1, \ldots, j_\ell$, verify the first property, then simulate $N_{\sigma_3}^{\mathrm{SAT}}$ and $N_{\pi_3}^{\mathrm{SAT}}$ for the different values of j_d. Also, observe that the B_ℓ's form a nested sequence

$$B_k \subseteq B_{k-1} \subseteq \ldots \subseteq B_2 \subseteq B_1.$$

Finally, note that if $r = \max\{\ \ell \mid w \in B_\ell\ \}$, then

$$w \in B_1 - (B_2 - (B_3 - (\cdots - B_k))) \iff r \text{ is odd.}$$

Example: Here we give an example which demonstrates that

$$w \in L_{u_3} \iff r = \max\{\ \ell \mid w \in B_\ell\ \} \text{ is odd.}$$

Let $k = 8$, the maximum order for length m be 5, and $N_{\sigma_3}^{\mathrm{SAT}}$ and $N_{\pi_3}^{\mathrm{SAT}}$ behave as shown below for the different values of s plugged in as the guess for the maximum order.

$s =$	0	1	2	3	4	5	6	7
$N_{\sigma_3}^{\mathrm{SAT}}(w, s, 1^m)$	rej	acc	acc	rej	rej	acc	rej	rej
$N_{\pi_3}^{\mathrm{SAT}}(w, s, 1^m)$	acc	rej	acc	rej	acc	rej	rej	rej

Note that since the maximum order is 5, both $N_{\sigma_3}^{\mathrm{SAT}}$ and $N_{\pi_3}^{\mathrm{SAT}}$ reject for $s = 6, 7$. Also, one of $N_{\sigma_3}^{\mathrm{SAT}}(w, 5, 1^m)$ and $N_{\pi_3}^{\mathrm{SAT}}(w, 5, 1^m)$ must accept and the other reject.

For smaller s, both may accept or reject, since their behavior is unpredictable. Finally, $w \in L_{u_3}$, so we want show that r is odd.

To determine if $w \in B_\ell$, look for an alternating (between the top and bottom row) sequence of accepts starting from the top row moving left to right. If there is such a sequence of ℓ accepts, then $w \in B_\ell$. In this example, $r = 3$.

- $w \in B_1$, because j_1 can be 1, 2 or 5.
- $w \in B_2$, because both $N_{\sigma_3}^{\mathrm{SAT}}(w, 1, 1^m)$ and $N_{\pi_3}^{\mathrm{SAT}}(w, 4, 1^m)$ accept.
- $w \in B_3$ with $j_1 = 1, j_2 = 2, j_3 = 5$.
- $w \notin B_4, B_5, \ldots, B_8$, because there is no alternating sequence longer than 3. The sequence $j_1 = 0, j_2 = 1, j_3 = 2, j_4 = 5$ does not count because the sequence must start from the top row.

Claim 1: If $w \in L_{u_3}$, then $r = \max\{\ \ell \mid w \in B_\ell\ \}$ is odd.
Proof: Let j be the maximum order for length m. Now suppose r is even and $w \in B_r$. Then, there exist $j_1, \ldots, j_r$ so that properties 1–3 in the definition above hold. Therefore

$$N_{\pi_3}^{\mathrm{SAT}}(w, j_r, 1^m) \text{ accepts}$$

(since r is even and $w \in B_r$). Since $w \in L_{u_3}$, for the true maximum order j,

$$N_{\pi_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ rejects.}$$

Therefore $j_r \neq j$. Observe that j_r cannot be greater than j either, since for all $s > j$, $N_{\pi_3}^{\mathrm{SAT}}(w, s, 1^m)$ rejects. Hence $j_r < j$.

Since we are given that $w \in L_{u_3}$, we know that $N_{\sigma_3}^{\mathrm{SAT}}(w, j, 1^m)$ must accept, by Lemma 4.4. Now consider the sequence $j_1, \ldots, j_{r+1}$ where $j_{r+1} = j$. $N_{\sigma_3}^{\mathrm{SAT}}$ accepts $(w, j_{r+1}, 1^m)$ and $r+1$ is odd which implies that $j_1, \ldots, j_{r+1}$ satisfies conditions 1–3, so $w \in B_{r+1}$. Thus if r is even, $r \neq \max\{\ \ell \mid w \in B_\ell\ \}$. Therefore, r must be odd.

Claim 2: If $w \notin L_{u_3}$ then $r = \max\{\ \ell \mid w \in B_\ell\ \}$ is even.
Proof: Similar to the proof of Claim 1.

Now, observe that if $r = \max\{\ \ell \mid w \in B_\ell\ \}$, then

$$w \in B_1 - (B_2 - (B_3 - (\cdots - B_k))) \iff r \text{ is odd,}$$

Combining Claims 1 and 2 with this observation, we have

$$w \in L_{u_3} \iff w \in B_1 - (B_2 - (B_3 - (\cdots - B_k))).$$ □

Theorem 4.9 also shows some unexpected connections between the Boolean and query hierarchies within Δ_2^{P} and Δ_3^{P}.

Corollary 4.10. $\mathrm{BH}_k = \text{co-BH}_k \Longrightarrow \mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}}) = \text{co-BH}_k(\mathrm{NP}^{\mathrm{NP}})$.

Corollary 4.11. $\mathrm{P}^{\mathrm{NP}[\mathrm{k}]} = \mathrm{P}^{\mathrm{NP}[\mathrm{k+1}]} \Longrightarrow \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[k+1]} = \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[k+2]}$.

Proof:

We use the fact that the Boolean and query hierarchies are intertwined:

$$\mathrm{P}^{\mathrm{NP}[\mathrm{k}]} \subseteq \mathrm{BH}_{k+1} \subseteq \mathrm{P}^{\mathrm{NP}[\mathrm{k+1}]} \quad \text{and} \quad \mathrm{BH}_{k+1}(\mathrm{NP}^{\mathrm{NP}}) \subseteq \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[k+1]}.$$

If $\mathrm{P}^{\mathrm{NP}[\mathrm{k}]} = \mathrm{P}^{\mathrm{NP}[\mathrm{k+1}]}$, then BH_{k+1} is closed under complementation. Moreover, Corollary 4.10 implies that $\mathrm{BH}_{k+1}(\mathrm{NP}^{\mathrm{NP}})$ is closed under complementation as well. Thus, the Boolean and query hierarchies in Δ_3^{P} collapse to $\mathrm{BH}_{k+1}(\mathrm{NP}^{\mathrm{NP}})$. □

4.2 Extensions

The results in the previous section can be extended in several ways. We give a brief survey of these extensions and encourage the reader to look at the proofs in the primary sources. The main result of Section 4.1 is:

$$\mathrm{BH}_k = \text{co-BH}_k \Longrightarrow \mathrm{PH} \subseteq \mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}}).$$

One obvious question to address is whether it was crucial for k to be a constant. Wagner resolved this question in part by showing the following theorem [Wag88, Wag89].

Theorem 4.12. [Wagner] For any polynomial time computable function $r(n) < n^{1-\epsilon}$ for some constant ϵ with $0 < \epsilon \leq 1$,

$$\mathrm{P}^{\mathrm{SAT}[r(n)]} = \mathrm{P}^{\mathrm{SAT}[r(n)+1]} \Longrightarrow \mathrm{PH} \subseteq \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[O(r(pol))]}.$$

Here, $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[O(r(pol))]}$ is the class of languages recognized by polynomial time Turing machines which ask at most $k_1 r(n^{k_2})$ queries to an $\mathrm{NP}^{\mathrm{NP}}$ language for

constants k_1 and k_2. Note that $r(n)$ must be sub-linear. The absence of a theorem with $r(n) \geq n$ presents the greatest challenge in this area complexity theory. Such a theorem would resolve the notorious $\mathrm{P}^{\mathrm{SAT}[\log n]}$ versus $\mathrm{P}^{\mathrm{SAT}}$ problem.

Another question about the results in the previous section is whether the collapse of the Polynomial Hierarchy is optimal. Beigel, Chang and Ogiwara have shown the following result [BCO91].

Theorem 4.13. [Beigel,Chang,Ogiwara]

$$\mathrm{BH}_k = \text{co-}\mathrm{BH}_k \Longrightarrow \mathrm{PH} \subseteq \mathrm{P}^{\mathrm{NP},(\mathrm{NP}^{\mathrm{NP}}\|[k-1])}.$$

The class of languages, $\mathrm{P}^{\mathrm{NP},(\mathrm{NP}^{\mathrm{NP}}\|[k-1])}$, is recognized by polynomial time Turing machines with access to two oracles. The first oracle is an NP language and there are no restrictions on the queries to this oracle. The second oracle is an $\mathrm{NP}^{\mathrm{NP}}$ language and only $k-1$ parallel queries are allowed. Using methods described in Chapter 2, one can show that

$$\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}\|[k-1]} \subseteq \mathrm{P}^{\mathrm{NP},(\mathrm{NP}^{\mathrm{NP}}\|[k-1])} \subseteq \mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}}) \cap \text{co-}\mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}}).$$

The results of the previous section show that PH collapses to $\mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}}) \cap$ co-$\mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}})$. The newer results show a collapse to the seemingly smaller class. However, there is no clear evidence that $\mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}}) \cap$ co-$\mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}})$ should be strictly larger than $\mathrm{P}^{\mathrm{NP},(\mathrm{NP}^{\mathrm{NP}}\|[k-1])}$. On the other hand, we do know that $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}\|[k-1]} \neq \mathrm{P}^{\mathrm{NP},(\mathrm{NP}^{\mathrm{NP}}\|[k-1])}$ unless PH collapses [BCO91]. The proof for this result uses an interesting application of the hard/easy argument over the exclusive or operator.

Finally, there have been some work on the collapse of the Boolean hierarchies over other complexity classes. For example, if the Boolean hierarchy over the counting class C$_=$P collapses, then the Polynomial hierarchy relative to C$_=$P collapses as well [BCO91,Gre91]. Also, in the Chapter 5 we will examine the Boolean hierarchy over incomplete languages in NP.

4.3 Summary

We have demonstrated a closer connection between the Boolean Hierarchy and the Polynomial Hierarchy—a deeper collapse of the Boolean Hierarchy implies a deeper collapse of the Polynomial Hierarchy. We would like to think that this

relationship is a consequence of some underlying structure connecting BH_k and $\mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}})$. However, attempts to simplify the proof along these lines have failed. Is there some straightforward argument which would show that $\mathrm{BH}_k = \text{co-}\mathrm{BH}_k$ implies $\mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}})$ is closed under complementation? Could such an argument be extended to show that $\Sigma_3^{\mathrm{P}} \subseteq \mathrm{BH}_k(\mathrm{NP}^{\mathrm{NP}})$? Finally, we ask: is this collapse optimal (for $k \geq 2$) or can it be shown that $\mathrm{BH}_k = \text{co-}\mathrm{BH}_k$ implies $\mathrm{PH} \subseteq \mathrm{BH}_k$?

Chapter 5

Incomplete Sets[1]

In the preceding chapters, we have concentrated on the complexity of the complete languages in NP. We have shown that if NP/*poly* $\neq$ co-NP/*poly*, then SAT has the additional query property. That is,

$$\mathrm{P}^{\mathrm{SAT}\|[k]} \subsetneq \mathrm{P}^{\mathrm{SAT}\|[k+1]} \quad \text{and} \quad \mathrm{P}^{\mathrm{SAT}[k]} \subsetneq \mathrm{P}^{\mathrm{SAT}[k+1]}.$$

In this chapter, we turn our attention to incomplete languages in NP. Can an incomplete language have the additional query property? We know that none of the languages in P have the property. We also know that there are languages A in NP $\cap$ co-NP such that

$$\text{m-1}(A) = \mathrm{P}^{A\|[1]} = \mathrm{P}^{A\|[2]} = \cdots = \mathrm{P}^{A}.$$

So, we should ask: How low can the complexity of a set in NP be and still have the additional query property? To answer this question, we need a method of classifying the degree of complexity of incomplete sets.

5.1 High and Low Sets

Schöning [Sch83] defined the high and low hierarchies for NP to classify sets between P and NP. Very roughly, the high-low classification measures the amount of information an NP language (acting as an oracle) can give to a Σ_k^{P} machine. If $A \in \mathrm{NP}$, then for all $k \geq 0$

$$\Sigma_k^{\mathrm{P}} \subseteq \Sigma_k^{\mathrm{P},A} \subseteq \Sigma_k^{\mathrm{P},\mathrm{SAT}} = \Sigma_{k+1}^{\mathrm{P}}.$$

[1]The results presented in this chapter have appeared in [Cha89].

If $\Sigma_k^{\mathrm{P},A} = \Sigma_k^{\mathrm{P},\mathrm{SAT}}$, then one might say that the oracle A provides the Σ_k^{P} machine a lot of information—as much as SAT does. If $\Sigma_k^{\mathrm{P}} = \Sigma_k^{\mathrm{P},A}$, then A does not give the Σ_k^{P} machine anything that it cannot compute itself. In the first case we call A a high set; in the latter, a low set.

Definition:

$$\mathrm{high}_k \stackrel{\mathrm{def}}{=} \{\ A \mid A \in \mathrm{NP} \text{ and } \Sigma_k^{\mathrm{P},A} = \Sigma_k^{\mathrm{P},\mathrm{SAT}}\ \},$$
$$\mathrm{low}_k \stackrel{\mathrm{def}}{=} \{\ A \mid A \in \mathrm{NP} \text{ and } \Sigma_k^{\mathrm{P}} = \Sigma_k^{\mathrm{P},A}\ \}.$$

Clearly, $\mathrm{low}_i \subseteq \mathrm{low}_{i+1}$ and $\mathrm{high}_i \subseteq \mathrm{high}_{i+1}$. However, the high and low hierarchies are not known to have the familiar upward collapsing behavior. For example, it is not known whether $\mathrm{low}_2 = \mathrm{low}_3$ implies $\mathrm{low}_2 = \mathrm{low}_4$.

Schöning also defined a refinement of the low hierarchy [Sch85]:

$$\widehat{\mathrm{low}}_k \stackrel{\mathrm{def}}{=} \{\ A \mid A \in \mathrm{NP} \text{ and } \Delta_k^{\mathrm{P}} = \Delta_k^{\mathrm{P},A}\ \}.$$

These classes lie between the levels of the low hierarchy,

$$\mathrm{low}_0 \subseteq \widehat{\mathrm{low}}_1 \subseteq \mathrm{low}_1 \subseteq \widehat{\mathrm{low}}_2 \subseteq \mathrm{low}_2 \subseteq \widehat{\mathrm{low}}_3 \subseteq \mathrm{low}_3 \ldots$$

and allow a finer classification of the language classes in the low hierarchy.

High and low sets have many interesting properties. We briefly mention some here. See [KS85,Sch82,Sch83,Sch85,Sch88] for proofs.

- $\mathrm{low}_k = \mathrm{high}_k \iff \mathrm{PH} \subseteq \Sigma_k^{\mathrm{P}}$.
- If PH is infinite, then $\exists I \in \mathrm{NP}\ \forall k\ [I \notin \mathrm{high}_k \text{ and } I \notin \mathrm{low}_k]$.
- If S is sparse and $S \in \mathrm{NP}$, then $S \in \widehat{\mathrm{low}}_2$.
- If $A \in \mathrm{NP}$ and for some sparse set S, $A \in \mathrm{P}^S$, then $A \in \mathrm{low}_3$.
- $\mathrm{R} \subseteq \mathrm{BPP} \cap \mathrm{NP} \subseteq \mathrm{low}_2$.
- $\mathrm{high}_0 = \{\ A \mid A \text{ is } \leq_{\mathrm{T}}^{\mathrm{P}}\text{-complete for NP}\ \}$.
- $\mathrm{low}_0 = \mathrm{P}$.
- $\mathrm{low}_1 = \mathrm{NP} \cap \text{co-NP}$.
- Graph Isomorphism $\in \mathrm{low}_2$.

5.2 Bounded Queries to Incomplete Sets

With all these definitions in place we can pose some questions originally posed for SAT. For example, we would like to know what happens if $\mathrm{P}^{A\|[k]} = \mathrm{P}^{A\|[k+1]}$? if $\mathrm{P}^{A[k]} = \mathrm{P}^{A[k+1]}$? or if $\mathrm{BL}_k(A) \leq^{\mathrm{P}}_{\mathrm{m}} \text{co-BL}_k(A)$? Also, we would like to know the relationship between queries to SAT and queries to A. We know $\mathrm{P}^{A\|[k]} \subseteq \mathrm{P}^{\mathrm{SAT}\|[k]}$, but where is $\mathrm{P}^{A\|[k]}$ in relation to $\mathrm{P}^{\mathrm{SAT}\|[k-1]}$? Instead of asking "Is one question to SAT as powerful as two?" we ask "Is one question to SAT as powerful as two questions to some other oracle?" The following theorem answers some of these questions.

Theorem 5.1. If $A,\ B \in \mathrm{NP}$ and $\mathrm{BL}_k(A) \leq^{\mathrm{P}}_{\mathrm{m}} \text{co-BL}_k(B)$, then $A \in \widehat{\mathrm{low}_3}$.

We prove this theorem in two parts. Lemma 5.2 is a straightforward generalization of Theorem 4.2. We relegate the proof of Lemma 5.2 to the next section.

Lemma 5.2. Suppose $B \in \mathrm{NP}$ and $\mathrm{BL}_k(A) \leq^{\mathrm{P}}_{\mathrm{m}} \text{co-BL}_k(B)$. Then, there exists an NP machine $\widetilde{N}_A$ and a polynomial advice function f computable in Δ^{P}_3 such that for all $m \geq |x|$,

$$x \in \overline{A} \iff \widetilde{N}_A(x, f(1^m)) \text{ accepts.}$$

(N.B. In Lemma 5.2, we do not assume $A \in \mathrm{NP}$.)

Lemma 5.3. If $A, B \in \mathrm{NP}$ and $\mathrm{BL}_k(A) \leq^{\mathrm{P}}_{\mathrm{m}} \text{co-BL}_k(B)$, then $A \in \widehat{\mathrm{low}_3}$.

Proof: The intuition for this proof is that since $A \in \mathrm{NP}$ and and since Lemma 5.2 provides us with an NP/f machine which recognizes $\overline{A}$, any NP^A can be simulated by an NP/f machine. However, it is not quite proper to say that $\mathrm{NP}^A \subseteq \mathrm{NP}/f$, because an NP^A machine can query strings of polynomial length, but f only gives advice about strings in A of linear length. To resolve this notational difficulty we define

$$f_i(1^n) = \langle f(1), f(11), \ldots, f(1^{n^i}) \rangle \quad \text{and} \quad \mathcal{F} = \{\ f_i \mid i \geq 1\ \}.$$

Then, it is clear that $\mathrm{NP}^A \subseteq \mathrm{NP}/\mathcal{F}$. Moreover, since f is computable in Δ^{P}_3, each f_i is also computable in Δ^{P}_3. The main idea of the proof is to show that

$$\Delta^{\mathrm{P},A}_3 = \Delta^{\mathrm{P},\mathrm{NP}^A}_2 \subseteq \Delta^{\mathrm{P},\mathrm{NP}/\mathcal{F}}_2.$$

Since the advice functions in $\mathcal{F}$ are computable in Δ_3^{P},

$$\Delta_3^{\mathrm{P},A} = \Delta_2^{\mathrm{P},\mathrm{NP}^A} \subseteq \Delta_2^{\mathrm{P},\mathrm{NP}/\mathcal{F}} \subseteq \Delta_2^{\mathrm{P},\mathrm{NP}} = \Delta_3^{\mathrm{P}}.$$

Now, we show how the oracle A can be replaced by advice functions in $\mathcal{F}$. Let L be any language in NP^A. Then, $L = L(N_i^A)$ for some NP machine N_i. Let n^r be an upper bound on the running time of N_i, then we know that on input x, $N_i^A(x)$ does not query strings of length greater than $|x|^r$. We construct an NP/f_r machine N_j which accepts L. N_j assumes that it is given the advice $f_r(1^n)$ and simulates N_i directly until N_i asks if some string q is in A. N_j guesses if $q \in A$, then verifies this guess by either running $N_A(q)$ or $\widetilde{N}_A(q, f_r(1^n))$. Exactly one of these guesses is correct, so

$$N_i^A(x) \text{ accepts} \iff N_j(x, f_r(1^n)) \text{ accepts.}$$

Therefore, $A \in \widehat{\mathrm{low}}_3$. □

The theorem gives us a sufficient condition for a set to be in $\widehat{\mathrm{low}}_3$. The following corollary clarifies the picture somewhat.

Corollary 5.4.

If $A \in \mathrm{NP}$ and one of the following conditions holds, then $A \in \widehat{\mathrm{low}}_3$.

1. $\mathrm{P}^{A[2]} \subseteq \mathrm{P}^{\mathrm{SAT}[1]}$.
2. $\mathrm{P}^{A\|[k+1]} \subseteq \mathrm{P}^{\mathrm{SAT}\|[k]}$, for some $k \geq 1$.
3. $\mathrm{P}^{A\|[k+1]} = \mathrm{P}^{A\|[k]}$, for some $k \geq 1$.
4. $\mathrm{P}^{A[k+1]} = \mathrm{P}^{A[k]}$, for some $k \geq 1$.

Proof:

1. If $\mathrm{P}^{A[2]} \subseteq \mathrm{P}^{\mathrm{SAT}[1]}$, then $\mathrm{BL}_2(A) \in \mathrm{P}^{A\|[2]} \subseteq \mathrm{P}^{A[2]} \subseteq \mathrm{P}^{\mathrm{SAT}[1]} \subseteq \text{co-}\mathrm{BH}_2$.

   However, co-BL_2(SAT) is $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for the class co-BH_2. So $\mathrm{BL}_2(A)$ $\leq_{\mathrm{m}}^{\mathrm{P}}$ co-BL_2(SAT). Then, by Theorem 5.1, A $\in \widehat{\mathrm{low}}_3$.

2. If $\mathrm{P}^{A\|[k+1]} \subseteq \mathrm{P}^{\mathrm{SAT}\|[k]}$, then $\mathrm{BL}_{k+1}(A) \in \mathrm{P}^{A\|[k+1]} \subseteq \mathrm{P}^{\mathrm{SAT}\|[k]} \subseteq \text{co-}\mathrm{BH}_{k+1}$. Since co-$\mathrm{BL}_{k+1}$(SAT) is $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for co-$\mathrm{BH}_{k+1}$, there exists a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from $\mathrm{BL}_{k+1}(A)$ to co-BL_{k+1}(SAT). Hence, $A \in \widehat{\mathrm{low}}_3$.

3. $\mathrm{P}^{A\|[k+1]} = \mathrm{P}^{A\|[k]}$ implies $\mathrm{P}^{A\|[k+1]} \subseteq \mathrm{P}^{\mathrm{SAT}\|[k]}$, so A $\in \widehat{\mathrm{low}}_3$ by part 2.

4. $\mathrm{P}^{A[k+1]} = \mathrm{P}^{A[k]}$ implies $\mathrm{P}^{A[2^k]} \subseteq \mathrm{P}^{A[k]}$, because the whole query hierarchy based on A collapses. Thus, $\mathrm{P}^{A\|[2^k]} \subseteq \mathrm{P}^{A[2^k]} \subseteq \mathrm{P}^{A[k]} \subseteq \mathrm{P}^{A\|[2^k-1]}$. Then, by part 3, $\mathrm{A} \in \widehat{\mathrm{low}_3}$. □

Parts 3 and 4 of Corollary 5.4 state that if the parallel or serial query hierarchies built from A collapse, then A is not very hard. Part 2 says that if a set A is not in $\widehat{\mathrm{low}_3}$, then not only is the parallel query hierarchy built from A proper, it also rises in lock step with $\mathrm{QH}_{\|}$ (see figure 1).

For high sets, Theorem 5.1 would imply a collapse of PH.

Corollary 5.5. If $A \in \mathrm{high}_j$ and one of the following holds, then PH is finite.

1. $\mathrm{P}^{A[2]} \subseteq \mathrm{P}^{\mathrm{SAT}[1]}$.
2. $\mathrm{P}^{A\|[k+1]} \subseteq \mathrm{P}^{\mathrm{SAT}\|[k]}$, for some $k \geq 1$.
3. $\mathrm{P}^{A\|[k+1]} = \mathrm{P}^{A\|[k]}$, for some $k \geq 1$.
4. $\mathrm{P}^{A[k+1]} = \mathrm{P}^{A[k]}$, for some $k \geq 1$.

Proof: Let $i = \max(3, j)$. By Corollary 5.4, $A \in \widehat{\mathrm{low}_3} \subseteq \mathrm{low}_i$, so $\Sigma_i^{\mathrm{P}} = \Sigma_i^{\mathrm{P},A}$. However, $A \in \mathrm{high}_j \subseteq \mathrm{high}_i$ implies that $\Sigma_i^{\mathrm{P},A} = \Sigma_{i+1}^{\mathrm{P}}$. Thus, $\Sigma_i^{\mathrm{P}} = \Sigma_i^{\mathrm{P},A} = \Sigma_{i+1}^{\mathrm{P}}$ and $\mathrm{PH} \subseteq \Sigma_i^{\mathrm{P}}$. □

As a special case of Corollary 5.5, if the query hierarchy built from an NP $\leq_{\mathrm{T}}^{\mathrm{P}}$-complete set collapses, then PH collapses to Δ_3^{P}. In the following corollary, we will use this fact to answer the question "Is one query to SAT as powerful as two queries to some other oracle?"

Corollary 5.6. If there exists A such that $\mathrm{P}^{A[2]} = \mathrm{P}^{\mathrm{SAT}[1]}$, then $\mathrm{PH} \subseteq \Delta_3^{\mathrm{P}}$.

Proof: Suppose that $\mathrm{P}^{A[2]} = \mathrm{P}^{\mathrm{SAT}[1]}$, then $A \in \mathrm{P}^{A[2]} = \mathrm{P}^{\mathrm{SAT}[1]}$. So, we know that $A \leq_{\mathrm{m}}^{\mathrm{P}} \mathrm{SAT} \oplus \overline{\mathrm{SAT}}$ via some polynomial time function g, since $\mathrm{SAT} \oplus \overline{\mathrm{SAT}}$ is $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for $\mathrm{P}^{\mathrm{SAT}[1]}$, where $X \oplus Y$ is defined by

$$X \oplus Y \stackrel{\mathrm{def}}{=} \{\ 0x \mid x \in X\ \} \cup \{\ 1y \mid y \in Y\ \}.$$

We split A into two sets,

$$A_0 = \{\ x \mid x \in A \text{ and } g(x) = 0F \text{ for some } F\ \},$$
$$A_1 = \{\ x \mid x \in A \text{ and } g(x) = 1F \text{ for some } F\ \}.$$

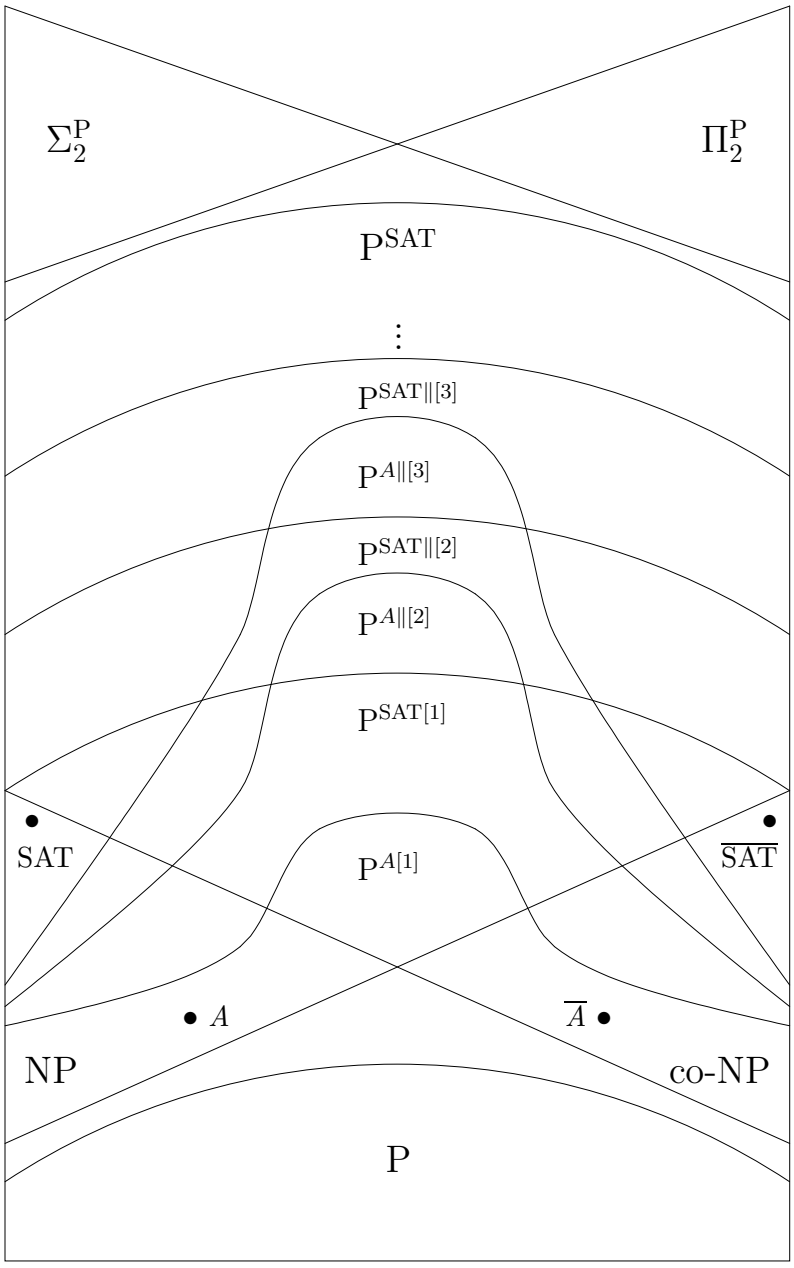

Figure 5.1: Let A be an incomplete set in NP $-$ $\widehat{\mathrm{low}_3}$.

Clearly, $A_0 \in \mathrm{NP}$ and $A_1 \in \text{co-NP}$. Now let $C = A_0 \oplus \overline{A_1}$. One can easily see that $C \in \mathrm{NP}$ and that $\mathrm{P}^{A[k]} = \mathrm{P}^{C[k]}$, for all k. So, $\mathrm{P}^{C[2]} = \mathrm{P}^{A[2]} = \mathrm{P}^{\mathrm{SAT}[1]}$. However, $\mathrm{SAT} \in \mathrm{P}^{C[2]}$ implies that C is $\leq^{\mathrm{P}}_{\mathrm{T}}$-complete for NP—i.e., C is a high$_0$ set. Then, by part 1 of Corollary 5.5, PH collapses to Δ^{P}_3. □

Corollary 5.6 holds for parallel queries as well. That is, $\mathrm{P}^{A\|[2]} = \mathrm{P}^{\mathrm{SAT}\|[1]} \Longrightarrow \mathrm{PH} \subseteq \Delta^{\mathrm{P}}_3$. In fact, for parallel queries, we can generalize the statement to work for arbitrary constants.

Corollary 5.7. If there exists any set A such that $\mathrm{P}^{A\|[r]} = \mathrm{P}^{\mathrm{SAT}\|[k]}$, and $r > k > 0$, then $\mathrm{PH} \subseteq \Delta^{\mathrm{P}}_3$.

Proof: Since $r > k$, we have $\mathrm{P}^{A\|[k+1]} \subseteq \mathrm{P}^{A\|[r]} = \mathrm{P}^{\mathrm{SAT}\|[k]} \subseteq \text{co-BH}_{k+1}$. So, $\mathrm{BL}_{k+1}(A) \leq^{\mathrm{P}}_{\mathrm{m}} \text{co-BL}_{k+1}(\mathrm{SAT})$. Then, by Lemma 5.2 there exists an advice function f computable in Δ^{P}_3 such that $\overline{A} \in \mathrm{NP}/f$. Since $\mathrm{P}^{\overline{A}\|[k+1]} = \mathrm{P}^{A\|[k+1]}$ and $\mathrm{P}^{A\|[k+1]} \subseteq \text{co-BH}_{k+1}$, repeating the argument for $\overline{A}$ yields f' such that $A \in \mathrm{NP}/f'$. Let g be the advice function which concatenates the advice from f and f'. I.e., $g(1^n) = \langle f(1^n), f'(1^n)\rangle$. As before, we define

$$g_i(1^n) = \langle g(1), g(11), \ldots, g(1^{n^i})\rangle \quad \text{and} \quad \mathcal{G} = \{\ g_i \mid i \geq 1\ \}.$$

Then,

$$\mathrm{NP}^A \subseteq \mathrm{NP}/\mathcal{G}.$$

However, $\mathrm{P}^{A\|[r]} = \mathrm{P}^{\mathrm{SAT}\|[k]}$ implies $\mathrm{SAT} \in \mathrm{P}^A$. So, $\mathrm{NP}^{\mathrm{SAT}} \subseteq \mathrm{NP}^A$. Thus,

$$\Delta^{\mathrm{P}}_4 = \Delta^{\mathrm{P},\mathrm{NP}^{\mathrm{SAT}}}_2 \subseteq \Delta^{\mathrm{P},\mathrm{NP}^A}_2 \subseteq \Delta^{\mathrm{P},\mathrm{NP}/\mathcal{G}}_2 \subseteq \Delta^{\mathrm{P}}_3,$$

(because every function in $\mathcal{G}$ is computable in Δ^{P}_3). Therefore, $\mathrm{PH} \subseteq \Delta^{\mathrm{P}}_3$. □

Corollaries 5.6 and 5.7 say that if PH is infinite, then there do not exist sets which contain *fractional* information with respect to SAT. For example, there would not be a set A where 3 parallel queries to A is exactly 2 parallel queries to SAT. (Each query to such a set A, would give as much information as two-thirds of a query to SAT.) However, we still do not know if there can be sets B such that $\mathrm{P}^{B\|[2]} = \mathrm{P}^{\mathrm{SAT}\|[3]}$. If we can show that such sets do not exist (under certain hypotheses), then we can put forward the thesis that parallel queries to SAT are "atomic" or "indivisible" in some sense.

Our theorems for the serial query hierarchies are somewhat weaker than those for the parallel query hierarchy. For parallel queries, we know that $\mathrm{P}^{A\|[k+1]} \not\subseteq \mathrm{P}^{\mathrm{SAT}\|[k]}$ unless $A \in \widehat{\mathrm{low}_3}$. We do not have a corresponding lemma for serial queries. That is, we do not know the consequences of the assumption that $\mathrm{P}^{A[k+1]} \subseteq \mathrm{P}^{\mathrm{SAT}[k]}$. If A is a set that has AND_2 and OR_2, then

$$\mathrm{P}^{A[k+1]} \subseteq \mathrm{P}^{\mathrm{SAT}[k]} \Longrightarrow \mathrm{P}^{A\|[2^{k+1}-1]} \subseteq \mathrm{P}^{\mathrm{SAT}\|[2^k-1]},$$

and the results for parallel queries would carry over. However, in the general case, A may not have AND_2 or OR_2. In the following corollaries, we explore technical conditions that allow us to strengthen the results about the serial query hierarchies.

Corollary 5.8. If $A \in \mathrm{NP} - \widehat{\mathrm{low}_3}$ then there exist $B \in \mathrm{NP}$ such that $A \leq^{\mathrm{P}}_{\mathrm{m}} B$, $\mathrm{P}^A = \mathrm{P}^B$ and for all k, $\mathrm{P}^{B[k+1]} \not\subseteq \mathrm{P}^{\mathrm{SAT}[k]}$.

Proof: If $\mathrm{P}^B = \mathrm{P}^A$ then A and B are in the same high or low level. So, this corollary also says that for every high and low level above $\widehat{\mathrm{low}_3}$ there is a set B whose *serial* query hierarchy rises in lock step with the *serial* query hierarchy for SAT. We construct B by modifying A slightly, so that B has OR_2 and AND_2. Then, by Theorem 3.19, $\mathrm{P}^{B\|[2^k-1]} = \mathrm{P}^{B[k]}$.

$$B = \{\ \langle F, x_1, \ldots, x_n\rangle \mid F \text{ is a boolean formula over } n \text{ variables } y_1, \ldots, y_n \text{ without negation, and when } y_i \text{ is evaluated as "} x_i \in A\text{", } F \text{ evaluates to 1. } \}$$

For example, if $a_1, a_2 \in A$ and $a_3, a_4 \notin A$ then:

$$\langle y_1 \wedge y_2, a_1, a_2\rangle \in B, \quad \langle y_1 \wedge y_2, a_1, a_3\rangle \notin B,$$
$$\langle y_1 \vee y_2, a_1, a_3\rangle \in B, \quad \langle y_1 \vee y_2, a_4, a_3\rangle \notin B.$$

Clearly, B is in NP and $A \leq^{\mathrm{P}}_{\mathrm{m}} B$. Also, $\mathrm{P}^B = \mathrm{P}^A$ implies $B \in \mathrm{NP} - \widehat{\mathrm{low}_3}$. Moreover, $\mathrm{OR}_2(B) \leq^{\mathrm{P}}_{\mathrm{m}} B$ and $\mathrm{AND}_2(B) \leq^{\mathrm{P}}_{\mathrm{m}} B$, so $\mathrm{P}^{B\|[2^k-1]} = \mathrm{P}^{B[k]}$. Thus,

$$\mathrm{BL}_{2^k}(B) \in \mathrm{P}^{B\|[2^k]} \subseteq \mathrm{P}^{B\|[2^{k+1}-1]} = \mathrm{P}^{B[k+1]} \quad \text{and}$$
$$\mathrm{P}^{\mathrm{SAT}[k]} = \mathrm{P}^{\mathrm{SAT}\|[2^k-1]} \subseteq \text{co-}\mathrm{BH}_{2^k},$$

Since we assume that $\mathrm{P}^{B[k+1]} \subseteq \mathrm{P}^{\mathrm{SAT}[k]}$, $\mathrm{BL}_{2^k}(B) \leq^{\mathrm{P}}_{\mathrm{m}} \text{co-}\mathrm{BL}_{2^k}(\mathrm{SAT})$. This contradicts the assumption that A is not in $\widehat{\mathrm{low}_3}$. □

In the following corollary, we examine the existence of incomplete sets whose serial query hierarchies rise in lock step with QH. (Note that it may be the case

that all of the sets in NP $-$ $\widehat{\text{low}_3}$ are complete in some sense. However, in this case PH is finite.)

Corollary 5.9. If the Polynomial Hierarchy is infinite, then there exists a set $B \in \text{NP}$ such that for all k, $\text{P}^{B[k+1]} \not\subseteq \text{P}^{\text{SAT}[k]}$, but B is not high.

Proof: If PH is infinite, then by a Ladner-like delayed diagonalization [Lad75, Sch82] we can construct a set $I \in \text{NP}$ that is neither high nor low (I stands for intermediate). In particular, $I \notin \widehat{\text{low}_3}$, so using Corollary 5.8 we can obtain a set B such that for all k,

$$\text{P}^{B[k+1]} \not\subseteq \text{P}^{\text{SAT}[k]}.$$

Since $\text{P}^B = \text{P}^I$, B is intermediate iff I is intermediate. So, B is not high. □

If B is not high, then B is not NP-hard under many-one, Turing, strong non-deterministic or other assorted reductions. In particular, B is not Turing complete for NP, so $\text{SAT} \notin \text{P}^B$. So, Corollary 5.9 says that the serial query hierarchy built from B rises in lock step with QH but never captures SAT.

5.3 Proof of Lemma 5.2

Lemma 5.2. Suppose $B \in \text{NP}$ and $\text{BL}_k(A) \leq^{\text{P}}_{\text{m}} \text{co-BL}_k(B)$. Then, there exists an NP machine $\widetilde{N}_A$ and a polynomial advice function f computable in Δ^{P}_3 such that for all $m \geq |x|$,

$$x \in \overline{A} \iff \widetilde{N}_A(x, f(1^m)) \text{ accepts.}$$

Note that we do not assume $A \in \text{NP}$. However,

$$A \leq^{\text{P}}_{\text{m}} \text{BL}_k(A), \quad \text{BL}_k(A) \leq^{\text{P}}_{\text{m}} \text{co-BL}_k(B), \quad \text{and} \quad \text{co-BL}_k(B) \in \text{P}^{\text{NP}},$$

implies that A must be in P^{NP}. The proof of this lemma uses the hard/easy argument. The outline of this proof is the same as the proofs of Lemmas 4.1 and 4.2, but we include the proof for the sake of completeness. The first step in the hard/easy argument is to define the hard strings. For the rest of this section, we fix A and B to be the languages in the hypothesis of Lemma 5.2 and we let h be the $\leq^{\text{P}}_{\text{m}}$-reduction from $\text{BL}_k(A)$ to $\text{co-BL}_k(B)$.

Definition: We call $\langle 1^m, \vec{x}\rangle = \langle 1^m, x_1, \ldots, x_j\rangle$ a *hard sequence* if $j = 0$ or if all of the following hold:

1. $1 \leq j \leq k-1$.
2. $|x_j| \leq m$.
3. $x_j \in \overline{A}$.
4. $\langle 1^m, x_1, \ldots, x_{j-1}\rangle$ is a hard sequence.
5. $\forall \vec{y} = \langle y_1, \ldots, y_\ell\rangle \in \{0,1\}^{\leq m\times\ell},\ \pi_{\ell+1}{\circ}h(\vec{y}, \vec{x}^R) \in \overline{B}$, where $\ell = k - j$.

The next step of the hard/easy argument is to show that a hard sequence induces a reduction of the Boolean languages at a lower level.

Lemma 5.10. Suppose that $\langle 1^m, \vec{x}\rangle = \langle 1^m, x_1, \ldots, x_j\rangle$ is a hard sequence. Then for all $\vec{y} = \langle y_1, \ldots, y_\ell\rangle \in \{0,1\}^{\leq m\times\ell}$ (where $\ell = k - j$)

$$\vec{y} \in \mathrm{BL}_\ell(A) \iff \pi_{(1,\ell)}{\circ}h(\vec{y}, \vec{x}^R) \in \text{co-BL}_\ell(B).$$

Proof: (by induction on j)
Base Case $j = 0$: In this case, $\vec{x}$ is the empty sequence, so $\ell = k$. By the hypothesis, h reduces $\mathrm{BL}_k(A)$ to co-$\mathrm{BL}_k(B)$, so

$$\vec{y} \in \mathrm{BL}_k(A) \iff h(\vec{y}) \in \text{co-BL}_k(B).$$

However, $\pi_{(1,k)}{\circ}h(\vec{y}) = h(\vec{y})$, so

$$\vec{y} \in \mathrm{BL}_k(A) \iff \pi_{(1,k)}{\circ}h(\vec{y}) \in \text{co-BL}_k(B).$$

Induction Case $j+1$:

Suppose that the lemma holds for j, we show that it holds for $j+1$. Let $\ell = k - j$. Let $\langle 1^m, \vec{x}\rangle = \langle 1^m, x_1, \ldots, x_{j+1}\rangle$ be a hard sequence. Then, by definition, $\langle 1^m, \vec{x}\,'\rangle = \langle 1^m, x_1, \ldots, x_j\rangle$ is also a hard sequence. Since the lemma holds for j, for all $\vec{y} = \langle y_1, \ldots, y_{\ell-1}\rangle \in \{0,1\}^{\leq m\times(\ell-1)}$

$$\langle \vec{y}, x_{j+1}\rangle \in \mathrm{BL}_\ell(A) \iff \pi_{(1,\ell)}{\circ}h(\vec{y}, x_{j+1}, \vec{x}\,'^R) \in \text{co-BL}_\ell(B).$$

If ℓ is even, then by the definitions of $\mathrm{BL}_\ell(A)$ and co-$\mathrm{BL}_\ell(B)$

$$\begin{aligned} &\vec{y} \in \mathrm{BL}_{\ell-1}(A) \ \text{ and } \ x_{j+1} \in \overline{A} \\ &\quad \iff \pi_{(1,\ell-1)}{\circ}h(\vec{y}, \vec{x}^R) \in \text{co-BL}_{\ell-1}(B) \text{ or } \pi_\ell{\circ}h(\vec{y}, \vec{x}^R) \in B. \end{aligned} \tag{5.1}$$

If ℓ is odd, then by the definitions of $\mathrm{BL}_\ell(A)$ and co-$\mathrm{BL}_\ell(B)$

$$\begin{aligned} &\vec{y} \in \mathrm{BL}_{\ell-1}(A) \ \text{ or } \ x_{j+1} \in A \\ &\quad \iff \pi_{(1,\ell-1)} \circ h(\vec{y}, \vec{x}^R) \in \text{co-}\mathrm{BL}_{\ell-1}(B) \text{ and } \pi_\ell \circ h(\vec{y}, \vec{x}^R) \in \overline{B}, \end{aligned} \tag{5.2}$$

or (by negating both sides of the iff),

$$\begin{aligned} &\vec{y} \notin \mathrm{BL}_{\ell-1}(A) \ \text{ and } \ x_{j+1} \in \overline{A} \\ &\quad \iff \pi_{(1,\ell-1)} \circ h(\vec{y}, \vec{x}^R) \notin \text{co-}\mathrm{BL}_{\ell-1}(B) \text{ or } \pi_\ell \circ h(\vec{y}, \vec{x}^R) \in B. \end{aligned} \tag{5.3}$$

Since $\langle 1^m, x_1, \ldots, x_{j+1} \rangle$ is a hard sequence, from parts 3 and 5 of the definition of a hard sequence we know that $x_{j+1} \in \overline{A}$ and $\pi_\ell \circ h(\vec{y}, \vec{x}^R) \in \overline{B}$. Therefore in equations (5.1) and (5.3), the second conjunct on the left side is true and the second disjunct on the right side is false. Hence

$$\vec{y} \in \mathrm{BL}_{\ell-1}(A) \iff \pi_{(1,\ell-1)} \circ h(\vec{y}, \vec{x}^R) \in \text{co-}\mathrm{BL}_{\ell-1}(B).$$

Then replacing $k - j$ for ℓ, we have

$$\begin{aligned} &\langle y_1, \ldots, y_{k-(j+1)} \rangle \in \mathrm{BL}_{k-(j+1)}(A) \\ &\quad \iff \pi_{(1,k-(j+1))} \circ h(\langle y_1, \ldots, y_{k-(j+1)}, x_{j+1}, \ldots, x_1 \rangle) \in \text{co-}\mathrm{BL}_{k-(j+1)}(B). \end{aligned}$$

So, we have established the lemma for $j + 1$. □

Again, note that Lemmas 4.1 and 5.10 are virtually identical. However, it is important to note that a hard sequence induces a reduction from $\mathrm{BL}_\ell(A)$ to co-$\mathrm{BL}_\ell(B)$ and not from co-$\mathrm{BL}_\ell(B)$ to $\mathrm{BL}_\ell(A)$. In the proofs of Chapter 4, $A = B = \mathrm{SAT}$, so such subtleties are not important. We point this out because one way to simplify the presentation of the proofs in Chapter 4 is to say that on even levels you use the reduction from BL_k to co-BL_k and on odd levels you use the reduction from co-BL_k to BL_k. The motivation for this argument is that you always reduce a conjunction to a disjunction. This approach fails when $A \neq B$.

Now we show that maximal hard sequences induce "reductions" from $\overline{A}$ to B.

Lemma 5.11. Suppose that $\langle 1^m, \vec{x} \rangle = \langle 1^m, x_1, \ldots, x_j \rangle$ is a maximal hard sequence. Then for all $v \in \{0,1\}^{\leq m}$

$$v \in \overline{A} \iff \exists\, \vec{y} = \langle y_1, \ldots, y_{\ell-1} \rangle \in \{0,1\}^{\leq m \times (\ell-1)},\ \pi_\ell \circ h(\vec{y}, v, \vec{x}^R) \in B,$$

where $\ell = k - j$.

Proof:

If $j = k-1$ ($\vec{y}$ is the empty sequence), then by Lemma 5.10 for all $v \in \{0,1\}^{\leq m}$

$$v \in \mathrm{BL}_1(A) \iff \pi_1 \circ h(y, \vec{x}^R) \in \text{co-BL}_1(B).$$

However, $\mathrm{BL}_1(A) = A$ and $\text{co-BL}_1(B) = \overline{B}$. So, we have

$$v \in A \iff \pi_1 \circ h(v, \vec{x}^R) \in \overline{B}.$$

or (by negating both sides of the iff)

$$v \in \overline{A} \iff \pi_1 \circ h(v, \vec{x}^R) \in \overline{B}.$$

Thus, the lemma holds when $j = k-1$ (i.e. when $\vec{y}$ is the empty sequence).

Now, consider the case when $j < k-1$.

($\Rightarrow$) Suppose $v \in \overline{A}$. Since $\langle 1^m, x_1, \ldots, x_j \rangle$ is maximal, $\langle 1^m, x_1, \ldots, x_j, v \rangle$ is not a hard sequence. However, $j+1 \leq k-1$, $|v| \leq m$, $v \in \overline{A}$, and $\langle 1^m, x_1, \ldots, x_j \rangle$ is a hard sequence. So, $\langle 1^m, x_1, \ldots, x_j, v \rangle$ must fail to be a hard sequence by failing to satisfy condition 5 of the definition of hard sequences. Thus,

$$\exists\, \vec{y} = \langle y_1, \ldots, y_{\ell-1} \rangle \in \{0,1\}^{\leq m \times (\ell-1)},\ \pi_\ell \circ h(\vec{y}, v, \vec{x}^R) \in B.$$

($\Leftarrow$) Suppose that for some $\vec{y} = \langle y_1, \ldots, y_{\ell-1} \rangle \in \{0,1\}^{\leq m \times (\ell-1)}$

$$\pi_\ell \circ h(\vec{y}, v, \vec{x}^R) \in B.$$

Since $\langle 1^m, \vec{x} \rangle$ is a hard sequence for length m, by Lemma 4.1

$$\langle \vec{y}, v \rangle \in \mathrm{BL}_\ell(A) \iff \pi_{(1,\ell)} \circ h(\vec{y}, v, \vec{x}^R) \in \text{co-BL}_\ell(B).$$

If ℓ is even, then by the definitions of $\mathrm{BL}_\ell(A)$ and $\text{co-BL}_\ell(B)$

$$\begin{aligned} &\vec{y} \in \mathrm{BL}_{\ell-1}(A) \ \text{ and } \ v \in \overline{A} \\ &\quad \iff \pi_{(1,\ell-1)} \circ h(\vec{y}, v, \vec{x}^R) \in \text{co-BL}_{\ell-1}(B) \text{ or } \pi_\ell \circ h(\vec{y}, v, \vec{x}^R) \in B. \end{aligned} \tag{5.4}$$

If ℓ is odd, then by the definitions of $\mathrm{BL}_\ell(A)$ and $\text{co-BL}_\ell(B)$

$$\begin{aligned} &\vec{y} \in \mathrm{BL}_{\ell-1}(A) \ \text{ or } \ v \in A \\ &\quad \iff \pi_{(1,\ell-1)} \circ h(\vec{y}, v, \vec{x}^R) \in \text{co-BL}_{\ell-1}(B) \text{ and } \pi_\ell \circ h(\vec{y}, v, \vec{x}^R) \in \overline{B}, \end{aligned} \tag{5.5}$$

or (by negating both sides of the iff)

$$\begin{aligned} &\vec{y} \notin \mathrm{BL}_{\ell-1}(A) \ \text{ and } \ v \in \overline{A} \\ &\quad \iff \pi_{(1,\ell-1)} \circ h(\vec{y}, v, \vec{x}^R) \notin \text{co-BL}_{\ell-1}(B) \text{ or } \pi_\ell \circ h(\vec{y}, v, \vec{x}^R) \in B. \end{aligned} \tag{5.6}$$

In either case, we already know by hypothesis that $\pi_\ell \circ h(\vec{y}, v, \vec{x}^R) \in B$, so the right sides of the iff in equations (5.4) and (5.6) are satisfied. Therefore, the left sides of equations (5.4) and (5.6) must also be satisfied, and we have $v \in \overline{A}$. □

Finally, we are ready to construct the NP/*poly* machine $\widetilde{N}_A$ and the advice function f as required by Lemma 5.2.

Proof: (of Lemma 5.2)

As the reader has probably suspected, the advice function f is just a function which, on input 1^m, prints out a maximal hard sequence for length m, say $\langle 1^m, \vec{x} \rangle = \langle 1^m, x_1, \ldots, x_j \rangle$. Given this advice, an NP machine can verify that $v \notin A$ for all v, $|v| \leq m$ using Lemma 5.11. That is, $\widetilde{N}_A$ on input $(v, \langle 1^m, \vec{x} \rangle)$ does the following

1. Guess $k - j - 1$ strings, $y_1, \ldots, y_{k-j-1}$, of length less than or equal to m.
2. Compute $\langle u_1, \ldots, u_k \rangle = h(y_1, \ldots, y_{k-j-1}, v, x_j, \ldots, x_1)$.
3. Accept if $u_{k-j} \in B$.

All that remains to show is that f is computable in Δ_3^{P}. For the sake of precision, let f be the function that prints out the lexically largest hard sequence of maximum order. Since $A \in \mathrm{P}^{\mathrm{NP}}$, an $\mathrm{NP}^{\mathrm{NP}}$ machine can guess a sequence and check that it is a hard sequence. So, using binary search, a Δ_3^{P} machine can find the lexically largest hard sequence of maximum order. □

5.4 Summary

In this chapter we have shown that except for the sets in $\widehat{\mathrm{low}_3}$, all the languages in NP have the additional query property. In fact, assuming that PH is infinite, there even exist incomplete languages such that each additional query to this language allows polynomial time computations to recognize new languages.

However, many open questions remain. For example, we know that query hierarchies built from sets in P always collapse. We also know that for any $A \in \mathrm{NP} \cap \text{co-NP}$ there exists $B \in \mathrm{NP} \cap \text{co-NP}$ such that $A \leq_{\mathrm{m}}^{\mathrm{P}} B$ and the query hierarchy built from B collapses. Are there sets in $\mathrm{NP} \cap \text{co-NP}$ that have proper query hierarchies? What about sets between $\widehat{\mathrm{low}_3}$ and $\mathrm{NP} \cap \text{co-NP}$? Are their query hierarchies proper? Many interesting language classes live in this region,

including sparse sets in NP, NP $\cap$ P/poly, R and BPP $\cap$ NP. Can we show that any of these sets have proper or collapsing hierarchies? Also, we would like to strengthen the results for serial query hierarchies. At the present time, we only know that having AND_2 and OR_2 is a sufficient condition.

Chapter 6

Unique Satisfiability and Randomized Reductions[1]

In this chapter, we apply the hard/easy argument to determine the complexity of the unique satisfiability problem, USAT. We show that

- if USAT $\equiv_{\mathrm{m}}^{\mathrm{P}} \overline{\mathrm{USAT}}$, then $\mathrm{D}^{\mathrm{P}} = \text{co-}\mathrm{D}^{\mathrm{P}}$ and PH collapses.
- if USAT $\in \text{co-}\mathrm{D}^{\mathrm{P}}$, then NP/*poly* = co-NP/*poly* and PH collapses.
- if USAT has OR_ω, then NP/*poly* = co-NP/*poly* and PH collapses.

The proofs of these results use only the fact that USAT is complete for D^{P} under randomized reductions—even though the probability bound of these reductions may be low. Furthermore, these results show that the structural complexity of USAT and of the sets many-one complete for D^{P} are very similar. So, they lend support to the argument that even sets complete under "weak" randomized reductions can capture the properties of the many-one complete sets. However, under these "weak" randomized reductions, USAT is complete for $\mathrm{P}^{\mathrm{SAT}[\log n]}$ as well, and in this case, USAT does not capture the properties of the sets many-one complete for $\mathrm{P}^{\mathrm{SAT}[\log n]}$. To explain this anomaly, we develop a concept of the *threshold behavior* of randomized reductions.

Randomized reductions have been defined in many ways and without much consideration for the error probability that the reductions are required to achieve.

[1]The results presented in this chapter were discovered jointly with J. Kadin and P. Rohatgi and have appeared in [CK90c,CR90a,CR90b,CKR91].

Typically, the reduction is required to behave correctly for a majority of the random trials, but sometimes the probability is as low as inverse polynomial. In this chapter, we examine randomized reductions under a new light. We find that in many situations there is a right definition — especially when one considers *completeness* under randomized reductions. Intuitively, when the probability of a correct reduction taking place is too low, even trivial sets can be complete under randomized reductions. Conversely, if the probability of correctness is required to be very high, then randomized reductions behave like many-one reductions. Hence, the complete languages under this kind of randomized reductions would have a complexity that is representative of the complexity of the entire class. (The meaning of trivial and representative will be made clear.) Supposedly, at some exact point, the probability of correctness is just high enough to make the definition right. We call this point the *threshold*. It turns out that this threshold is different for different complexity classes. In this chapter, we give tight upper and lower bounds on the thresholds for NP, co-NP, $\mathrm{D^P}$ and co-$\mathrm{D^P}$. These results are a consequence of the new theorems about the complexity of USAT listed above.

In the following sections, we give an historical review of the role of randomized reductions in the complexity of the class $\mathrm{D^P}$ and the Unique Satisfiability problem. Then, we show that the often quoted statement "USAT is complete for $\mathrm{D^P}$ under randomized reductions" can give misleading results about the complexity of optimization problems, because under the same type of randomized reductions, USAT is complete for $\mathrm{P}^{\mathrm{SAT}[\log n]}$ as well. Next, we show that thresholds are natural concepts for the classes NP and co-NP. The probability thresholds for NP and co-NP can be identified by some simple observations. Finally, we go on to prove that the threshold probability is $1/pol$ for $\mathrm{D^P}$ and $1/2 + 1/pol$ for co-$\mathrm{D^P}$.

6.1 An Historical Account

From the beginning, the study of the complexity of unique satisfiability has been tied to the class $\mathrm{D^P}$ and to randomized reductions. Papadimitriou and Yannakakis [PY84] first defined $\mathrm{D^P}$ to study the complexity of the facets of polytopes and the complexity of various optimization problems.

The set of uniquely satisfiable Boolean formulas, USAT, is contained in $\mathrm{D^P}$. So, the natural question to ask is: Can USAT be complete for $\mathrm{D^P}$? Blass and Gurevich

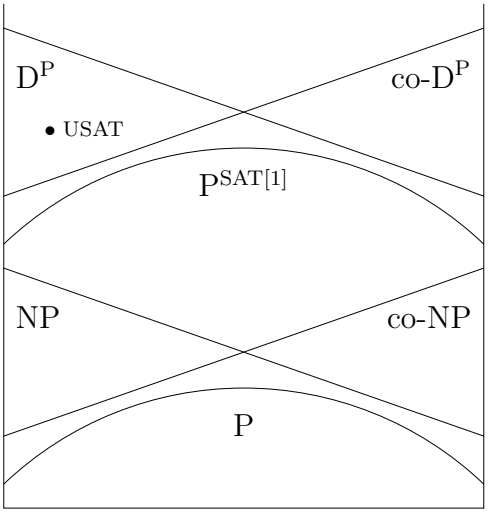

Figure 6.1: USAT and related complexity classes.

[BG82] partially answered this question. They noticed that

$$\text{USAT is } \leq^{\mathrm{P}}_{\mathrm{m}}\text{-complete for } \mathrm{D}^{\mathrm{P}}$$
$$\iff \mathrm{SAT} \wedge \overline{\mathrm{SAT}} \leq^{\mathrm{P}}_{\mathrm{m}} \mathrm{USAT} \iff \mathrm{SAT} \leq^{\mathrm{P}}_{\mathrm{m}} \mathrm{USAT}.$$

So, the question of whether USAT can be $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for D^{P} hinges on whether there is a $\leq^{\mathrm{P}}_{\mathrm{m}}$-reduction from SAT to USAT. Then, they showed that there are oracle worlds where no such reduction can exist. This meant a non-relativizing proof technique would be needed to answer the question—a formidable obstacle, indeed.

Valiant and Vazirani [VV86] did not surmount this obstacle, but they did manage to circumvent it. They were able to construct a *randomized reduction* from SAT to USAT. More precisely, they constructed a polynomial time function f such that

$$x \in \mathrm{SAT} \Longrightarrow \mathrm{Prob}_z[\ f(x,z) \in \mathrm{USAT}\ ] \geq \frac{1}{4|x|}$$
$$x \notin \mathrm{SAT} \Longrightarrow \mathrm{Prob}_z[\ f(x,z) \notin \mathrm{USAT}\ ] = 1.$$

Thus, USAT becomes complete for D^{P} under randomized reductions[2]. However, this variety of randomized reduction is not quite satisfying, because the probability

[2]Valiant and Vazirani credit Alan Selman for this application of their randomized reduction.

of the reduction being correct can approach zero as the length of x increases. One would have expected a probability bound of 1/2 (in keeping with the Adleman-Manders [AM77] definition). The justification for the Valiant-Vazirani definition is that in many situations the probability bound can be amplified, in which case, the definitions would be equivalent. Before we continue, we need to introduce some notation and terminology to facilitate our discussion of randomized reductions with different probability bounds.

Definition: We say that A randomly reduces to B (written $A \leq_{\mathrm{m}}^{\mathrm{rp}} B$) with probability δ, if there exists a polynomial time function f and a polynomial bound q such that

$$x \in A \Longrightarrow \mathrm{Prob}_z[\ f(x,z) \in B\ ] \geq \delta$$
$$x \notin A \Longrightarrow \mathrm{Prob}_z[\ f(x,z) \notin B\ ] = 1,$$

where z is chosen uniformly over $\{0,1\}^{q(n)}$.

Using this terminology, Valiant and Vazirani showed that SAT randomly reduces to USAT with probability $1/(4n)$. As a special case, we will write $A \leq_{\mathrm{m}}^{\mathrm{vv}} B$ if $A \leq_{\mathrm{m}}^{\mathrm{rp}} B$ with probability $1/p(n)$ for some polynomial bound p. We will reserve the term "the Valiant-Vazirani reduction" to name the randomized reduction from SAT to USAT. Similarly, the Adleman and Manders definition of randomized reductions would be randomized reductions with probability 1/2. Also, in statements where the exact probability bound is not important, we will use the terms $1/pol$ and $1/exp$ to indicate that the statement holds for any inverse polynomial and inverse exponential function. As we mentioned before, under certain conditions, $\leq_{\mathrm{m}}^{\mathrm{vv}}$-reductions and randomized reductions with probability 1/2 are equivalent. (The proof is standard, but we include it for completeness.)

Lemma 6.1. If $A \leq_{\mathrm{m}}^{\mathrm{vv}} B$ and B has OR_ω, then for all polynomials $p'(n)$, there exist a polynomial time function f' and a polynomial $q'(n)$ such that

$$x \in A \Longrightarrow \mathrm{Prob}_z[\ f'(x,z) \in B] \geq 1 - 2^{-p'(n)}$$
$$x \notin A \Longrightarrow \mathrm{Prob}_z[\ f'(x,z) \notin B] = 1,$$

where $n = |x|$ and z is chosen uniformly over $\{0,1\}^{q'(n)}$.

Proof: Let f be the $\leq_{\mathrm{m}}^{\mathrm{rp}}$-reduction from A to B. That is, there are polynomials

p and q such that

$$x \in A \Longrightarrow \mathrm{Prob}_z[\ f(x,z) \in B] \geq 1/p(n)$$
$$x \notin A \Longrightarrow \mathrm{Prob}_z[\ f(x,z) \notin B] = 1,$$

where $n = |x|$ and z is chosen uniformly over $\{0,1\}^{q(n)}$. Consider the set

$$\mathrm{OR}_\omega(B) = \{\ \langle x_1, \ldots, x_m \rangle \mid \exists i,\ 1 \leq i \leq m,\ x_i \in B\ \}.$$

By assumption, $\mathrm{OR}_\omega(B) \leq^{\mathrm{P}}_{\mathrm{m}} B$ via some polynomial time function g. Now, define $r(n) = p(n) \cdot p'(n)$ and $q'(n) = r(n) \cdot q(n)$. We construct the new $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reduction f' as follows. On input (x,z), where $z \in \{0,1\}^{q'(n)}$ and $n = |x|$, f' divides z into $r(n)$ segments $z_1, \ldots, z_{r(n)}$ of length $q(n)$. Let

$$f'(x,z) = g\langle f(x,z_1), f(x,z_2), \ldots, f(x,z_{r(n)})\rangle,$$

where g is the $\leq^{\mathrm{P}}_{\mathrm{m}}$-reduction from $\mathrm{OR}_\omega(B)$ to B. Clearly, if $x \notin A$ then for all i, $f(x,z_i) \notin B$. So,

$$x \notin A \Longrightarrow \forall z \in \{0,1\}^{q'(|x|)},\ f'(x,z) \notin B \Longrightarrow \mathrm{Prob}_z[\ f(x,z) \notin B\ ] = 1.$$

Suppose $x \in A$, then $\mathrm{Prob}_z[\ f(x,z) \in B\ ] \geq 1/p(n)$. So, the probability that for some i, $1 \leq i \leq r(n)$, $f(x,z_i) \in B$ is bounded below by:

$$\begin{aligned} &1 - \mathrm{Prob}_{z_1, \ldots, z_{r(n)}}[\ \forall i,\ 1 \leq i \leq r(n),\ f(x,z_i) \notin B\ ] \\ &\quad \geq 1 - (1 - 1/p(n))^{r(n)} \\ &\quad \geq 1 - ((1 - 1/p(n))^{p(n)})^{p'(n)}, \end{aligned}$$

Since $(1 - 1/p(n))^{p(n)}$ converges to e^{-1}, we have

$$x \in A \Longrightarrow \mathrm{Prob}_z[\ f'(x,z) \in B\ ] \geq 1 - e^{-p'(n)} \geq 1 - 2^{-p'(n)},$$

where z is chosen uniformly over $\{0,1\}^{q'(n)}$. □

Robust languages such as SAT and $\overline{\mathrm{SAT}}$ have both OR_ω and AND_ω. So, in cases where one randomly reduces to a robust language, it does not matter which definition of randomized reduction is used. However, there are some good reasons to believe that USAT does not have OR_ω (see Theorem 6.9). Thus, there is no obvious way to amplify the Valiant-Vazirani reduction from SAT to USAT. In the next section, we investigate some anomalies which are consequences of the non-robustness of USAT.

6.2 Anomalous Behavior

The first and most obvious problem with the statement "USAT is complete for D^{P} under randomized reductions" is that it fails to consider that USAT can be complete for larger classes as well. In fact, a simple observation will show that USAT is $\leq_{\mathrm{m}}^{\mathrm{vv}}$-complete for a much larger class, $\mathrm{P}^{\mathrm{SAT}[\log n]}$.

Lemma 6.2. USAT is $\leq_{\mathrm{m}}^{\mathrm{vv}}$-complete for $\mathrm{P}^{\mathrm{SAT}[\log n]}$.

Proof: First, observe that there is a trivial $\leq_{\mathrm{m}}^{\mathrm{vv}}$-reduction from $\mathrm{OR}_{\omega}(\mathrm{SAT} \wedge \overline{\mathrm{SAT}})$ to $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$. On input $\langle x_1, \ldots, x_n \rangle$, the reduction chooses $1 \leq i \leq n$ at random and prints out x_i. This randomized reduction will succeed with probability $1/n$. Now, using the fact that $\mathrm{OR}_{\omega}(\mathrm{SAT} \wedge \overline{\mathrm{SAT}})$ is $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for $\mathrm{P}^{\mathrm{SAT}[\log n]}$, we can reduce any language in $\mathrm{P}^{\mathrm{SAT}[\log n]}$ to USAT by combining the reduction described above with the Valiant-Vazirani reduction. This new reduction has probability $1/(4n^2)$. □

In Section 3.3, we showed that the complete languages for D^{P} and $\mathrm{P}^{\mathrm{SAT}[\log n]}$ have different characterizations. Since we expect complete sets to inherit structural properties of the classes they represent, USAT being $\leq_{\mathrm{m}}^{\mathrm{vv}}$-complete for both these classes raises doubts as to whether completeness under $\leq_{\mathrm{m}}^{\mathrm{vv}}$ reductions makes sense. Lemma 6.2 can also give misleading results about the complexity of certain optimization problems. For instance, it implies that all the Unique Optimization problems in $\mathrm{P}^{\mathrm{SAT}[\log n]}$ can be reduced to Unique Satisfiability. At first glance, this appears to highlights the power of randomization. After all, it shows that using randomization one can solve optimization problems without doing optimization. However, the probability of a correct reduction taking place is only $1/(4n^2)$. Moreover, there is no known way to improve this probability bound (see Theorem 6.9). In fact, this lemma really demonstrates the anomalies created by randomized reductions that allow very low probability bounds.

One would think that these anomalies would disappear if we used only the Adleman-Manders definition of randomized reductions. However, in the next section, we will show that even if we restrict our attention to $\leq_{\mathrm{m}}^{\mathrm{rp}}$-reductions with probability $1/2$, anomalies still exist. First, we must explain what *threshold behavior* means.

6.3 Threshold Behavior

Considering the anomalous behavior of randomized reductions described above, the reader may be tempted to dismiss completeness under randomized reductions as meaningless. However, the Valiant-Vazirani reduction from SAT to USAT proved to be useful in many areas of research. For example, Beigel [Bei88] used it to show that SAT is superterse unless RP = NP. Also, Toda [Tod89] used a similar reduction in his proof that PH $\subseteq$ $\mathrm{P}^{\#\mathrm{P}[1]}$. This result, in turn, led to the Lund, Fortnow, Karloff and Nisan [LFKN90] result: PH $\subseteq$ IP. So, there should be little doubt in the reader's mind regarding the usefulness of the Valiant-Vazirani reduction. The more pertinent questions are: What does this reduction mean? Does USAT being $\leq_{\mathrm{m}}^{\mathrm{vv}}$-complete for D^{P} mean that it somehow inherits the hardness of the whole class? How does the complexity of $\leq_{\mathrm{m}}^{\mathrm{vv}}$-complete sets compare with the $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete languages? We answer these questions in terms of *thresholds* of probability bounds.

6.4 Examples of Threshold Behavior

To illustrate what we mean by threshold behavior, we turn to a more familiar setting—namely that of NP and co-NP. Let A be any $\leq_{\mathrm{m}}^{\mathrm{vv}}$-complete set for NP. Then, we know that A has the following properties [AM77,ZH86,BHZ87,Sch89].

- $A \notin$ P, unless RP = NP and PH collapses.
- $A \notin$ RP, unless RP = NP and PH collapses.
- $A \notin$ co-NP, unless NP/*poly* = co-NP/*poly* and PH collapses.

This compares favorably with the $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete set SAT, which has the following properties.

- SAT $\notin$ P, unless P = NP.
- SAT $\notin$ RP, unless RP = NP and PH collapses.
- SAT $\notin$ co-NP, unless NP = co-NP.

From this comparison, we can draw the conclusion that the $\leq_{\mathrm{m}}^{\mathrm{vv}}$-complete set, A, behaves like the $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete set, SAT, and that somehow the complexity of the set A is representative of the complexity of the entire class NP.

In contrast, the trivial singleton set $B = \{1\}$ is complete for NP under randomized reductions with probability 2^{-n}. Clearly, B is not representative of the complexity of NP. So, completeness under $\leq_{\mathrm{m}}^{\mathrm{rp}}$-reductions with probability 2^{-n} does not make sense for NP. However, they *do* make sense for co-NP. Let C be a set complete for co-NP under $\leq_{\mathrm{m}}^{\mathrm{rp}}$-reductions with probability 2^{-n}. Then, $C \notin$ NP, unless NP = co-NP.

These observations show that completeness for randomized reductions makes sense only if the randomized reductions have a probability bound above a certain threshold. Moreover, the threshold is different for different complexity classes. For NP, the threshold is $1/pol$; for co-NP, $1/exp$. Alternatively, we can look at these results in terms of randomized reductions from a set to its complement.

- SAT $\leq_{\mathrm{m}}^{\mathrm{rp}} \overline{\mathrm{SAT}}$ with probability 2^{-n}.
- SAT $\not\leq_{\mathrm{m}}^{\mathrm{rp}} \overline{\mathrm{SAT}}$ with probability $1/p(n)$, for any polynomial $p(n)$,
  unless NP/*poly* = co-NP/*poly* and PH collapses.

These two statements show that when we consider randomized functions which reduce SAT to $\overline{\mathrm{SAT}}$, a probability threshold occurs at $1/pol$. Similarly, for $\overline{\mathrm{SAT}}$, the probability threshold occurs at $1/exp$.

- There is no known $\leq_{\mathrm{m}}^{\mathrm{rp}}$-reduction from $\overline{\mathrm{SAT}}$ to SAT with probability > 0.
- $\overline{\mathrm{SAT}} \not\leq_{\mathrm{m}}^{\mathrm{rp}}$ SAT with probability $2^{-p(n)}$, for any polynomial $p(n)$,
  unless NP = co-NP.

6.5 Threshold Behavior in D^{p} and co-D^{p}

In this section, we show how probability thresholds can explain the anomaly presented in Section 6.2. We will show that beyond a certain probability threshold, the $\leq_{\mathrm{m}}^{\mathrm{rp}}$-complete sets for D^{P} and for co-D^{P} inherit many of the properties of the $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete sets. Recall from Chapters 3 and 4 that some of these properties are:

- SAT$\wedge\overline{\text{SAT}}$ and $\overline{\text{SAT}}\vee$SAT are not equivalent under $\leq^{\text{P}}_{\text{m}}$-reductions, unless $\text{D}^{\text{P}} = \text{co-D}^{\text{P}}$ and PH collapses.
- SAT$\wedge\overline{\text{SAT}} \notin \text{co-D}^{\text{P}}$ and $\overline{\text{SAT}}\vee\text{SAT} \notin \text{D}^{\text{P}}$, unless $\text{D}^{\text{P}} = \text{co-D}^{\text{P}}$ and PH collapses.
- SAT$\wedge\overline{\text{SAT}}$ does not have OR_2, unless $\text{D}^{\text{P}} = \text{co-D}^{\text{P}}$ and PH collapses.
- $\overline{\text{SAT}}\vee$SAT does not have AND_2, unless $\text{D}^{\text{P}} = \text{co-D}^{\text{P}}$ and PH collapses.

We will use these properties as a benchmark to test if a particular concept of completeness for D^{P} and co-D^{P} makes sense. First, we show that for completeness under $\leq^{\text{rp}}_{\text{m}}$-reductions, the probability threshold for D^{P} is bounded below by 2^{-n}.

Lemma 6.3.
$\overline{\text{SAT}}$ is complete for D^{P} under $\leq^{\text{rp}}_{\text{m}}$-reductions with probability 2^{-n}.

Proof: To reduce $(F_1, F_2) \in \text{SAT}\wedge\overline{\text{SAT}}$ to $\overline{\text{SAT}}$, the reduction guesses a satisfying assignment for F_1. If F_1 is satisfiable, the guess will succeed with probability at least 2^{-n}. If a satisfying assignment is found, the reduction simply prints out F_2. Otherwise, it prints out a fixed satisfiable formula. □

Corollary 6.4. $\text{SAT}\wedge\overline{\text{SAT}} \leq^{\text{rp}}_{\text{m}} \overline{\text{SAT}}\vee\text{SAT}$ with probability 2^{-n}.

Since $\overline{\text{SAT}} \in \text{co-D}^{\text{P}}$ and since $\overline{\text{SAT}}$ has OR_ω and AND_ω, we can safely say that completeness under $\leq^{\text{rp}}_{\text{m}}$-reductions with probability $1/\mathit{exp}$ does not make sense for D^{P}. The following theorems, show that completeness under $\leq^{\text{rp}}_{\text{m}}$-reductions starts making sense when the probability bound is $1/\mathit{pol}$. Hence, the probability threshold for D^{P} is bounded above by $1/\mathit{pol}$.

Theorem 6.5.
$\text{SAT}\wedge\overline{\text{SAT}} \not\leq^{\text{vv}}_{\text{m}} \overline{\text{SAT}}\vee\text{SAT}$, unless $\text{NP}/\mathit{poly} = \text{co-NP}/\mathit{poly}$ and $\text{PH} \subseteq \Delta^{\text{P}}_3$.

Proof: Before we go on, we need to define some probabilistic classes.

Definition: For any class $\mathcal{C}$, $A \in \text{BP}\cdot\mathcal{C}$ if there exists a language $B \in \mathcal{C}$ and polynomials p and q such that

$$\forall x,\ \text{Prob}_y[x \in A \iff (x, y) \in B] > \frac{1}{2} + \frac{1}{p(n)},$$

where y is chosen uniformly from $\{0,1\}^{q(n)}$.

The hard/easy argument used the following line of reasoning. Suppose $\mathrm{D^P} = \text{co-}\mathrm{D^P}$, then there is a many-one reduction from $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ to $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$. With the help of an advice function f, this reduction can be converted into a reduction from $\overline{\mathrm{SAT}}$ to SAT. Thus, $\overline{\mathrm{SAT}} \in \mathrm{NP}/f$. Then, by Yap's theorem [Yap83], PH collapses to Σ_3^{P}. In the next theorem, we will show that the $\leq_{\mathrm{m}}^{\mathrm{rp}}$-reduction from $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ to $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$ can be converted into an $\leq_{\mathrm{m}}^{\mathrm{rp}}$-reduction from $\overline{\mathrm{SAT}}$ to SAT (with help from an advice function). It follows that $\overline{\mathrm{SAT}}$ is contained in the rather awkward class $\mathrm{BP}\cdot(\mathrm{NP}/f)$.

Now, suppose there exists a $\leq_{\mathrm{m}}^{\mathrm{vv}}$-reduction from $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ to $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$. Since $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$ has OR_ω, the probability bound of this $\leq_{\mathrm{m}}^{\mathrm{vv}}$-reduction can be amplified[3] by Lemma 6.1. That is, using the reduction from $\mathrm{OR}_\omega(\overline{\mathrm{SAT}} \vee \mathrm{SAT})$ to $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$, we can combine the results of polynomially many $\leq_{\mathrm{m}}^{\mathrm{vv}}$-reductions and construct a polynomial time function h and a polynomial q such that

$$(F_1, F_2) \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}} \implies \mathrm{Prob}_z[h(F_1, F_2, z) \in \overline{\mathrm{SAT}} \vee \mathrm{SAT}] \geq 1 - 2^{-n}$$

$$(F_1, F_2) \in \overline{\mathrm{SAT}} \vee \mathrm{SAT} \implies \mathrm{Prob}_z[h(F_1, F_2, z) \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}}] = 1$$

where $n = |(F_1, F_2)|$ and z is chosen uniformly over $\{0,1\}^{q(n)}$.

To prove the theorem, we will construct an advice function f computable in Δ_3^{P} such that $\overline{\mathrm{SAT}} \in \mathrm{BP}\cdot(\mathrm{NP}/f)$. Then, using the techniques in Schöning's proof (q.v. [Sch89] Corollary 3.6) that $\mathrm{BP}\cdot(\mathrm{NP}/f) \subseteq \mathrm{NP}/\mathit{poly}$ and the fact that f is computable in Δ_3^{P}, we show that $\mathrm{PH} \subseteq \Delta_3^{\mathrm{P}}$.

Now, we construct the advice function f. In this hard/easy argument, we call F *easy* if $F \in \overline{\mathrm{SAT}}$ and

$$\exists x, y \text{ such that } |x| = |F|,\ y \in \{0,1\}^{q(n)},\ \pi_2(h(x, F, y)) \in \mathrm{SAT},$$

As usual, if $F \in \overline{\mathrm{SAT}}$ and F is not easy, then we call F a *hard* string. On input 1^n, the advice function simply outputs the lexically smallest hard string of length n if it exists. Otherwise, it outputs the empty string, ε. From the definition of easy, it is clear that checking whether a string is hard is a co-NP question. So, using binary search and an $\mathrm{NP}^{\mathrm{NP}}$ oracle, the advice function f can be computed in Δ_3^{P}.

[3] We can prove this theorem without using amplification, but this simplifies the presentation. In the general case, one cannot rely on amplification.

Now, consider the following NP program, N. On input (F, H, z), N treats F as a Boolean formula of length n, takes H as an advice string of length 0 or n, and parses z into a $q(n)$-bit long guess string. (If the input does not conform to this syntax, N rejects outright.) Then, N does the following.

1. If the advice string H is the empty string, accept if and only if there exists x, $|x| = n$ and $y \in \{0,1\}^{q(n)}$ such that $\pi_2(h(x, F, y)) \in \mathrm{SAT}$.

2. If $|H| = n$, accept if and only if $\pi_1(h(F, H, z)) \in \mathrm{SAT}$.

Claim: The NP program above shows that $\overline{\mathrm{SAT}} \in \mathrm{BP}\cdot(\mathrm{NP}/f)$. That is,

$$\mathrm{Prob}_z[\ F \in \overline{\mathrm{SAT}} \iff N(F, f(1^n), z) \text{ accepts }] \geq 1 - 2^{-\mathrm{n}}.$$

Note that whether there is a hard string of length n does not depend on the guess string z. So, we can analyze the program in two cases.

Case 1: Consider the case where all the strings in $\overline{\mathrm{SAT}}$ of length n are easy—i.e., $f(1^n) = \varepsilon$. If the input $F \in \overline{\mathrm{SAT}}$, then F must also be easy which means the appropriate x and y would be found in step 1. So,

$$F \in \overline{\mathrm{SAT}} \Longrightarrow \mathrm{Prob}_z[\ N(F, \varepsilon, z) \text{ accepts }] = 1.$$

If $F \in \mathrm{SAT}$, then for all x, $(x, F) \in \overline{\mathrm{SAT}} \vee \mathrm{SAT}$. So, by the description of the reduction h,

$$\mathrm{Prob}_z[\ h(x, F, z) \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}}\ ] = 1.$$

However, $h(x, F, z) \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ implies that $\pi_2(h(x, F, z)) \in \overline{\mathrm{SAT}}$. So, N must reject in step 1. Thus, in the easy case

$$\mathrm{Prob}_z[\ F \in \overline{\mathrm{SAT}} \iff N(F, \varepsilon, z) \text{ accepts }] = 1.$$

Case 2: Suppose the advice string H is a hard string of length n. If $F \in \overline{\mathrm{SAT}}$, then $(F, H) \in \overline{\mathrm{SAT}} \vee \mathrm{SAT}$. By the description of the reduction h,

$$\mathrm{Prob}_z[\ h(F, H, z) \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}}\ ] = 1.$$

So, for all z, $\pi_1(h(F, H, z)) \in \mathrm{SAT}$. Thus, for all z, N will accept in step 2 and

$$F \in \overline{\mathrm{SAT}} \Longrightarrow \mathrm{Prob}_z[\ N(F, H, z) \text{ accepts }] = 1.$$

If $F \in \mathrm{SAT}$, then $(F, H) \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ because H is hard implies that $H \in \overline{\mathrm{SAT}}$. Also, since H is hard,

$$\forall x, |x| = n,\ \forall z, z \in \{0,1\}^{q(n)},\ \pi_2(h(x, H, z)) \in \overline{\mathrm{SAT}}.$$

Therefore, for all choices of z, we know that

$$h(F, H, z) \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}} \iff \pi_1(h(F, H, z)) \in \mathrm{SAT}$$

Moreover, by the description of h and the fact that $(F, H) \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}}$,

$$\mathrm{Prob}_z[\ h(F, H, z) \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}}\ ] \leq 2^{-n}.$$

So, $\mathrm{Prob}_z[\ \pi_1(h(F, H, z)) \in \mathrm{SAT}\ ] \leq 2^{-n}$. Thus,

$$F \in \mathrm{SAT} \Longrightarrow \mathrm{Prob}_z[\ N(F, H, z) \text{ accepts }] < 2^{-n}.$$

□

An immediate corollary of Theorem 6.5 is that completeness under $\leq^{\mathrm{vv}}_{\mathrm{m}}$-reductions *does* make sense for D^{P}. Moreover, we can show that languages complete for D^{P} under $\leq^{\mathrm{vv}}_{\mathrm{m}}$-reductions have properties very similar to the properties of the $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete language $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$.

Theorem 6.6. Let A be complete for D^{P} under $\leq^{\mathrm{vv}}_{\mathrm{m}}$-reductions. Then,

1. $A \not\leq^{\mathrm{P}}_{\mathrm{m}} \overline{A}$, unless $\mathrm{NP}/\mathit{poly} = \text{co-NP}/\mathit{poly}$ and $\mathrm{PH} \subseteq \Delta^{\mathrm{P}}_3$.
2. $A \notin \text{co-D}^{\mathrm{P}}$, unless $\mathrm{NP}/\mathit{poly} = \text{co-NP}/\mathit{poly}$ and $\mathrm{PH} \subseteq \Delta^{\mathrm{P}}_3$.
3. A does not have OR_ω, unless $\mathrm{NP}/\mathit{poly} = \text{co-NP}/\mathit{poly}$ and $\mathrm{PH} \subseteq \Sigma^{\mathrm{P}}_3$.

Proof: Parts 1 and 2 follow from Theorem 6.5. Part 3 requires a short proof. First, note that if A has OR_ω, then $\overline{\mathrm{SAT}} \vee \mathrm{SAT} \leq^{\mathrm{vv}}_{\mathrm{m}} A$. Now, since A has OR_ω, this reduction can be amplified using Lemma 6.1. So, there is a randomized reduction from $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$ to A with very high probability. However, $A \in \mathrm{D}^{\mathrm{P}}$, so $A \leq^{\mathrm{P}}_{\mathrm{m}} \mathrm{SAT} \wedge \overline{\mathrm{SAT}}$. Thus, there is also a randomized reduction from $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$ to $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ with very high probability. That is, there exist a polynomial time function h and a polynomial q such that

$$(F_1, F_2) \in \overline{\mathrm{SAT}} \vee \mathrm{SAT} \implies \mathrm{Prob}_z[\ h(F_1, F_2, z) \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}}\ ] \geq 1 - 2^{-n^2}$$

$$(F_1, F_2) \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}} \implies \mathrm{Prob}_z[\ h(F_1, F_2, z) \in \overline{\mathrm{SAT}} \vee \mathrm{SAT}\ ] = 1$$

where $n = |(F_1,F_2)|$ and z is chosen uniformly over $\{0,1\}^{q(n)}$. We will show in Theorem 6.12 that these conditions NP/*poly* = co-NP/*poly* and hence PH $\subseteq \Sigma_3^{\mathrm{P}}$. □

As a special case of a language $\leq_{\mathrm{m}}^{\mathrm{vv}}$-complete for D^{P}, USAT has all the properties listed above. However, since $\overline{\mathrm{SAT}} \leq_{\mathrm{m}}^{\mathrm{P}} \mathrm{USAT}$ the results can be made stronger. We list these properties separately, because these results give a new understanding about the complexity of USAT.

Corollary 6.7. USAT $\notin$ co-D^{P}, unless PH collapses.

Theorem 6.8. $\mathrm{USAT} \not\equiv_{\mathrm{m}}^{\mathrm{P}} \overline{\mathrm{USAT}}$, unless $\mathrm{D}^{\mathrm{P}} = \text{co-}\mathrm{D}^{\mathrm{P}}$ and PH collapses.

Proof: Since $\overline{\mathrm{SAT}} \leq_{\mathrm{m}}^{\mathrm{P}} \mathrm{USAT}$, we know that $\mathrm{SAT} \leq_{\mathrm{m}}^{\mathrm{P}} \overline{\mathrm{USAT}}$. However, we assumed that $\mathrm{USAT} \equiv_{\mathrm{m}}^{\mathrm{P}} \overline{\mathrm{USAT}}$, so SAT $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduces to USAT as well. Thus, USAT becomes $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for D^{P}, so $\mathrm{D}^{\mathrm{P}} = \text{co-}\mathrm{D}^{\mathrm{P}}$ and PH collapses. □

Corollary 6.9. USAT does not have OR_ω, unless PH collapses.

So, we can conclude that Valiant and Vazirani made the right decision when they used $\leq_{\mathrm{m}}^{\mathrm{vv}}$-reductions to talk about completeness for D^{P} under randomized reductions. However, completeness under $\leq_{\mathrm{m}}^{\mathrm{vv}}$-reductions may not make sense for other complexity classes. In fact, the following theorems show that the threshold for co-D^{P} is $1/2 + 1/pol$.

Lemma 6.10. $\mathrm{SAT} \oplus \overline{\mathrm{SAT}}$ is complete for co-D^{P} under $\leq_{\mathrm{m}}^{\mathrm{rp}}$-reductions with probability $1/2 + 2^{-n^2}$.

Proof (Sketch): Recall that $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$ is $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for co-$\mathrm{D}^{\mathrm{P}}$and that

$$\mathrm{SAT} \oplus \overline{\mathrm{SAT}} = \{\ 0F \mid F \in \mathrm{SAT}\ \} \cup \{\ 1F \mid F \in \overline{\mathrm{SAT}}\ \}.$$

It is simple to construct a randomized reduction with probability greater than or equal to 1/2, because

$$(F_1, F_2) \in \overline{\mathrm{SAT}} \vee \mathrm{SAT} \iff F_1 \in \overline{\mathrm{SAT}} \text{ or } F_2 \in \mathrm{SAT}.$$

Thus, a randomized function can choose F_1 or F_2 with equal probability, then output $1F_1$ or $0F_2$. If (F_1, F_2) is indeed an element of $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$, then

$$\mathrm{Prob}_{i \in \{0,1\}}[\ iF_{2-i} \in \mathrm{SAT} \oplus \overline{\mathrm{SAT}}\ ] \geq 1/2.$$

Conversely, if $(F_1, F_2) \notin \overline{\mathrm{SAT}} \vee \mathrm{SAT}$, then $\mathrm{Prob}_{i \in \{0,1\}}[\ iF_{2-i} \in \mathrm{SAT} \oplus \overline{\mathrm{SAT}}\ ] = 0$.

To improve the probability beyond 1/2, simply observe that if $F_2 \in \mathrm{SAT}$, then there is a small probability of guessing a satisfying assignment. This fact can be used to improve the probability to $1/2 + 2^{-n^2}$. (This additional 2^{-n^2} probability is of interest only because it makes the probability *strictly* greater than 1/2.) □

Corollary 6.11. $\overline{\mathrm{SAT}} \vee \mathrm{SAT} \leq_{\mathrm{m}}^{\mathrm{rp}} \mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ with probability $1/2 + 2^{-n^2}$.

Since $\mathrm{SAT} \oplus \overline{\mathrm{SAT}} \in \mathrm{D^P}$, we again conclude that this set does not meet our criteria for a sensible complete language. However, if we require the reduction to have a probability bound above $1/2 + 1/pol$, then completeness makes sense.

Theorem 6.12. $\overline{\mathrm{SAT}} \vee \mathrm{SAT} \not\leq_{\mathrm{m}}^{\mathrm{rp}} \mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ with probability $1/2 + 1/p(n)$, for any polynomial $p(n)$, unless $\mathrm{NP}/poly = \text{co-NP}/poly$ and $\mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}$.

Proof: Suppose that $\overline{\mathrm{SAT}} \vee \mathrm{SAT} \leq_{\mathrm{m}}^{\mathrm{rp}} \mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ with probability $1/2 + 1/p(n)$. Then, there exists a reduction h and polynomials p and q such that

$$(F_1, F_2) \in \overline{\mathrm{SAT}} \vee \mathrm{SAT} \Longrightarrow \mathrm{Prob}_z[\, h(F_1, F_2, z) \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}}\,] \geq \frac{1}{2} + \frac{1}{p(n)}$$
$$(F_1, F_2) \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}} \Longrightarrow \mathrm{Prob}_z[\ h(F_1, F_2, z) \in \overline{\mathrm{SAT}} \vee \mathrm{SAT}\ ] = 1,$$

where $n = |(F_1, F_2)|$ and z is chosen uniformly over $\{0,1\}^{q(n)}$.

Again, we use a variation of the hard/easy argument to prove this theorem. We call a string F *easy* if $F \in \overline{\mathrm{SAT}}$ and $\exists x$, $|x| = |F|$ such that

$$\mathrm{Prob}_z(\ \pi_2(h(x, F, z)) \in \mathrm{SAT}\ ) \geq \frac{1}{2}.$$

We call F a *hard* string if $F \in \overline{\mathrm{SAT}}$ and F is not *easy*. We construct an advice function f which on input 1^n outputs the lexically smallest hard string of length n, if it exists. Thus, on input F our advice string can either be the empty string ε (which means that there is no hard string of length $|F|$) or some string H of length $|F|$.

Now, we construct an NP machine N to show that $\overline{\mathrm{SAT}}$ can be recognized by a $\exists \cdot \mathrm{BP} \cdot (\mathrm{NP}/f)$ machine. The machine N take inputs of the form (F, H, y, z). Here, F is the formula whose unsatisfiability is in question. The string H is given by the non-uniform advice function. The string y is chosen by the existential quantifier. Finally, z is chosen randomly by the BP quantifier.

If $H = \varepsilon$, then N accepts iff $\pi_2(h(y, F, z)) \in \text{SAT}$. Otherwise, if the input is of the form (F, H, y, z), where $H \neq \varepsilon$, then N accepts iff $\pi_1(h(F, H, z) \in \text{SAT}$. Note that in the second case, the output of the machine N is independent of y. As usual there are two cases to analyze, depending on the advice string H.

Case 1: The advice string is empty.

Since $H = f(1^{|F|}) = \varepsilon$, we know that all strings of size $|F|$ which are in $\overline{\text{SAT}}$ are *easy*. Thus, if $F \in \overline{\text{SAT}}$ then F is easy. Therefore, $\exists y,\ |y| = |F|$, such that

$$\text{Prob}_z[\ \pi_2(h(y, F, z)) \in \text{SAT}\ ] \geq \frac{1}{2}.$$

Thus, $\exists y,\ |y| = |F|,\ \text{Prob}_z[\ N(F, \varepsilon, y, z) \text{ accepts }\ ] \geq \frac{1}{2}$. On the other hand, if $F \in \text{SAT}$, then $\forall y, (y, F) \in \overline{\text{SAT}} \vee \text{SAT}$. Thus, by the random reduction, $h(y, F, z) \in \text{SAT} \wedge \overline{\text{SAT}}$ with probability $1/2 + 1/p(n)$. So,

$$\forall y,\ |y| = |F|,\ \text{Prob}_z[\ \pi_2(h(y, F, z)) \in \overline{\text{SAT}}\ ] \geq \frac{1}{2} + \frac{1}{p(n)}.$$

That is, $\forall y,\ |y| = |F|$, $\text{Prob}_z[\ N(F, \varepsilon, y, z) \text{ accepts }\ ] < \frac{1}{2} - \frac{1}{p(n)}$.

Case 2: The advice is not empty.

By construction, H is a *hard* string of size n, which implies $H \in \overline{\text{SAT}}$. If $F \in \overline{\text{SAT}}$ then $(F, H) \in \overline{\text{SAT}} \vee \text{SAT}$ and by the definition of the random reduction h,

$$\text{Prob}_z[\ \pi_1(h(F, H, z)) \in \text{SAT}\ ] \geq \frac{1}{2} + \frac{1}{p(n)}.$$

For the sake of uniformity, we use a dummy quantifier and state that

$$\exists y,\ |y| = |F|,\ \text{Prob}_z[\ \pi_1(h(F, H, z)) \in \text{SAT}\ ] \geq \frac{1}{2} + \frac{1}{p(n)},$$

(since $\pi_1(h(F, H, z)) \in \text{SAT}$ is independent of y). Thus,

$$\exists y,\ |y| = |F|,\ \text{Prob}_z[\ N(F, H, y, z) \text{ accepts }\ ] \geq \frac{1}{2} + \frac{1}{p(n)}.$$

If $F \in \text{SAT}$ then $(F, H) \in \text{SAT} \wedge \overline{\text{SAT}}$, and by the definition of the reduction h we know that $\text{Prob}_z[\ h(F, H, z) \in \overline{\text{SAT}} \vee \text{SAT}\ ] = 1$. However, H is a hard string, so

$$\forall x,\ |x| = |H|,\ \text{Prob}_z[\ \pi_2(h(x, H, z)) \in \text{SAT}\ ] < \frac{1}{2}.$$

In particular, with $x = F$, $\text{Prob}_z[\ \pi_2(h(F, H, z)) \in \text{SAT}\ ] < \frac{1}{2}$. Therefore,

$$\text{Prob}_z[\ \pi_1(h(F, H, z)) \in \overline{\text{SAT}}\ ] \geq \frac{1}{2},$$

which implies $\mathrm{Prob}_z[\ \pi_1(h(F,H,z)) \in \mathrm{SAT}\ ] < \frac{1}{2}$. Again, by adding an additional dummy quantifier, we obtain

$$\forall y,\ |y| = |F|,\ \mathrm{Prob}_z[\ N(F,H,y,z) \text{ accepts }] < \frac{1}{2}.$$

To summarize, we have shown that N behaves in the following manner:
If $f(1^{|F|}) = H = \varepsilon$, then

$$F \in \overline{\mathrm{SAT}} \Longrightarrow \exists y,\ \mathrm{Prob}_z[\ N(F,H,y,z) \text{ accepts }] \geq \tfrac{1}{2}$$

$$F \in \mathrm{SAT} \Longrightarrow \forall y,\ \mathrm{Prob}_z[\ N(F,H,y,z) \text{ accepts }] \leq \tfrac{1}{2} - \tfrac{1}{p(n)}.$$

If $f(1^{|F|}) = H \neq \varepsilon$, then

$$F \in \overline{\mathrm{SAT}} \Longrightarrow \exists y,\ \mathrm{Prob}_z[\ N(F,H,y,z) \text{ accepts }] \geq \tfrac{1}{2} + \tfrac{1}{p(n)}$$

$$F \in \mathrm{SAT} \Longrightarrow \forall y,\ \mathrm{Prob}_z[\ N(F,H,y,z) \text{ accepts }] < \tfrac{1}{2},$$

where $y \in \{0,1\}^{|F|}$, $z \in \{0,1\}^{q(n)}$ and $n = |(F,F)|$.

These conditions show that $\overline{\mathrm{SAT}}$ can be recognized by a $\exists\cdot\mathrm{BP}\cdot(\mathrm{NP}/f)$ machine. In the next section we will prove some technical lemmas which show that under these circumstances it follows that co-NP $\subseteq \mathrm{BP}\cdot(\mathrm{NP}/\mathit{poly})$. Since $\mathrm{BP}\cdot(\mathrm{NP}/\mathit{poly}) \subseteq \mathrm{NP}/\mathit{poly}$, $\mathrm{NP}/\mathit{poly} = \text{co-NP}/\mathit{poly}$ and PH collapses to Σ_3^{P}. □

Now we can conclude that the languages which are complete for co-D^{P} under $\leq_{\mathrm{m}}^{\mathrm{rp}}$-reductions with probability beyond the threshold of $1/2 + 1/\mathit{pol}$ do inherit the hardness of the $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete languages for co-D^{P}.

Theorem 6.13. Let A be complete for co-D^{P} under $\leq_{\mathrm{m}}^{\mathrm{rp}}$-reductions with probability $1/2 + 1/p(n)$, for some polynomial p. Then,

1. $A \not\equiv_{\mathrm{m}}^{\mathrm{P}} \overline{A}$, unless $\mathrm{NP}/\mathit{poly} = \text{co-NP}/\mathit{poly}$ and PH $\subseteq \Sigma_3^{\mathrm{P}}$.
2. $A \notin \mathrm{D}^{\mathrm{P}}$, unless $\mathrm{NP}/\mathit{poly} = \text{co-NP}/\mathit{poly}$ and PH $\subseteq \Sigma_3^{\mathrm{P}}$.
3. A does not have AND_2, unless $\mathrm{NP}/\mathit{poly} = \text{co-NP}/\mathit{poly}$ and PH $\subseteq \Delta_3^{\mathrm{P}}$.

Proof: Parts 1 and 2 are direct corollaries of Theorem 6.12. To see Part 3, note that if A has AND_2, then $\mathrm{SAT} \wedge \overline{\mathrm{SAT}} \leq_{\mathrm{m}}^{\mathrm{rp}} A$ with probability greater than 1/2. Now, Theorem 6.5 shows that $\mathrm{NP}/\mathit{poly} = \text{co-NP}/\mathit{poly}$ and PH collapses to Δ_3^{P}. □

6.6 Merlin-Arthur-Merlin Games

A Merlin-Arthur-Merlin game begins with an existential move by Merlin, followed by a probabilistic move by Arthur and another existential move by Merlin. Languages which have Merlin-Arthur-Merlin games are exactly those recognized by $\exists\cdot\mathrm{BP}\cdot\mathrm{NP}$ machines. In this section, we prove some technical lemmas which show that under certain assumptions, NP/*poly* = co-NP/*poly* and PH collapses. These lemmas are mostly modifications to familiar theorems in the literature. However, due to the nonuniform nature of the hard/easy argument, it is not possible to cite these theorems directly. We need to show that these theorems hold even for nonuniform probabilistic bounds. The theorems and lemmas in this section are geared towards normalizing probabilistic NP machines so they fit the standard definitions. The first lemma shows that we can always reprogram a probabilistic NP machine so its probability bounds are centered at 1/2.

Lemma 6.14. Let N be any NP machine. Let A and B be two disjoint events such that for some polynomial q and rational constants α, β and r

$$x \in A \Longrightarrow \mathrm{Prob}_z[\ N(x,z) \text{ accepts }] \geq \alpha$$
$$x \in B \Longrightarrow \mathrm{Prob}_z[\ N(x,z) \text{ accepts }] \leq \beta,$$

where z is chosen uniformly over $\{0,1\}^{q(|x|)}$ and $\alpha - \beta \geq 1/r$. Then, there is an NP machine N', a polynomial q' and a constant r' such that

$$x \in A \Longrightarrow \mathrm{Prob}_z[\ N'(x,z) \text{ accepts }] \geq \frac{1}{2} + \frac{1}{r'}$$
$$x \in B \Longrightarrow \mathrm{Prob}_z[\ N'(x,z) \text{ accepts }] \leq \frac{1}{2} - \frac{1}{r'}\ ,$$

where z is chosen uniformly over $\{0,1\}^{q'(|x|)}$.

Proof: The idea of behind the proof is quite simple. Let $m = (\alpha+\beta)/2$. The probabilities α and β are centered around m. We simply have to shift these probabilities so they are centered around 1/2.

First, assume that $m > 1/2$. In this case, the machine N accepts too often, so we simply need to add more cases where the machine rejects. Let $q'(n) = q(n) + c$, where c is roughly twice the number of bits required to specify α and β in binary. The new machine N' does the following on input (x,z) where $|z| = q'(n)$ and $n = |x|$.

1. Divide z into two parts v and w of lengths $q(n)$ and c respectively.

2. Interpret w as a number between 0 and 2^c.

3. If $w/2^c < 1/(2m)$, then simulate $N(x, v)$.

4. Otherwise, reject the input string.

Analysis: Now we claim that N' accepts and rejects with the prescribed probabilities. If $x \in A$, then the probability that $N'(x, z)$ accepts is the probability that N' reaches step 3, simulates $N(x, v)$, and $N(x, v)$ accepts. Thus,

$$x \in A \Longrightarrow \text{Prob}_z[\ N'(x,z) \text{ accepts }] \geq \frac{\alpha}{2m} \geq \frac{1}{2m}\left(m + \frac{1}{2r}\right) = \frac{1}{2} + \frac{1}{4mr}\ .$$

Similarly, we can calculate the probability that $N'(x, z)$ accepts when $x \in B$.

$$x \in B \Longrightarrow \text{Prob}_z[\ N'(x,z) \text{ accepts }] \leq \frac{\beta}{2m} \leq \frac{1}{2m}\left(m - \frac{1}{2r}\right) = \frac{1}{2} - \frac{1}{4mr}\ .$$

Thus, if we let $r' = 4r$, we have satisfied the statement of the lemma (since $1/2 < m \leq 1$). Note that in the preceding calculations, we used $1/(2m)$ as the probability that $w/2^c < 1/(2m)$. There is an inherent error in this estimation, but the error can be made arbitrary small by increasing c.

Finally, we consider the case where $m < 1/2$. In this case, we simply need to increase the probability of accepting. So, $N'(x, z)$ would simulate $N(x, v)$ with probability $1/(2 - 2m)$ and accept outright in the remaining cases. A similar analysis yields:

$$\begin{aligned} x \in A \Longrightarrow \text{Prob}_z[\ N'(x,z) \text{ accepts }] &\geq 1 - \frac{1}{2-2m} + \frac{\alpha}{2-2m} \\ &= \frac{1}{2} + \frac{1}{4(1-m)r} \\ x \in B \Longrightarrow \text{Prob}_z[\ N'(x,z) \text{ accepts }] &\leq 1 - \frac{1}{2-2m} + \frac{\beta}{2-2m} \\ &= \frac{1}{2} - \frac{1}{4(1-m)r}\ . \end{aligned}$$

Again, since $1/2 < 1 - m \leq 1$, we satisfy the statement of the lemma by letting $r' = 4r$. □

Note that the preceding proof did not use the fact that α and β are constants. In fact, since r' did not depend on m, it is not even necessary for $1/\alpha$ and $1/\beta$

to be polynomially bounded. The only important point is that α and β can be represented in c bits. So, in order to generalize this lemma, we need the following definition.

Definition: A function γ is a *nice* nonuniform probability bound if there exists a polynomial d such that for all n, $0 \leq \gamma(n) \leq 1$ and $|\gamma(n)| \leq d(n)$.

Using the definition of nice probability bounds, we can restate Lemma 6.14 as follows. We will not repeat the proof for this lemma because it is a straightforward modification of the proof for Lemma 6.14.

Lemma 6.15. Let N be any NP machine and let the functions α and β be nice nonuniform probability bounds. Suppose there exist two disjoint events A and B such that for some polynomial q and r

$$x \in A \Longrightarrow \text{Prob}_z[\ N(x,z) \text{ accepts }] \geq \alpha(n)$$
$$x \in B \Longrightarrow \text{Prob}_z[\ N(x,z) \text{ accepts }] \leq \beta(n),$$

where $n = |x|$, z is chosen uniformly over $\{0,1\}^{q(n)}$ and $\alpha(n) - \beta(n) > 1/r(n)$. Then, there is an NP machine N' and polynomials q' and r' such that

$$x \in A \Longrightarrow \text{Prob}_z[\ N'(x,\alpha(n),\beta(n),z) \text{ accepts }] \geq \frac{1}{2} + \frac{1}{r'(n)}$$
$$x \in B \Longrightarrow \text{Prob}_z[\ N'(x,\alpha(n),\beta(n),z) \text{ accepts }] \leq \frac{1}{2} - \frac{1}{r'(n)}\ ,$$

where $n = |x|$ and z is chosen uniformly over $\{0,1\}^{q'(n)}$.

In the next lemma we use the standard amplification techniques (q.v. Lemma 3.4 in [Sch85]) to achieve very high probabilities.

Lemma 6.16. Let N be any NP machine. Suppose there exist two disjoint events, A and B, such that for some polynomials q and r

$$x \in A \Longrightarrow \text{Prob}_z[\ N(x,z) \text{ accepts }] \geq \frac{1}{2} + \frac{1}{r(n)}$$
$$x \in B \Longrightarrow \text{Prob}_z[\ N(x,z) \text{ accepts }] \leq \frac{1}{2} - \frac{1}{r(n)}\ ,$$

where $n = |x|$ and z is chosen uniformly over $\{0,1\}^{q(n)}$. Then, for any polynomial p, there is an NP machine N' and a polynomial q' such that

$$x \in A \Longrightarrow \text{Prob}_z[\ N'(x,z) \text{ accepts }] > 1 - 2^{-p(n)}$$
$$x \in B \Longrightarrow \text{Prob}_z[\ N'(x,z) \text{ accepts }] < 2^{-p(n)},$$

where $n = |x|$ and z is chosen uniformly over $\{0,1\}^{q'(n)}$.

The next lemma gives an alternate characterization of of the class NP/*poly*. It is known that languages which can be characterized by *Merlin-Arthur-Merlin* games (q.v. [Bab85]) or by the quantifier structure $[\exists\exists^{+}\exists/\forall\exists^{+}\forall]$ (q.v. [Zac86]) are in the class BP·NP. Lemma 6.17 states that the nonuniform versions of such languages are in the class BP·(NP/*poly*) which is the same as the class NP/*poly*.

Lemma 6.17. Let N be an NP machine and let p, q and r be polynomials. Let the functions α and β be nice nonuniform probability bounds. Suppose that a language L satisfies the following:

$$x \in L \Longrightarrow \exists y \ \mathrm{Prob}_z[\ N(x, y, f(0^n), z) \text{ accepts }] \geq \alpha(n)$$

$$x \notin L \Longrightarrow \forall y \ \mathrm{Prob}_z[\ N(x, y, f(0^n), z) \text{ accepts }] \leq \beta(n),$$

where $n = |x|$, $\alpha(n) - \beta(n) > 1/r(n)$, f is an advice function, y is taken from $\{0,1\}^{p(n)}$ and z is chosen uniformly over $\{0,1\}^{q(n)}$. Then, $L \in \mathrm{NP}/poly$.

Proof: The goal of the proof is to show that $L \in \mathrm{NP}/poly$. The proof starts by invoking the previous lemmas to center and amplify the probability bounds. After the probability bounds has been sufficiently amplified, the $\exists$ and $\forall$ quantifiers can be moved inside the probability quantifier. This finally results in a BP·(NP/*poly*) expression which, in turn, implies that $L \in \mathrm{NP}/poly$.

First, we define the events A and B as follows:

$$A \stackrel{\text{def}}{=} \{(x,a,y) \mid x \in L,\ a = f(0^n),\ y \in \{0,1\}^{p(n)},\ \text{and}$$
$$\mathrm{Prob}_z[\ N(x, y, f(0^n), z) \text{ accepts }] \geq \alpha(n)\}$$

$$B \stackrel{\text{def}}{=} \{(x,a,y) \mid x \notin L,\ a = f(0^n),\ \text{and } y \in \{0,1\}^{p(n)}\}.$$

where $n = |x|$, and z is chosen uniformly over $\{0,1\}^{q(n)}$. Then, a straightforward application of Lemmas 6.15 and 6.16 produces an NP machine N' such that

$$x \in L \Longrightarrow \exists y \ \mathrm{Prob}_z[\ N'(x, y, f(0^n), \alpha(n), \beta(n), z) \text{ accepts }] > 1 - 2^{-2p(n)}$$

$$x \notin L \Longrightarrow \forall y \ \mathrm{Prob}_z[\ N'(x, y, f(0^n), \alpha(n), \beta(n), z) \text{ accepts }] < 2^{-2p(n)},$$

where $n = |x|$, f is a non-uniform advice function, y is taken from $\{0,1\}^{p(n)}$ and z is chosen uniformly over $\{0,1\}^{q'(n)}$. To shorten the notation, we define a new advice function g:

$$g(0^n) = (f(0^n), \alpha(n), \beta(n)).$$

Thus, we have

$$x \in L \Longrightarrow \exists y \ \mathrm{Prob}_z[\ N'(x, y, g(0^n), z) \text{ accepts }] > 1 - 2^{-2p(n)}$$

$$x \notin L \Longrightarrow \forall y \ \mathrm{Prob}_z[\ N'(x, y, g(0^n), z) \text{ accepts }] < 2^{-2p(n)}.$$

In the next step, we construct a new NP machine N'' such that

$$N''(x, g(0^n), z) \text{ accepts } \iff \exists y,\ N'(x, y, g(0^n), z) \text{ accepts } .$$

(N'' simply guesses the witness y.) Now, suppose $x \in L$. Let y_0 be a witness such that $N'(x, y_0, g(0^n), z)$ accepts with high probability. Then, $N''(x, g(0^n), z)$ will also accept with high probability, since N'' will accept by guessing the same y_0. Thus,

$$x \in L \Longrightarrow \mathrm{Prob}_z[\ N''(x, g(0^n), z) \text{ accepts }] > 1 - 2^{-2p(n)}.$$

On the other hand, suppose that $x \notin L$. Then,

$$\mathrm{Prob}_z[\ \exists y,\ N'(x, y, g(0^n), z) \text{ accepts }] < 2^{-p(n)}.$$

To see this, suppose that the probability is greater than $2^{-p(n)}$. Then, let m be the number of pairs (y, z) such that $N'(x, y, g(0^n), z)$ accepts. By assumption, $m > 2^{-p(n)} \cdot 2^{q'(n)} = 2^{q'(n)-p(n)}$. Since there are only $2^{p(n)}$ many different y's, for some particular y_0, there must be more than $m/2^{p(n)} = 2^{q'(n)-2p(n)}$ many z's such that $N'(x, y_0, g(0^n), z)$ accepts. However, this means

$$\mathrm{Prob}_z[\ N'(x, y_0, g(0^n), z) \text{ accepts }] > 2^{-2p(n)},$$

which violates the condition that $x \notin L$. Thus,

$$x \notin L \Longrightarrow \mathrm{Prob}_z[\ N''(x, g(0^n), z) \text{ accepts }] < 2^{-p(n)}.$$

Combining the two cases, we have

$$\mathrm{Prob}_z[\ x \in L \iff N''(x, g(0^n), z) \text{ accepts }] > 1 - 2^{-p(n)}.$$

Using more standard notation, this statement says $L \in \mathrm{BP}\cdot(\mathrm{NP}/poly)$. Moreover, by a lemma due to Schöning (q.v. [Sch89] Corollary 3.6), we know that $\mathrm{BP}\cdot(\mathrm{NP}/poly) \subseteq (\mathrm{NP}/poly)/poly = \mathrm{NP}/poly$. Thus, $L \in \mathrm{NP}/poly$. □

Corollary 6.18. If the language $\overline{\mathrm{SAT}}$ satisfies the properties in Lemma 6.17, then $\mathrm{NP}/poly = \text{co-NP}/poly$ and $\mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}$.

Proof: By Lemma 6.17, $\overline{\mathrm{SAT}} \in \mathrm{NP}/poly$. Then, using Yap's theorems [Yap83], it follows that $\mathrm{NP}/poly = \text{co-NP}/poly$ and $\mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}$. □

6.7 Summary

Randomization is a very useful tool for classifying the computational complexity of problems. In this chapter, we have shown that randomization allows us to describe the complexity of the unique satisfiability problem in a way that deterministic complexity classes have not. That is, we have shown that USAT is complete for D^{P}. Even though, Valiant and Vazirani were the ones who have provided the randomized reduction from $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ to USAT, it is the proofs in this chapter which actually establish the "completeness" of USAT.

Bibliography

[AM77] L. M. Adleman and K. Manders. Reducibility, randomness, and intractibility [*sic*]. In *ACM Symposium on Theory of Computing*, pages 151–163, 1977.

[Bab85] L. Babai. Trading group theory for randomness. In *ACM Symposium on Theory of Computing*, pages 421–429, 1985.

[BBJ+89] A. Bertoni, D. Bruschi, D. Joseph, M. Sitharam, and P. Young. Generalized Boolean hierarchies and the hierarchy over RP. Technical Report 809, Department of Computer Sciences, University of Wisconsin—Madison, 1989.

[BCO91] R. Beigel, R. Chang, and M. Ogiwara. A relationship between difference hierarchies and relativized polynomial hierarchies. Technical Report TR 91-1184, Cornell Department of Computer Science, January 1991.

[BDG88] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.

[BDG90] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II*, volume 22 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1990.

[Bei87] R. Beigel. Bounded queries to SAT and the Boolean hierarchy. Technical Report 7, Department of Computer Science, The Johns Hopkins University, 1987. To appear in *Theoretical Computer Science*.

[Bei88] R. Beigel. NP-hard sets are p-superterse unless R = NP. Technical Report 4, Department of Computer Science, The Johns Hopkins University, 1988.

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

[BG82] A. Blass and Y. Gurevich. On the unique satisfiability problem. *Information and Control*, 55(1–3):80–88, 1982.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P =? NP question. *SIAM Journal on Computing*, 4(4):431–442, December 1975.

[BH77] L. Berman and J. Hartmanis. On isomorphism and density of NP and other complete sets. *SIAM Journal on Computing*, 6:305–322, June 1977.

[BHZ87] R. Boppana, J. Håstad, and S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2):127–132, 1987.

[BJY91] D. Bruschi, D. Joseph, and P. Young. A structural overview of NP optimization problems. *Algorithms Review*, 2(1):1–26, May 1991.

[BK88] R. V. Book and K. Ko. On sets truth-table reducible to sparse sets. *SIAM Journal on Computing*, 17:903–919, 1988.

[CGH+88] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The Boolean hierarchy I: Structural properties. *SIAM Journal on Computing*, 17(6):1232–1252, December 1988.

[CGH90] B. Chor, O. Goldreich, and J. Håstad. The random oracle hypothesis is false. Technical Report 631, Department of Computer Science, Technion, 1990. Revised paper to appear in the *Journal of Computer and System Sciences*.

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

[Cha89] R. Chang. On the structure of bounded queries to arbitrary NP sets. In *Proceedings of the 4th Structure in Complexity Theory Conference*, pages 250–258, June 1989. Revised paper in *SIAM Journal on Computing,* 21(4):743–754, August 1992.

[Cha90] R. Chang. An example of a theorem that has contradictory relativizations and a diagonalization proof. *Bulletin of the European Association for Theoretical Computer Science*, 42:172–173, October 1990.

[CK90a] R. Chang and J. Kadin. The Boolean hierarchy and the polynomial hierarchy: a closer connection. In *Proceedings of the 5th Structure in Complexity Theory Conference*, pages 169–178, July 1990. To appear in *SIAM Journal on Computing*.

[CK90b] R. Chang and J. Kadin. On computing Boolean connectives of characteristic functions. Technical Report TR 90-1118, Cornell Department of Computer Science, May 1990. To appear in *Mathematical Systems Theory*.

[CK90c] R. Chang and J. Kadin. On the structure of uniquely satisfiable formulas. Technical Report TR 90-1124, Cornell Department of Computer Science, May 1990.

[CKR91] R. Chang, J. Kadin, and P. Rohatgi. Connections between the complexity of unique satisfiability and the threshold behavior of randomized reductions. In *Proceedings of the 6th Structure in Complexity Theory Conference*, pages 255–269, July 1991. Revised paper to appear in *Journal of Computer and System Sciences*.

[CR90a] R. Chang and P. Rohatgi. On unique satisfiability and random reductions. *Bulletin of the European Association for Theoretical Computer Science*, 42:151–159, October 1990.

[CR90b] R. Chang and P. Rohatgi. Random reductions in the Boolean hierarchy are not robust. Technical Report TR 90-1154, Cornell Department of Computer Science, October 1990.

[GMW86] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 174–187, 1986.

[Gre91] F. Green. On the power of deterministic reductions to $C_{=}P$. Technical Report LSI-91-17, UPC Barcelona, 1991.

[GS86] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In *ACM Symposium on Theory of Computing*, pages 59–68, 1986.

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

[Har87a] J. Hartmanis. Collapsing hierarchies. *Bulletin of the European Association for Theoretical Computer Science*, 33:26–39, October 1987.

[Har87b] J. Hartmanis. Sparse complete sets for NP and the optimal collapse of the polynomial hierarchy. *Bulletin of the European Association for Theoretical Computer Science*, 39:73–81, June 1987.

[HCRR90] J. Hartmanis, R. Chang, D. Ranjan, and P. Rohatgi. Structural complexity theory: Recent surprises. In *Proceedings of the 2nd Scandinavian Workshop on Algorithm Theory*, volume 447 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 1990. Revised paper to appear in the *Journal of Computer and System Sciences*.

[Hem87] L. A. Hemachandra. The strong exponential hierarchy collapses. In *ACM Symposium on Theory of Computing*, pages 110–122, 1987.

[HL91] S. Homer and L. Longpre. On reductions of NP sets to sparse sets. In *Proceedings of the 6th Structure in Complexity Theory Conference*, pages 79–88, July 1991.

[Hof79] C. M. Hoffman. *Group-Theoretic Algorithms and Graph Isomorphism*, volume 136 of *Lecture Notes in Computer Science*. Springer-Verlag, 1979.

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

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

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

[Kad87] J. Kadin. Is one NP question as powerful as two? Technical Report TR 87-842, Cornell Department of Computer Science, June 1987.

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

[Kad89] J. Kadin. $P^{NP[\log n]}$ and sparse Turing complete sets for NP. *Journal of Computer and System Sciences*, 39:282–298, December 1989.

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

[KL82] R. Karp and R. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28:191–209, 1982.

[Koz80] D. Kozen. Indexings of subrecursive classes. *Theoretical Computer Science*, 11:277–301, 1980.

[Kre88] M. W. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36(3):490–509, 1988.

[KS85] K. Ko and U. Schöning. On circuit size complexity and the low hierarchy for NP. *SIAM Journal on Computing*, 14(1):41–51, February 1985.

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

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

[Lad75] R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22(1):155–171, January 1975.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 2–10, 1990.

[LJK87] K. Lange, B. Jenner, and B. Kirsig. The logarithmic alternation hierarchy collapses: $A\Sigma_2^L = A\Pi_2^L$. In *14th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 267 of *Lecture Notes in Computer Science*, pages 529–541. Springer-Verlag, 1987.

[LLS75] R. E. Ladner, N. A. Lynch, and A. L. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–123, 1975.

[Lon82] T. Long. A note on sparse oracles for NP. *Journal of Computer and System Sciences*, 24:224–232, 1982.

[LP81] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, 1981.

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

[OW90] M. Ogiwara and O. Watanabe. On polynomial time truth-table reducibility of NP sets to sparse sets. In *ACM Symposium on Theory of Computing*, pages 457–467, 1990.

[PY84] C. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28(2):244–259, April 1984.

[Sch82] U. Schöning. A uniform approach to obtain diagonal sets in complexity classes. *Theoretical Computer Science*, 18:95–103, 1982.

[Sch83] U. Schöning. A low and a high hierarchy in NP. *Journal of Computer and System Sciences*, 27:14–28, 1983.

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

[Sch88] U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37:312–323, December 1988.

[Sch89] U. Schöning. Probabilistic complexity classes and lowness. *Journal of Computer and System Sciences*, 39(1):84–100, 1989.

[Sha90] A. Shamir. IP = PSPACE. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 11–15, 1990.

[SHL65] R. E. Stearns, J. Hartmanis, and P. M. Lewis II. Hierarchies of memory limited computations. In *Proceedings of the Sixth Annual Symposium on Switching Circuit Theory and Logical Design*, pages 179–190, 1965.

[Soa87] R. I. Soare. *Recursively Enumerable Sets and Degrees*. Springer-Verlag, 1987.

[SW87] U. Schöning and K. Wagner. Collapsing oracle hierarchies, census functions and logarithmically many queries. Technical Report 140, Institut für Mathematik, University of Augsburg, April 1987.

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

[Tod89] S. Toda. On the computational power of PP and $\oplus$P. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 514–519, 1989.

[VV86] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(1):85–93, 1986.

[Wag88] K. Wagner. Bounded query computations. In *Proceedings of the 3rd Structure in Complexity Theory Conference*, pages 260–277, June 1988.

[Wag89] K. Wagner. Number-of-query hierarchies. Technical Report 4, Institut für Informatik, Universität Würzburg, February 1989.

[WW85] K. Wagner and G. Wechsung. On the Boolean closure of NP. In *Proceedings of the 1985 International Conference on Fundamentals of Computation Theory*, volume 199 of *Lecture Notes in Computer Science*, pages 485–493. Springer-Verlag, 1985.

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

[Zac86] S. Zachos. Probabilistic quantifiers, adversaries, and complexity classes: An overview. In *Structure in Complexity Theory*, volume 223 of *Lecture Notes in Computer Science*, pages 383–400. Springer-Verlag, 1986.

[ZH86] S. Zachos and H. Heller. A decisive characterization of BPP. *Information and Control*, 69(1–3):125–135, 1986.