Boolean Hierarchies - On Collapse Properties and Query Order

Dissertation

zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat)

vorgelegt dem Rat der
Fakultät für Mathematik und Informatik
der Friedrich-Schiller-Universität Jena

von Diplom-Mathematiker
Harald Hempel
geboren am 7.August 1969 in Jena

Gutachter:	Prof. Dr. Gerd Wechsung
	Prof. Edith Hemaspaandra
	Prof. Dr. Klaus Wagner

Tag des Rigorosums:	19.8.1998
Tag der öffentlichen Verteidigung:	2.10.1998

To my family

Acknowledgements

Words can not express my deep gratitude to my advisor Professor Gerd Wechsung. Generously he offered support, guidance, and encouragement throughout the past four years. Learning from him and working with him was and still is a pleasure and privilege I much appreciate. Through all the ups and downs of my research his optimism and humane warmth have made the downs less frustrating and the ups more encouraging.

I want to express my deep gratitude to Professor Lane Hemaspaandra and Professor Edith Hemaspaandra. Allowing me to become part of so many joint projects has been a wonderful learning experience and I much benefited from their scientific expertise. Their generous help and advice helped me to gain insights into how research is done and made this thesis possible.

For serving as referees for this thesis I am grateful to Professor Edith Hemaspaandra and Professor Klaus Wagner.

I want to thank all my colleagues at Jena, especially, Haiko Müller, Dieter Kratsch, Jörg Rothe, Johannes Waldmann, and Maren Hinrichs for generously offering help and support regarding the many little things that scientific work seems to require these days (LaTeX, unix, email, etc). Working in the theory group of Professor Gerd Wechsung has made the past years a wonderful time.

I want to thank my parents who brought about my interest into mathematics. For their encouragement and support I am grateful to my family, my mother-in-law, and most of all, my loving wife Ines.

The research underlying this thesis has been supported in part by grant NSF-INT-9513368/DAAD-315-PRO-fo-ab and was in part done while visiting Le Moyne College.

Chapter 3 and Section 4.2 are based on joint work with Edith Hemaspaandra and Lane Hemaspaandra [HHH98d] and part of it appeared in journal as [HHH98e]. The research underlying Section 4.3 is joint work with Edith Hemaspaandra and Lane Hemaspaandra [HHH98a]. The results of Section 5.2 were discovered jointly with Lane Hemaspaandra and Gerd Wechsung and will appear in journal as [HHW99]. Section 5.4 is based on joint work with Edith Hemaspaandra and Lane Hemaspaandra and has (partially) appeared in journal as [HHH97a].

Contents

List of Figures

Chapter 1

Introduction

Complexity theory studies the inherent computational complexity (difficulty) of problems. By problem we usually mean decision problems, the question of whether a given object belongs to a certain set (in other words, has a certain property) or not. But how do we measure difficulty? Quite naturally, the time needed to come up with a solution is a reasonable and often used measure. More precisely, the time measure of a problem A is a function t satisfying that for each input x the answer to the question "$x \in A$?" can be found in at most $t(|x|)$ time units, where $|x|$ is the length of the representation for x. In complexity theory time is measured in terms of steps needed by a Turing machine to solve the problem. Turing machines, first defined by Turing [Tur36] and Post [Pos36], are a computational model that is simple yet powerful enough to capture the very notion of computability. The time measure allows to classify problems. Those being computable by a deterministic Turing machine with a time function that is bounded by a polynomial form the class P [Edm65]. The class P is a theoretical concept intended to capture the spirit of feasible computation. The word "feasible" should be viewed in a rather theoretical context. Clearly, problems having a time function that is a polynomial of high degree, say n^{23}, are certainly not considered to be efficiently solvable. However, polynomials do not grow too fast and possess a number of nice properties, for instance they are closed under composition. Furthermore, the class P is a very robust notion not depending on the definitional variations of the underlying computational model. In contrast, problems having a time function that is bounded by 2^{cn}, where c is some constant and n the length of the input, are (until one shows a better time bound) not considered to be feasibly computable. The class E is the collection of all those problems.

It is known that there are problems in E that are not in P [HS66]. Examples of problems in P are addition of natural numbers and sorting natural numbers. The EULER TOUR problem emerging from Eulers famous Königsberger Brücken-Problem [Eul36], that given a graph asks whether it is possible to walk through the graph along its edges in such a way that every *edge* is touched exactly once, is another prominent member of the class P.

A seemingly slight but crucial, as we will see in a moment, variation of this problem leads to the so-called HAMILTON CIRCUIT problem (HC). Given a graph one is asked

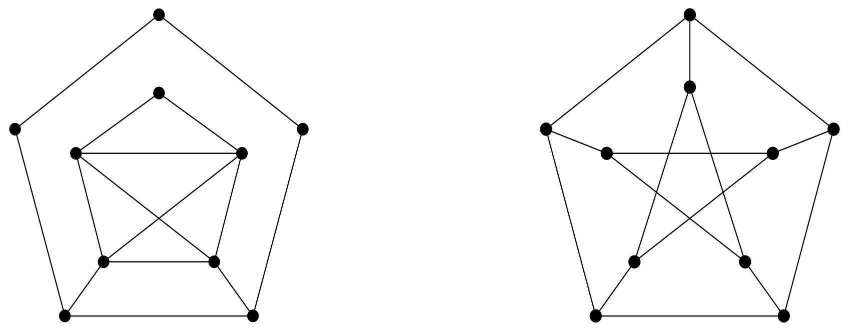


Figure 1.1: The left graph contains both, an Euler tour and a Hamilton Circuit, whereas the right graph contains neither an Euler tour nor a Hamilton circuit.

to find a way of moving through the graph along its edges such that starting from some vertex one returns to it while visiting every other *vertex* exactly once. The best known algorithms for HC need exponential time, HC $\in$ E, and consist essentially in testing for all permutations of the vertices of the input graph whether the vertices can be visited in the order given by the permutation.

It is not known whether HC $\in$ P and people have tried to understand why this question has resisted all solution efforts. It turned out that a variant of deterministic Turing machines, so-called nondeterministic Turing machines can solve HC in polynomial time, the amount of nondeterminism needed is growing exponentially with the size of the input graph. This gives rise to a new class of problems between P and E, namely, NP, the class of problems that can be solved by some nondeterministic polynomial-time Turing machine. It is known that HC is among the hardest problems in NP in the sense that HC $\in$ P would immediately imply P = NP. In fact, the question P $\overset{?}{=}$ NP is the most famous open question in complexity theory having reformulations in many of its areas. While being concerned with resolving the P $\overset{?}{=}$ NP question researchers quickly found out that there is a much richer structure of complexity classes between P and E. Variations to the acceptance mechanism of nondeterministic Turing machines and the notion of oracle Turing machines are a few examples for other computing paradigms that allow to exactly pinpoint many naturally arising computation problems. For instance, oracle Turing machines are Turing machines that have access to external information sources; the oracle machines make so called oracle queries and receive–quite in contrast to the Greek mythology–always a clear answer "yes" or "no" in unit time.

The observation that seemingly no nondeterministic polynomial-time Turing machine can solve the MINIMAL EQUIVALENT EXPRESSION problem (MEE) [MS72, Sto77, GJ79] (see also [HW97a] for latest results on the complexity of MEE), which, informally, is the question of whether a given boolean formula has a shorter representation, together with the fact that a nondeterministic oracle Turing machine equipped with an oracle from NP can well solve MEE in polynomial time led to the definition of the polynomial hierarchy [MS72, Sto77]. The polynomial hierarchy is inductively defined via the concept of deterministic and nondeterministic polynomial-time oracle Turing machines. P, NP, and the complement class coNP form its first level. Much has been learned about the structure of the polynomial hierarchy and a number of very elegant techniques have been developed while studying its properties. The literature on the polynomial hierarchy, in particular, on complete sets, the existence of tally and sparse complete sets, relativizations, characterizations, collapse properties, refinements, and much more is immense (for references see [BDG88, BDG90, WW86, Pap94]). Over the years the conviction has grown that the polynomial hierarchy is infinite, though a rigorous proof has yet to be found. Part of its importance today stems from the fact that it still serves as the main yardstick for classifying other complexity classes. Relatedly, it is so widely accepted that the polynomial hierarchy is strict and infinite that an implied collapse of it often serves as a strong sign that other events are quite unlikely to happen.

Though the polynomial hierarchy allows to classify many of the arising computation problems, it often seemed to be to rough to exactly classify all of it. Many naturally arising computation problems are contained in Δ_2^{p} (a class from the second level of the polynomial hierarchy), though one easily verifies that they do not require the full computational power of Δ_2^{p}. Recall that HC belongs to the class NP. For the weighted version of HC, often called TRAVELING SALESPERSON problem (TSP), each edge in the input graph is assigned a positive integer weight (length) and one is asked to decide whether there is a Hamilton circuit of length less then or equal to a given integer k (the length of a circuit is simply the sum of the length of the edges on this circuit). It is known that TSP can be solved with a nondeterministic Turing machine in polynomial time, TSP $\in$ NP. Yet EXACT TSP, that asks whether the shortest Hamilton circuit in the input graph has length *exactly* k is not known to be in NP. Though EXACT TSP is contained in the second level of the polynomial hierarchy, Δ_2^{p}, not all of this levels power is needed to solve it. Essentially, every instance of EXACT TSP can be easily reduced to two TSP problems, the first one accounting for the fact that there is a Hamilton circuit of length at most k in the graph and the second one testing that no Hamilton circuit of length $k-1$ exists. This computing paradigm is reflected by the class DP [PY84], DP $= \{L_1 \cap \overline{L_2} \mid L_1, L_2 \in \mathrm{NP}\}$. Its generalization, for instance as nested differences or alternating sums of NP sets, leads to the concept of the boolean hierarchy. In fact, DP is exactly its second level. The boolean hierarchy was introduced almost simultaneously by several groups of authors with a variety of definitions ranging from acceptance types [Wec85], boolean hardware over NP [CH86], and symmetric differences of NP sets [KSW87], but can in concept already be found in the work of Hausdorff [Hau14]. Though it is known that NP is closed under union and intersection, it is not known to be closed under complementation. The boolean hierarchy provides a rich structure inside the closure of NP under the boolean operations union, intersection, and complementation. Being sandwiched between the polynomial hierarchies' first and second levels the boolean hierarchy forms a refinement of the polynomial hierarchy. As it is the case for the polynomial hierarchy, complete sets, relativized separation, and collapse results have been studied for the boolean hierarchy as well in detail [CGH$^+$88, CGH$^+$89].

1.1 The Boolean and Polynomial Hierarchies Connection

Whenever a hierarchy is defined, one of the first questions going to be asked is whether the hierarchy is infinite or not. For quite a number of hierarchies in computational complexity the answer to this question has been found. For instance, Hemaspaandra showed that the strong exponential hierarchy is a finite hierarchy [Hem89], a number of authors proved that the logarithmic space and linear space hierarchies are finite structures [SW88, Tod87, LJK89], and Immerman and Szelepcsényi independently obtained the breakthrough result that the nondeterministic space classes are closed under complementation implying that the nondeterministic logarithmic and linear space hierarchies collapse to its first levels [Imm88, Sze88].

However, for the polynomial and the boolean hierarchies we do not know whether they are infinite or not. For both hierarchies relativized worlds have been constructed in which they are infinite or finite [Yao85, Ko88, CGH^{+}88], respectively. Hence, non relativizable proof techniques are required for ultimately determining the dimensions of those hierarchies. Though most researchers believe that both hierarchies are infinite, the hope for a proof of that conviction has somewhat diminished over the years. Unfortunately, proving strict inclusions is in general extremely difficult, especially for classes within the polynomial hierarchy. In fact, though it is known that $\mathrm{P} \subsetneq \mathrm{E}$ [HS66], it is not even known whether $\mathrm{P} \overset{?}{=} \mathrm{PSPACE}$, not to speak of the famous $\mathrm{P} \overset{?}{=} \mathrm{NP}$ question.

A number of techniques have been developed to overcome this situation by at least giving evidence that certain inclusions are strict. Linking the collapse of complexity classes to a collapse of other complexity classes is a technique being used since the early years of complexity theory. It unifies the issues of the relative strength of different computation models. Earliest examples were found while studying the famous $\mathrm{P} \overset{?}{=} \mathrm{NP}$ question, for instance, “If P = NP then NP = coNP.” Today this technique is also often used to add more weight to the already existing evidence that two complexity classes are not equal by showing that their equality would immediately provide a very unlikely collapse of other complexity classes, for instance a collapse of the polynomial hierarchy. This technique has also been used to link the collapse of various function classes to collapses of complexity classes [Val76, Sel94b, VW93, HW97c].

Since the boolean hierarchy does form a refinement between the first and the second levels of the polynomial hierarchy, as it is sandwiched between P and Δ_2^{p}, does a collapse of the boolean hierarchy imply a collapse of the polynomial hierarchy? In the trivial case, the boolean hierarchy collapsing at its first level which happens to be also the first level of the polynomial hierarchy, the answer is yes. A yes answer in the general case would immediately yield that infinity of the polynomial hierarchy (a belief shared by almost all complexity theorists) implies infinity of the boolean hierarchy.

A first successful step into that direction has been made by Kadin [Kad88] in proving that a collapse of the boolean hierarchy implies a collapse of the polynomial hierarchy to Σ_3^{p}. This first result is significant in two ways. First, the structures of the polynomial and the boolean hierarchies are tied together. Second, the underlying proof technique, the so-called easy-hard technique, was introduced into complexity theory. The easy-hard technique has much influenced research in computational complexity not only in the attempts of linking boolean and polynomial hierarchy collapses even closer together but also, for instance, in the recent interest in downward translation of equality, a line of research we will in detail consider in Chapter 4.

Modifications to the easy-hard technique together with a more sophisticated use of it allowed Wagner [Wag87, Wag89], Chang and Kadin [CK96], and Beigel, Chang, and Ogihara [BCO93] to subsequently improve Kadin's original result. The best known result today has been obtained by Beigel, Chang, and Ogihara [BCO93] and shows that a collapse of the boolean hierarchy at level m implies a collapse of the polynomial hierarchy to a level within the mth level of the boolean hierarchy over Σ_2^{p}.

In Chapter 3 we will take a close look at the series of papers which led to this result. After stating the key results of those five papers, [Kad88, Wag87, Wag89, CK96, BCO93], we analyze the development of the easy-hard technique and its use in (conditionally) collapsing the polynomial hierarchy in Section 3.2. The refined structure of complexity classes inside the mth level of the boolean hierarchy over Σ_2^{p}, which was first defined and studied by Selivanov [Sel94a, Sel95], allows to pinpoint the induced collapse of the polynomial hierarchy at an even deeper level than being observed in [BCO93]. In Section 3.3 we prove our new result which based on a careful analysis of the proof given in [BCO93] and double application of one of its key ideas yields that a collapse of the boolean hierarchy at level m, $m \geq 2$, implies a collapse of the polynomial hierarchy to $\mathrm{BH}_m \boldsymbol{\Delta} \mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$.

This result adds much insight to the connection between the boolean hierarchy and the boolean hierarchy over Σ_2^{p}. Chang and Kadin [CK96] in proving $\mathrm{BH}_m = \mathrm{coBH}_m \implies \mathrm{DIFF}_m(\Sigma_2^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_2^{\mathrm{p}}) = \mathrm{PH}$ argued that some underlying connection between the boolean hierarchy and the boolean hierarchy over Σ_2^{p} is responsible for this result. In particular, they asked whether there is some straightforward argument showing that $\mathrm{BH}_m = \mathrm{coBH}_m \implies \mathrm{DIFF}_m(\Sigma_2^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_2^{\mathrm{p}})$ which could be used to prove the collapse of the polynomial hierarchy. Though the result of Beigel, Chang, and Ogihara [BCO93] made this question disappear from the list of open problems, our result sheds new light on this connection. It is quite easy to verify that $\mathrm{BH}_m = \mathrm{coBH}_m$ implies $\mathrm{BH}_m \boldsymbol{\Delta} \mathrm{DIFF}_j(\Sigma_2^{\mathrm{p}}) = \mathrm{co}(\mathrm{BH}_m \boldsymbol{\Delta} \mathrm{DIFF}_j(\Sigma_2^{\mathrm{p}}))$ for all $j \geq 1$, especially for $j = m - 1$. Unfortunately this result says nothing about the collapse of the polynomial hierarchy. In fact, the main difficulty to overcome in our proof is to show that $\Sigma_3^{\mathrm{p}} \subseteq \mathrm{BH}_m \boldsymbol{\Delta} \mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$.

1.2 Downward Collapse

The collapse of hierarchies has been a central topic in complexity theory from the beginning. Relatedly, does the relative strength of more powerful computing paradigms depend on the relative strength of less powerful ones, or vice versa? Results with this general flavor are refered to as upward and downward collapse, respectively. The very nature of upward and downward collapse can be observed, for example, at the architectural feature of the Roman semicircular arch (see Figure 1.2). On one hand, the Roman arch collapses when removing

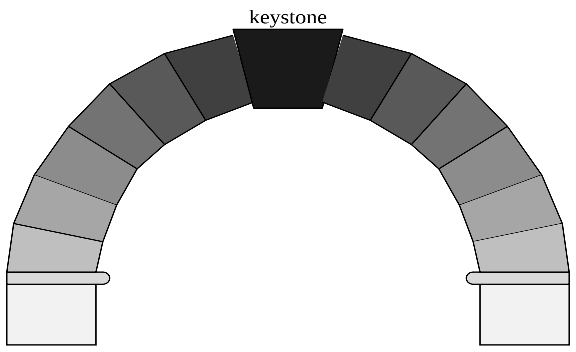


Figure 1.2: The Roman Semicircular Arch Displays Upward and Downward Collapse

one of the building blocks at its foundation. A collapse at lower levels causes dramatic collapses at higher levels, the *collapse* translates *upwards*. On the other hand, the arch also collapses when removing its keystone and so displays *downward collapse*. The keystone though located at the very top of the arch provides stability and strength to the entire construction.

Formally, for complexity classes $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{D}_1$, and $\mathcal{D}_2$ such that $\mathcal{C}_1 \cup \mathcal{C}_2 \subseteq \mathcal{D}_1 \cap \mathcal{D}_2$ (ideally the inclusion is strict or at least strongly believed to be strict), upward collapse would be "If $\mathcal{C}_1 = \mathcal{C}_2$ then $\mathcal{D}_1 = \mathcal{D}_2$." Similarly, a result "If $\mathcal{D}_1 = \mathcal{D}_2$ then $\mathcal{C}_1 = \mathcal{C}_2$" is called downward collapse. Roughly speaking, the equality $\mathcal{D}_1 = \mathcal{D}_2$ has the same effect on the lower complexity classes $\mathcal{C}_1$ and $\mathcal{C}_2$ as a removal of the keystone of the arch had on its lower building blocks, they collapse.

Upward collapse is frequently observed in computational complexity theory. Most hierarchies due to their inductive definition display upward collapse properties, especially the polynomial and the boolean hierarchies. For instance, $\Sigma_k^{\mathrm{p}} = \Pi_k^{\mathrm{p}} \implies \mathrm{PH} = \cdots = \Sigma_{k+1}^{\mathrm{p}} = \Sigma_k^{\mathrm{p}}$ [Sto77] and $\mathrm{BH}_m = \mathrm{coBH}_m \implies \mathrm{BH} = \cdots = \mathrm{BH}_{m+1} = \mathrm{BH}_m$ [CGH$^+$88]. The results regarding the collapse of the polynomial hierarchy induced by a collapse of the boolean hierarchy we review and prove in Chapter 3 are all upward collapse results.

Downward collapse is a rather rare event in structural complexity theory. Perhaps the most well known result having the flavor of downward collapse though containing an unspecified parameter is "If PH = PSPACE then $(\exists k)[\Sigma_k^{\mathrm{p}} = \Pi_k^{\mathrm{p}} = \mathrm{PH}]$" [Wra77]. Examples of downward collapse though not all satisfy our formal description above are "If NP $\subseteq$ BPP then NP = R" [Ko82], "If EH = E then P = BPP" [BFNA93], and "If $\mathrm{P}^{\mathrm{NP}[1]} = \mathrm{EXP}$ then NP = EXP" [HKR93]. There are also examples involving degenerate certificate schemes [HRW97] and circuit-related classes [All86, HY84]. Cases in which the collapse of larger classes implies that smaller classes collapse on sets of small density can for instance be found in [Boo74, HIS85, RRW94] (in contrast see [HJ95]).

The first downward collapse result linking classes of the bounded-query hierarchies and classes of the polynomial hierarchy has been obtained by Hemaspaandra, Hemaspaandra, and Hempel [HHH96a]. They showed that for all $k > 2$, $\mathrm{P}^{\Sigma_k^{\mathrm{p}}[1]} = \mathrm{P}^{\Sigma_k^{\mathrm{p}}[2]} \implies \Sigma_k^{\mathrm{p}} = \Pi_k^{\mathrm{p}}$. Note that this result contains no unspecified parameter, it is not restricted to sets of small density, and the inclusion $\Sigma_k^p \cup \Pi_k^p \subseteq \mathrm{P}^{\Sigma_k^p[1]} \cap \mathrm{P}^{\Sigma_k^p[2]}$ does hold and is strongly believed to be strict. The result follows from an innovative application of the easy-hard technique, a technique developed by Kadin [Kad88] in order to link boolean and polynomial hierarchy collapses. This result has been generalized into two directions. Buhrman and Fortnow [BF98] showed that the analogue for $k = 2$ does also hold and furthermore constructed a relativized world in which the $k = 1$ analogue fails. Hemaspaandra, Hemaspaandra, and Hempel [HHH99] generalized from one versus two queries to m versus $m + 1$ queries obtaining for $k > 2$ and $m \geq 1$, $\mathrm{P}^{\Sigma_k^p}_{m\text{-tt}} = \mathrm{P}^{\Sigma_k^p}_{m+1\text{-tt}} \implies \mathrm{DIFF}_m(\Sigma_k^p) = \mathrm{coDIFF}_m(\Sigma_k^p)$. This latter result displays crisply the strong connection between the levels of boolean and the bounded-truth-table hierarchy over Σ_k^p, $k > 2$. In a very recent paper by Hemaspaandra, Hemaspaandra, and Hempel [HHH97b] the key ideas from [BF98] and [HHH99] have been combined with new methods to prove a very general downward collapse result; for $m \geq 1$ and $k > 1$, $\mathrm{P}^{\Sigma_k^p}_{m\text{-tt}} = \mathrm{P}^{\Sigma_k^p}_{m+1\text{-tt}} \implies \mathrm{DIFF}_m(\Sigma_k^p) = \mathrm{coDIFF}_m(\Sigma_k^p)$.

We will in Chapter 4 review the history and the proof techniques of this recent outburst of downward collapse results and show a new downward collapse result that strengthens a theorem from [HHH97b]. Since the easy-hard technique plays a major role also in this chapter's proofs, we will very much alike the approach in Chapter 3 first study in detail its development in the downward collapse setting. This is done in Section 4.2. In Section 4.3 we prove our new result. We show that for all $s, m \geq 1$ and all $0 < i < k - 1$, $\mathrm{DIFF}_s(\Sigma_i^p)\boldsymbol{\Delta}\mathrm{DIFF}_m(\Sigma_k^p) = \mathrm{co}(\mathrm{DIFF}_s(\Sigma_i^p)\boldsymbol{\Delta}\mathrm{DIFF}_m(\Sigma_k^p)) \implies \mathrm{DIFF}_m(\Sigma_k^p) = \mathrm{coDIFF}_m(\Sigma_k^p)$. The proof of this result in some sense merges proof ideas from Chapter 3 and Section 4.2. In particular, the proof contains two applications of the easy-hard technique. First, we make use of the easy-hard technique to provide the general framework for proving the downward collapse as it was done in [HHH99, BF98, HHH97b]. And second, inside one of the resulting cases the easy-hard technique is being used in the spirit of Chapter 3 though in a more technical form. Chapter 4 closes with some remarks and applications of the obtained results.

1.3 Query Order

The importance of order in everyday-life is beyond any doubt. So is the order in which we access information sources, databases for instance.

In complexity theory, information sources are modeled by oracles. A natural arising question thus is, does the order in which a deterministic oracle Turing machine accesses oracles from different complexity classes make a difference in the resulting computational power? In Chapter 5 we study query order in computational complexity theory. In Section 5.2 we pursue this question in the context of the boolean hierarchy and provide the first query order result in complexity theory. In particular, we study the computational power of $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$, the class of languages that are accepted by deterministic polynomial-time Turing machines that on every input make at most one query to a BH_j oracle followed by at most one query to a BH_k oracle. Does $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ equal $\mathrm{P}^{\mathrm{BH}_k:\mathrm{BH}_j}$, or are they incomparable, or does one strictly contain the other? We show that, unless the polynomial hierarchy collapses, the order of oracle access is crucial for the relative power of the complexity class. In particular, assuming that the polynomial hierarchy does not collapse, we have that if $1 \leq j \leq k$ then $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ and $\mathrm{P}^{\mathrm{BH}_k:\mathrm{BH}_j}$ differ unless $(j = k) \vee (j \text{ is even} \wedge k = j+1)$. This result is based on a characterization of query order classes in terms of reducibility closures of NP. We show that for $j, k \geq 1$,

$$\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} = \begin{cases} \mathrm{R}^{\mathrm{p}}_{j+2k-1\text{-tt}}(\mathrm{NP}) & \text{if } j \text{ is even and } k \text{ is odd,} \\ \mathrm{R}^{\mathrm{p}}_{j+2k\text{-tt}}(\mathrm{NP}) & \text{otherwise.} \end{cases}$$

Two interesting features of this characterization should be emphasized. First, informally put, the second query counts more towards the power of the class than the first query does. This follows from the fact that in the context of the boolean hierarchy the two different second queries which, depending on the answer to the first query, can be potentially asked crucially affect the resulting computational power. In sharp contrast, the fact that two different second queries can be potentially asked is irrelevant for query order classes in the polynomial hierarchy, where it has been shown that $\mathrm{P}^{\Sigma^{\mathrm{p}}_j:\Sigma^{\mathrm{p}}_k} = \mathrm{P}^{(\Sigma^{\mathrm{p}}_j,\Sigma^{\mathrm{p}}_k)}$ for all $j \neq k$ [HHH98b]. Second, there is a loss of one level when j is even and k is odd. In some sense, $j + 2k$ NP questions underpin this class. However, by arguing that a certain underlying graph must contain an odd cycle, we show that $j+2k-1$ queries suffice. The main theorem of Section 5.2 is generalized to apply also to classes with tree-like query structure.

In Section 5.3 we study query order in the polynomial hierarchy. Query order in the polynomial hierarchy is a topic being studied after the results of Section 5.2 first appeared in [HHW95]. The first result on query order in the polynomial hierarchy has been obtained by Hemaspaandra, Hemaspaandra, and Hempel [HHH98b]. They show that in sharp contrast to the boolean hierarchy query order never matters in the polynomial hierarchy. This result has been generalized by Beigel and Chang [BC] and Wagner [Wag98] to cases of more than two query rounds and more than one query in each round. We give a short overview over these previous results. However, no paper studying general query order classes in the polynomial hierarchy considers query order classes where two (or more) consecutive rounds of parallel queries are made such that though all queries are made to the same oracle the number of parallel queries made in each round differs. In order to close that gap we study query order classes of the form $\mathrm{P}^{\Sigma^{\mathrm{p}}_i:\Sigma^{\mathrm{p}}_i}_{j,k\text{-tt}}$, the class of languages that are accepted by some deterministic polynomial-time Turing machine making one round of at most j parallel queries to a Σ^{p}_i oracle followed by one round of at most k parallel queries to a Σ^{p}_i oracle. We show that for all $i, j, k \geq 1$, $\mathrm{P}^{\Sigma^{\mathrm{p}}_i:\Sigma^{\mathrm{p}}_i}_{j,k\text{-tt}} = \mathrm{R}^{\mathrm{p}}_{j+jk+k\text{-tt}}(\Sigma^{\mathrm{p}}_i)$. This result is interesting in two ways. First, query order in the polynomial hierarchy does not matter also in this case. This can be easily derived from the above result due to its symmetry. Second, though we have $\mathrm{P}^{\mathrm{NP}}_{m\text{-tt}} = \mathrm{P}^{\mathrm{BH}_m[1]}$ (see Lemma 5.2.1) an analogues statement for query order classes does not hold in general unless the polynomial hierarchy collapses. In light of Theorem 5.2.6 and the above characterization, we have $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} \neq \mathrm{P}^{\mathrm{NP}:\mathrm{NP}}_{j,k\text{-tt}}$ unless $j = 1$ or the polynomial hierarchy collapses.

In Section 5.4 we mention a number of results that are closely related to query order. A general result regarding query order with respect to base classes other than P has been obtained by Hemaspaandra, Hemaspaandra, Hempel [HHH98b]. Since the notion of query order has been first investigated in [HHW95] a number of results either growing out of the study of query order or involving the notion of query order have been obtained in various areas of computational complexity. Those results include the first downward collapse result completely within the polynomial hierarchy obtained by Hemaspaandra, Hemaspaandra, and Hempel [HHH99] (see also Chapter 4), the study of self-specifying machines by Hemaspaandra, Hempel and Wechsung [HHW95, HHW97], and results on robust completeness by Hemaspaandra, Hemaspaandra, and Hempel [HHH98c].

Chapter 2

Preliminaries

In this chapter we define basic concepts of computational complexity that are used throughout this thesis. With slight variations, everything in this chapter might also be found in any standard book on computational complexity theory, for instance [WW86, BDG88, Pap94]. We assume that the reader is familiar with the meaning and notation of the basic set theoretic and logical concepts.

2.1 Strings, Languages, and Operators

Let $\mathbb{N}$ denote the set of natural numbers. Let Pol denote the set of all polynomials in one variable over $\mathbb{N}$.

Complexity theory studies the complexity of sets of strings over a finite alphabet. Let $\Sigma = \{0, 1\}$ be our alphabet. Let # be a symbol not in Σ, $\# \notin \Sigma$, and let ϵ denote the empty string. The concatenation of strings u and v is denoted by uv. For letters $a \in \Sigma$, let $a^0 = \epsilon$ and $a^{n+1} = aa^n$ for all $n \in \mathbb{N}$. Define $\Sigma^0 = \{\epsilon\}$ and $\Sigma^{i+1} = \{uv \mid u \in \Sigma \wedge v \in \Sigma^i\}$ for all $i \in \mathbb{N}$. So Σ^i is the set of all strings of length i over Σ. By Σ^* we denote the set of all finite strings over Σ, more formally, $\Sigma^* = \bigcup_{i \in \mathbb{N}} \Sigma^i$. The length of a string x, the unique i such that $x \in \Sigma^i$, is denoted by $|x|$.

Let $\leq_{lex}$ denote the standard (quasi-) lexicographical ordering on Σ^*, in particular for strings $u, v \in \Sigma^*$, $u \leq_{lex} v$ if and only if $|u| < |v|$, or $u = v$, or $|u| = |v|$ and there exist some $w, u', v' \in \Sigma^*$ such that $u = w0u'$ and $v = w1v'$.

We consider subsets (often called languages) of Σ^*. For a set $A \subseteq \Sigma^*$, $A^{\leq n}$ denotes the set of all strings from A of length at most n, $A^{\leq n} = A \cap (\bigcup_{i \leq n} \Sigma^i)$. Similarly $A^{=n} = A \cap \Sigma^n$. For a finite set A, $||A||$ denotes the cardinality of the set A. A language A is called sparse, if and only if there exists a polynomial p such that for all $n \in \mathbb{N}$, $||A^{\leq n}|| \leq p(n)$. The characteristic function of a set A, χ_A, is defined as

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \notin A. \end{cases}$$

The complement of a set A in Σ^*, denoted by $\overline{A}$, is the set of all strings from Σ^* not being in A, $\overline{A} = \Sigma^* - A$. For languages A and B let $A \Delta B = \{x \mid x \in (A - B) \cup (B - A)\}$ denote the symmetric difference of A and B. Define $A \tilde{\Delta} B = \{\langle x, y \rangle \mid x \in A \iff y \notin B\}$.

It is often needed to map finite sequences of strings to strings. Let $\langle . \rangle$ be a bijective function mapping from $\Sigma^* \times \Sigma^*$ to Σ^* being computable and invertible in polynomial time. Such pairing functions exist. A pairing function for finite sequences of strings can be obtained from $\langle . \rangle$ in an obvious way and is also denoted by $\langle . \rangle$. In places we will for simplicity use $\langle . \rangle$ also to denote a pairing function that maps sequences of strings from $\Sigma^* \cup \{\#\}$ to Σ^*, where $\# \notin \Sigma$. Without loss of generality we require these pairing functions for finite sequences of strings to have two additional properties. First, we require that the length of the encoded sequence can be easily obtained. And second, we assume that for every fixed constant k, there exists a polynomial s_k such that for all $\ell \leq k$ and all sequences of strings $x_1, x_2, \ldots, x_\ell$,

$$|\langle x_1, x_2, \ldots, x_\ell \rangle| \leq s_k(\max\{|x_1|, |x_2|, \ldots, |x_\ell|\}).$$

In structural complexity theory sets of languages (also called complexity classes) are studied. A variety of quite useful operators that map complexity classes to complexity classes has been defined. Some of them will be of interest in this thesis and are defined below. For a set of languages $\mathcal{C}$, co$\mathcal{C} = \{\overline{C} \mid C \in \mathcal{C}\}$. For a complexity class $\mathcal{C}$, $\exists \cdot \mathcal{C}$ is defined to be the set of all languages L such that there exist a set $C \in \mathcal{C}$ and a polynomial p satisfying for all $x \in \Sigma^*$,

$$x \in L \iff (\exists y : |y| \leq p(|x|))[\langle x, y \rangle \in C].$$

Quite similarly, $\forall \cdot \mathcal{C}$ is defined to be the set of all languages L such that there exist a set $C \in \mathcal{C}$ and a polynomial p satisfying for all $x \in \Sigma^*$,

$$x \in L \iff (\forall y : |y| \leq p(|x|))[\langle x, y \rangle \in C].$$

For complexity classes $\mathcal{C}$ and $\mathcal{D}$ let

$$\mathcal{C} \boldsymbol{-} \mathcal{D} = \{C - D \mid C \in \mathcal{C} \wedge D \in \mathcal{D}\},$$

and

$$\mathcal{C} \boldsymbol{\Delta} \mathcal{D} = \{C \Delta D \mid C \in \mathcal{C} \wedge D \in \mathcal{D}\}.$$

Throughout this thesis we will try to provide figures that illustrate the inclusion structure of the studied complexity classes. Since $\subseteq$ is a partial order on the set of all subsets of Σ^* we will use Hasse diagrams. In particular, in every figure a class $\mathcal{C}$ is contained in a class $\mathcal{D}$ if there is a strictly upward directed path of bold or dotted lines leading from $\mathcal{C}$ to $\mathcal{D}$.

2.2 Turing Machines and Reductions

Our computational model is that of a k-tape multi-head Turing machine, for a more formal definition see [HU79, WW86]. Every Turing machine can potentially be equipped with oracles. Polynomial-time Turing machines are Turing machines that for a fixed polynomial p make on every input x at most $p(|x|)$ computation steps before reaching a final state. We consider nondeterministic and deterministic polynomial-time (oracle) Turing machines, DPTM and NPTM, respectively.

Without loss of generality let the Turing machines be clocked with clocks that are independent of the oracle. M^A denotes the DPTM (or NPTM) M with oracle A. By $M^A(x)$ we denote the computation of the DPTM (or NPTM) M with oracle A on input x. In places where we write $M(x)$ though M was previously said to be an oracle machine we refer to that part of the computation on input x that is independent of the oracle. A DPTM M^A accepts a language L if and only if on every input $x \in \Sigma^*$, $M^A(x)$ halts in an accepting configuration (equivalently, M^A accepts the input x) if and only if $x \in L$. An NPTM N^A accepts a language L if and only if on every input $x \in \Sigma^*$, there *exists* an accepting computation branch of $N^A(x)$ if and only if $x \in L$. $L(M)$ denotes the language accepted by some DPTM or NPTM M.

Reductions are a standard method to compare languages with respect to their complexity. Informally, a problem A reduces to a problem B if solutions for A can efficiently be computed with the help of one or several solutions for B. Many-one reductions [Kar72] (also known as Karp reductions), truth-table reductions [LLS75], and Turing-reductions [Coo71] (also called Cook reductions) are the types of reductions we are going to be concerned with.

Definition 2.2.1 Let A and B be two languages.

1. A is said to be many-one reducible to B ($A \leq_{\mathrm{m}}^{\mathrm{p}} B$) if and only if there exists a polynomial-time computable function f such that for all $x \in \Sigma^*$,
$$x \in A \iff f(x) \in B.$$

2. A is said to be truth-table reducible to B ($A \leq_{\mathrm{tt}}^{\mathrm{p}} B$) if and only if there exists a polynomial-time computable function f such that for all $x \in \Sigma^*$, $f(x)$ computes a number of strings $y_1, y_2, \ldots, y_m$ and an m-ary boolean function α such that
$$x \in A \iff \alpha(\chi_B(y_1), \chi_B(y_2), \ldots, \chi_B(y_m)) = 1.$$

3. A is said to be Turing-reducible to B ($A \leq_{\mathrm{T}}^{\mathrm{p}} B$) if and only if there exists a DPTM M such that for all $x \in \Sigma^*$,
$$x \in A \iff M^B(x) \text{ accepts.}$$

Bounding the number of strings computed by the function f in part 2 or the number of oracle queries allowed to be made by the DPTM M in part 3 by a fixed constant k leads to the notions of k-truth-table ($\leq^{\mathrm{p}}_{k\text{-tt}}$) and k-Turing ($\leq^{\mathrm{p}}_{k\text{-T}}$) reducibility [LLS75], respectively, in the obvious way.

For any reduction $\leq^{\mathrm{p}}_{\omega}$ defined above and any complexity class $\mathcal{C}$, a set A is called $\leq^{\mathrm{p}}_{\omega}$-complete for $\mathcal{C}$ if and only if $A \in \mathcal{C}$ and for all $C \in \mathcal{C}$, $C \leq^{\mathrm{p}}_{\omega} A$. $\mathrm{R}^{\mathrm{p}}_{\omega}(\mathcal{C}) = \{L \mid (\exists C \in \mathcal{C})[L \leq^{\mathrm{p}}_{\omega} C]\}$ denotes the reducibility closure (hull) of $\mathcal{C}$ with respect to $\leq^{\mathrm{p}}_{\omega}$. A complexity class $\mathcal{C}$ is said to be closed under $\leq^{\mathrm{p}}_{\omega}$ if and only if $\mathrm{R}^{\mathrm{p}}_{\omega}(\mathcal{C}) = \mathcal{C}$.

2.3 Central Complexity Classes and Hierarchies

Very informally, a complexity class is a set of languages that have the same complexity with respect to some resource of a Turing machine. The complexity class P is the set of all languages $L \in \Sigma^*$ that are accepted by some DPTM. FP denotes the set of all functions that are computable by some DPTM. Similarly, NP is the set of all languages $L \in \Sigma^*$ that are accepted by some NPTM. NP is closed under many-one reductions. SATISFIABILITY (SAT), the set of all satisfiable boolean formulas, is a many-one complete language for NP. Observe that for every boolean formula F it holds that $F \in \mathrm{SAT} \iff F_0 \in \mathrm{SAT} \vee F_1 \in \mathrm{SAT}$, where F_0 and F_1 are obtained from F by assigning 0 and 1, respectively, to one fixed variable. This property is called the self-reduction of SAT.

For a complexity class $\mathcal{C}$, $\mathrm{P}^{\mathcal{C}}$ ($\mathrm{FP}^{\mathcal{C}}$) and $\mathrm{NP}^{\mathcal{C}}$ denote the set of languages (functions) that are accepted (computed) by some DPTM or NPTM, respectively, with some oracle from $\mathcal{C}$.

2.3.1 The Polynomial Hierarchy

The polynomial hierarchy was introduced by Meyer and Stockmeyer [MS72, Sto77] as a tool for classifying computational problems. Its importance in complexity theory has grown since. The polynomial hierarchy, PH, is built inductively on P.

Definition 2.3.1 [MS72, Sto77]

1. $\Delta^{\mathrm{p}}_0 = \Sigma^{\mathrm{p}}_0 = \Pi^{\mathrm{p}}_0 = \mathrm{P}$.

2. For $k \geq 1$, $\Delta^{\mathrm{p}}_k = \mathrm{P}^{\Sigma^{\mathrm{p}}_{k-1}}$, $\Sigma^{\mathrm{p}}_k = \mathrm{NP}^{\Sigma^{\mathrm{p}}_{k-1}}$, and $\Pi^{\mathrm{p}}_k = \mathrm{co}\Sigma^{\mathrm{p}}_k$.

3. The polynomial hierarchy PH is defined by

$$\mathrm{PH} = \bigcup_{k \in \mathbb{N}} \Sigma^{\mathrm{p}}_k,$$

while its kth level consists of the classes Δ^{p}_k, Σ^{p}_k, and Π^{p}_k.

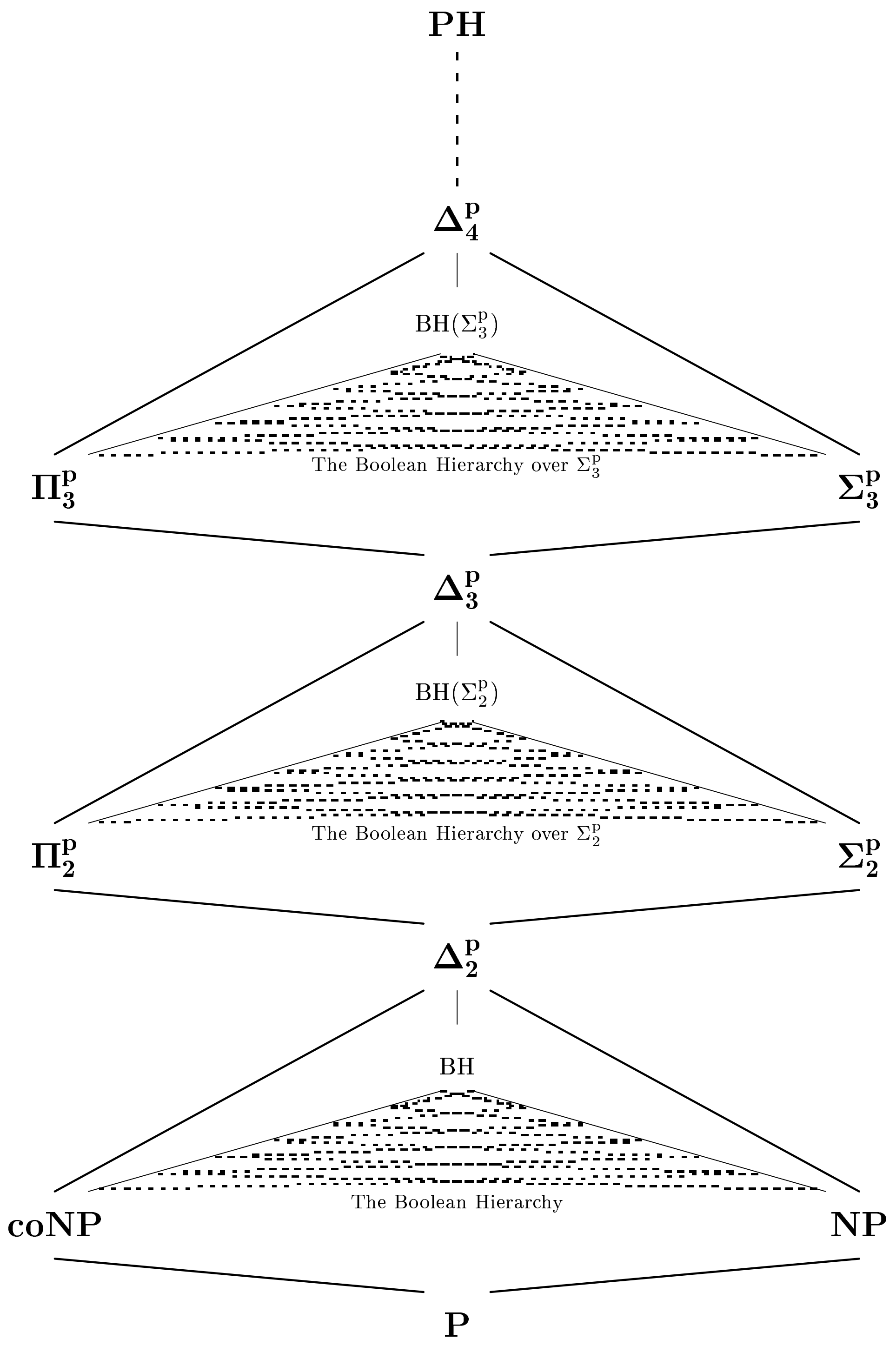


Figure 2.1: The Polynomial Hierarchy and its Levels

So for instance, $\Delta_1^{\mathrm{p}} = \mathrm{P}$, $\Sigma_1^{\mathrm{p}} = \mathrm{NP}$, $\Delta_2^{\mathrm{p}} = \mathrm{P}^{\mathrm{NP}}$, and $\Pi_2^{\mathrm{p}} = \mathrm{coNP}^{\mathrm{NP}}$. As it is standard, the term polynomial hierarchy will be used simultaneously for the complexity class PH and the hierarchy formed by the classes Σ_k^{p}, Π_k^{p}, and Δ_k^{p}, $k \geq 1$.

The Σ_k^{p} and Π_k^{p} classes of the polynomial hierarchy can be characterized in terms of k alternating operators $\exists$ and $\forall$ [MS73].

$$\Sigma_k^{\mathrm{p}} = \underbrace{\exists \cdot \forall \cdot \exists \cdots \mathcal{Q}}_{k \text{ alternating operators}} \cdot \mathrm{P},$$

where $\mathcal{Q} = \exists$ if k is odd and $\mathcal{Q} = \forall$ if k is even. Similarly,

$$\Pi_k^{\mathrm{p}} = \underbrace{\forall \cdot \exists \cdot \forall \cdots \mathcal{Q}}_{k \text{ alternating operators}} \cdot \mathrm{P},$$

where $\mathcal{Q} = \forall$ if k is odd and $\mathcal{Q} = \exists$ if k is even.

The classes of the polynomial hierarchy are all closed under many-one reductions and contain many-one complete sets. If $L_{\Sigma_k^{\mathrm{p}}}$ and $L_{\Sigma_{k-1}^{\mathrm{p}}}$, $k \geq 1$, are complete languages for Σ_k^{p} and $\Sigma_{k-1}^{\mathrm{p}}$, respectively, one can without loss of generality assume that

$$L_{\Sigma_k^{\mathrm{p}}} = \{x \mid (\exists y : |y| \leq p(|x|))[\langle x, y\rangle \notin L_{\Sigma_{k-1}^{\mathrm{p}}}\}$$

for some polynomial p. One can even assume that $p(n) = n$ for all $n \in \mathbb{N}$.

Though the question of whether the polynomial hierarchy collapses or not is still open many conditions are known under which the polynomial hierarchy does collapse. In particular, the polynomial hierarchy is known to possess the upward collapse property [Sto77]. For every $k \geq 1$,

1. $\Sigma_k^{\mathrm{p}} = \Pi_k^{\mathrm{p}} \implies \mathrm{PH} = \Sigma_k^{\mathrm{p}}$.
2. $\Sigma_k^{\mathrm{p}} = \Sigma_{k+1}^{\mathrm{p}} \implies \mathrm{PH} = \Sigma_k^{\mathrm{p}}$.
3. $\Delta_k^{\mathrm{p}} = \Sigma_k^{\mathrm{p}} \implies \mathrm{PH} = \Sigma_k^{\mathrm{p}}$.

It follows immediately from the definition of many-one reductions that $L \in \Sigma_k^{\mathrm{p}}$ for some Π_k^{p} complete language L implies $\Sigma_k^{\mathrm{p}} = \Pi_k^{\mathrm{p}}$ and thus $\mathrm{PH} = \Sigma_k^{\mathrm{p}}$. This fact will be exploited in Chapters 3 and 4.

The extent of the collapse of the polynomial hierarchy has been studied intensely for the case that the boolean hierarchy collapses (see Chapter 3).

Much more can be said about the polynomial hierarchy. The rather sparse selection of results presented above has been made with respect to the topics covered in this thesis. For more results and references we refer the reader to any text book on complexity theory, for instance, [BDG88, BDG90, WW86, Pap94].

2.3.2 The Boolean Hierarchy

The structure of complexity classes inside Δ_2^{p} has received much attention. Three hierarchies inside Δ_2^{p} are of particular interest for our work, the boolean (or difference) hierarchy, the bounded-query hierarchy, and the bounded-truth-table hierarchy.

The notion of the boolean hierarchy over some class $\mathcal{C}$ being closed under union and intersection in concept can be found in the work of Hausdorff [Hau14]. It has been introduced into complexity theory via the boolean hierarchy over NP independently by a number of authors with a variety of definitions [Wec85, CH86, KSW87, CGH$^+$88, CGH$^+$89]. We use the definition based on nested differences of sets. Though for an arbitrary complexity class $\mathcal{C}$ the notions of the difference hierarchy and the boolean hierarchy over $\mathcal{C}$ are not equivalent (for instance, see [HR97]), for $\mathcal{C} = \Sigma_k^{\mathrm{p}}$, $k \geq 1$, they are [KSW87, CGH$^+$88].

Definition 2.3.2 For all $k \geq 1$,

1. $\mathrm{DIFF}_1(\Sigma_k^{\mathrm{p}}) = \Sigma_k^{\mathrm{p}}$.

2. For $m \geq 1$, $\mathrm{DIFF}_{m+1}(\Sigma_k^{\mathrm{p}}) = \Sigma_k^{\mathrm{p}} \boldsymbol{-} \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$.

3. The boolean or difference hierarchy over Σ_k^{p} is defined as

$$\mathrm{BH}(\Sigma_k^{\mathrm{p}}) = \bigcup_{m \geq 1} \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}).$$

   $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$ and $\mathrm{coDIFF}_m(\Sigma_k^{\mathrm{p}})$ form its mth level.

We will refer to the boolean hierarchy over NP as the boolean hierarchy and use the classical notation for it, that is $\mathrm{BH(NP)} = \mathrm{BH}$ and $\mathrm{DIFF}_m(\mathrm{NP}) = \mathrm{BH}_m$. So for instance, BH_2 is exactly the class DP [PY84].

The boolean hierarchy (over NP) is a well studied object, a few papers shall be mentioned, [Wec85, CH86, KSW87, CGH$^+$88, CGH$^+$89, Wag90, Bei91]. Many results of the boolean hierarchy immediately carry over to the boolean hierarchy over Σ_k^{p}, $k \geq 1$. It is known that the levels of the boolean hierarchy over some Σ_k^{p} are closed under many-one reductions and contain many-one complete sets. Furthermore, for a set L, $L \in \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$ if and only if there exist sets $L_1, L_2, \ldots, L_m \in \Sigma_k^{\mathrm{p}}$ such that $L = L_1 - (L_2 - (\cdots - (L_{m-1} - L_m) \cdots))$. One can without loss of generality even assume $L_1 \supseteq L_2 \supseteq \cdots \supseteq L_m$ [CGH$^+$88].

The inclusion structure of the boolean hierarchy and the relationships between the boolean hierarchy and the bounded-truth-table hierarchy are illustrated in Figure 2.2.

A number of techniques have been developed while studying boolean hierarchies. Two of them will be heavily exploited in this thesis. The mind change technique, developed by Wagner [Wag79] and later applied to complexity theory [Wec85, Wag90, Bei91], is a tool particularly well suited to study inclusion relations with respect to boolean hierarchies. The mind change technique (for an example see Chapter 5) will have applications in Chapter 3 when linking boolean and polynomial hierarchy collapses as well as in Chapter 5 while characterizing query order classes. The easy-hard technique has been developed by

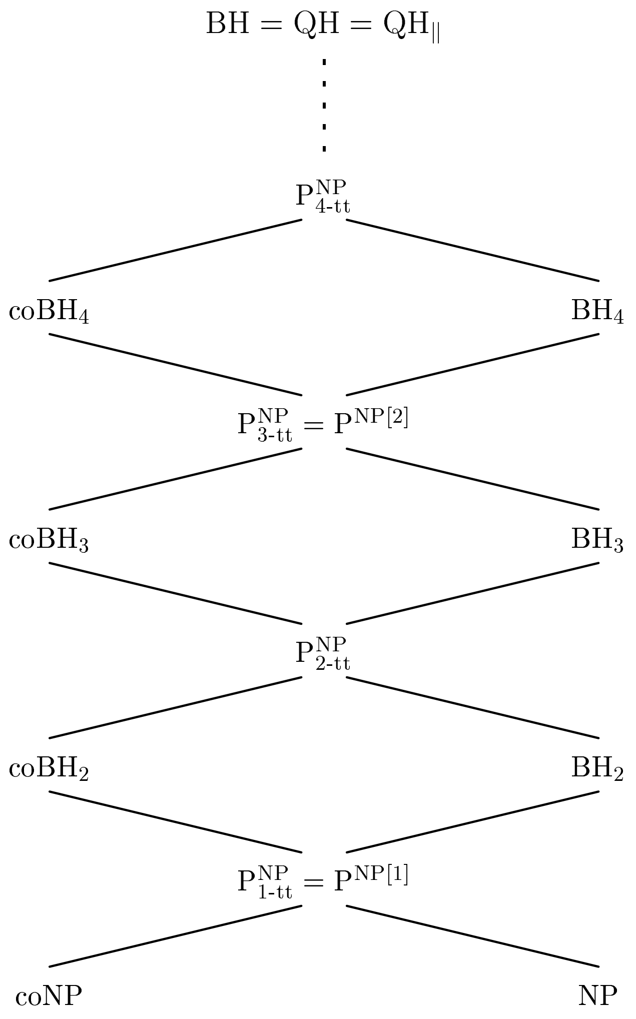

$\mathrm{BH} = \mathrm{QH} = \mathrm{QH}_{\|}$
$\mathrm{P}^{\mathrm{NP}}_{4\text{-tt}}$
coBH_4
BH_4
$\mathrm{P}^{\mathrm{NP}}_{3\text{-tt}} = \mathrm{P}^{\mathrm{NP}[2]}$
coBH_3
BH_3
$\mathrm{P}^{\mathrm{NP}}_{2\text{-tt}}$
coBH_2
BH_2
$\mathrm{P}^{\mathrm{NP}}_{1\text{-tt}} = \mathrm{P}^{\mathrm{NP}[1]}$
coNP
NP

Figure 2.2: The Boolean, Bounded-Query, and Bounded-Truth-Table Hierarchies

Kadin [Kad88] to prove a collapse of the polynomial hierarchy from the assumption that the boolean hierarchy collapses. The easy-hard technique has turned into a key tool for proving collapses of complexity classes induced by a collapse of boolean or generalized boolean hierarchies [Wag87, Wag89, CK96, BCO93, HHH99, BF98, HHH97b]. This interesting development will in detail be discussed in Chapters 3 and 4.

We mention that it is well known that the boolean hierarchy over Σ_k^{p}, $k \geq 1$, possesses the upward collapse property. In particular, for all $m, k \geq 1$,

1. $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_k^{\mathrm{p}}) \implies \mathrm{BH}(\Sigma_k^{\mathrm{p}}) = \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$.
2. $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) = \mathrm{DIFF}_{m+1}(\Sigma_k^{\mathrm{p}}) \implies \mathrm{BH}(\Sigma_k^{\mathrm{p}}) = \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$.

Boolean hierarchies over other classes than Σ_k^{p} have also been investigated [GNW90, BJY90, BCO93, HR97, HW97b].

2.3.3 The Bounded-Truth-Table and the Bounded-Query Hierarchies

Starting from the definition of $\Delta_{k+1}^{\mathrm{p}} = \mathrm{P}^{\Sigma_k^{\mathrm{p}}}$ itself researchers have studied limitations placed on the way an oracle can be accessed. Limitations being used are bounding the total number of queries or requiring all queries being made at once without knowing the answer to any query. First results in that direction have been obtained in [Bei91, Wag90] and much more has been learned since.

Definition 2.3.3 1. $\mathrm{P}^{\mathcal{C}[j]}$ denotes the set of all languages accepted by some DPTM making on every input at most j queries to an oracle from $\mathcal{C}$.

2. The bounded-query hierarchy over $\mathcal{C}$, $\mathrm{QH}(\mathcal{C}) = \bigcup_{j\geq 1} \mathrm{P}^{\mathcal{C}[j]}$.

3. $\mathrm{P}_{j\text{-tt}}^{\mathcal{C}}$ denotes the set of all languages accepted by some DPTM making on every input at most j parallel queries (all queries have to be generated before asking one of them to the oracle) to an oracle from $\mathcal{C}$.

4. The bounded-truth-table hierarchy over $\mathcal{C}$, $\mathrm{QH}_{||}(\mathcal{C}) = \bigcup_{j\geq 1} \mathrm{P}_{j\text{-tt}}^{\mathcal{C}}$.

Since the the bounded-query hierarchy and the bounded-truth-table hierarchy over NP where the first those hierarchies studied we will henceforth refer to them simply as the bounded-query hierarchy and the bounded-truth-table hierarchy.

For every $k \geq 1$, the levels of the bounded-query hierarchy over Σ_k^{p} and the levels of the bounded-truth-table hierarchy over Σ_k^{p} are exactly the bounded-Turing and bounded-truth-table reducibility closures of Σ_k^{p}, respectively. More precisely, for all $m, k \geq 1$,

1. $\mathrm{P}^{\Sigma_k^{\mathrm{p}}[m]} = \mathrm{R}_{m\text{-T}}^{\mathrm{p}}(\Sigma_k^{\mathrm{p}})$.

2. $\mathrm{P}_{m\text{-tt}}^{\Sigma_k^{\mathrm{p}}} = \mathrm{R}_{m\text{-tt}}^{\mathrm{p}}(\Sigma_k^{\mathrm{p}})$.

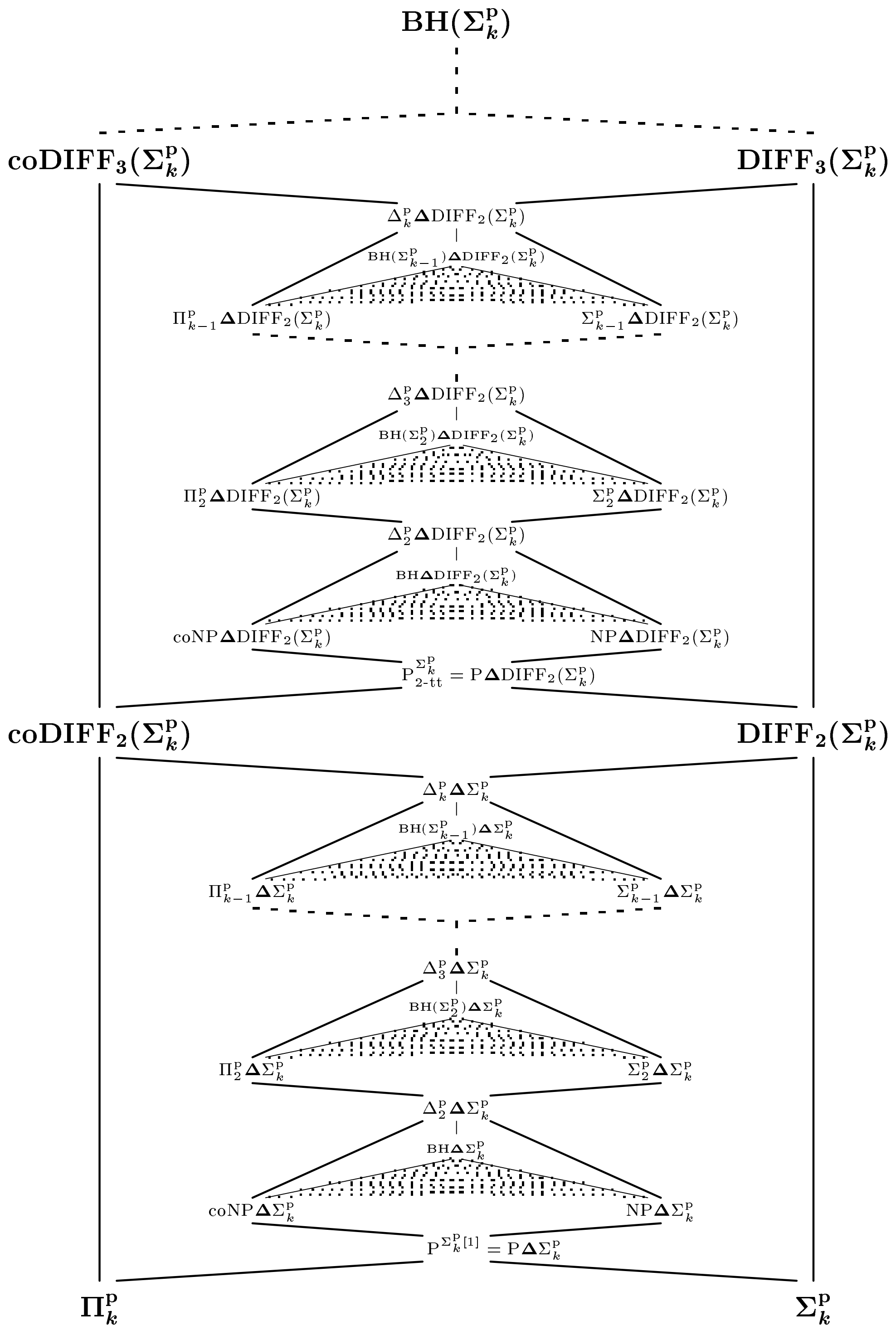
The levels of the boolean hierarchy, the bounded-query hierarchy, and the bounded-truth-table hierarchy over Σ_k^{p}, $k \geq 1$, intertwine. The following relations are known [KSW87, Wag90, Bei91] (see also Figure 2.2). For all $m, k \geq 1$,

1. $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) \cup \mathrm{coDIFF}_m(\Sigma_k^{\mathrm{p}}) \subseteq \mathrm{P}_{m\text{-tt}}^{\Sigma_k^{\mathrm{p}}} \subseteq \mathrm{DIFF}_{m+1}(\Sigma_k^{\mathrm{p}}) \cap \mathrm{coDIFF}_{m+1}(\Sigma_k^{\mathrm{p}})$.

2. $\mathrm{P}^{\Sigma_k^{\mathrm{p}}[m]} = \mathrm{P}_{2^m-1\text{-tt}}^{\Sigma_k^{\mathrm{p}}}$.

3. $\mathrm{BH}(\Sigma_k^{\mathrm{p}}) = \mathrm{QH}(\Sigma_k^{\mathrm{p}}) = \mathrm{QH}_{||}(\Sigma_k^{\mathrm{p}})$.

The above properties ensure that the collapse of one of the three hierarchies implies also a collapse of the other two.



$\mathbf{BH}(\Sigma_k^p)$
$\mathbf{coDIFF_3}(\Sigma_k^p)$
$\mathbf{DIFF_3}(\Sigma_k^p)$
$\Delta_k^p \mathbf{\Delta}\mathrm{DIFF}_2(\Sigma_k^p)$
$\mathrm{BH}(\Sigma_{k-1}^p)\mathbf{\Delta}\mathrm{DIFF}_2(\Sigma_k^p)$
$\Pi_{k-1}^p\mathbf{\Delta}\mathrm{DIFF}_2(\Sigma_k^p)$
$\Sigma_{k-1}^p\mathbf{\Delta}\mathrm{DIFF}_2(\Sigma_k^p)$
$\Delta_3^p \mathbf{\Delta}\mathrm{DIFF}_2(\Sigma_k^p)$
$\mathrm{BH}(\Sigma_2^p)\mathbf{\Delta}\mathrm{DIFF}_2(\Sigma_k^p)$
$\Pi_2^p\mathbf{\Delta}\mathrm{DIFF}_2(\Sigma_k^p)$
$\Sigma_2^p\mathbf{\Delta}\mathrm{DIFF}_2(\Sigma_k^p)$
$\Delta_2^p \mathbf{\Delta}\mathrm{DIFF}_2(\Sigma_k^p)$
$\mathrm{BH}\mathbf{\Delta}\mathrm{DIFF}_2(\Sigma_k^p)$
$\mathrm{coNP}\mathbf{\Delta}\mathrm{DIFF}_2(\Sigma_k^p)$
$\mathrm{NP}\mathbf{\Delta}\mathrm{DIFF}_2(\Sigma_k^p)$
$\mathrm{P}_{2\text{-tt}}^{\Sigma_k^p} = \mathrm{P}\mathbf{\Delta}\mathrm{DIFF}_2(\Sigma_k^p)$
$\mathbf{coDIFF_2}(\Sigma_k^p)$
$\mathbf{DIFF_2}(\Sigma_k^p)$
$\Delta_k^p\mathbf{\Delta}\Sigma_k^p$
$\mathrm{BH}(\Sigma_{k-1}^p)\mathbf{\Delta}\Sigma_k^p$
$\Pi_{k-1}^p\mathbf{\Delta}\Sigma_k^p$
$\Sigma_{k-1}^p\mathbf{\Delta}\Sigma_k^p$
$\Delta_3^p\mathbf{\Delta}\Sigma_k^p$
$\mathrm{BH}(\Sigma_2^p)\mathbf{\Delta}\Sigma_k^p$
$\Pi_2^p\mathbf{\Delta}\Sigma_k^p$
$\Sigma_2^p\mathbf{\Delta}\Sigma_k^p$
$\Delta_2^p\mathbf{\Delta}\Sigma_k^p$
$\mathrm{BH}\mathbf{\Delta}\Sigma_k^p$
$\mathrm{coNP}\mathbf{\Delta}\Sigma_k^p$
$\mathrm{NP}\mathbf{\Delta}\Sigma_k^p$
$\mathrm{P}^{\Sigma_k^p[1]} = \mathrm{P}\mathbf{\Delta}\Sigma_k^p$
$\mathbf{\Pi_k^p}$
$\mathbf{\Sigma_k^p}$

Figure 2.3: The Boolean Hierarchy over Σ_k^p, $k > 1$, and its Refined Levels

2.3.4 A Refinement of the Boolean Hierarchy over Σ_k^{p}, $k > 1$

The operator $\mathbf{\Delta}$ allows to refine the levels of the boolean hierarchy over Σ_k^{p}, $k > 1$ (see Figure 2.3).

In particular, we have for every $m \geq 1$ and every pair of complexity classes $\mathcal{C}$ and $\mathcal{D}$ such that $\mathrm{P} \subseteq \mathcal{C} \subseteq \mathcal{D} \subseteq \Sigma_k^{\mathrm{p}}$,

$$\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) \subseteq \mathrm{P}_{m\text{-tt}}^{\Sigma_k^{\mathrm{p}}} \subseteq \mathcal{C}\mathbf{\Delta}\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) \subseteq \mathcal{D}\mathbf{\Delta}\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) \subseteq \mathrm{DIFF}_{m+1}(\Sigma_k^{\mathrm{p}}).$$

This refinement, in a more general form, has first been studied by Selivanov [Sel94a, Sel95]. Selivanov extended the concept of the boolean hierarchy away from being based on one single complexity class to being based on a family of complexity classes. It was observed by Selivanov [Sel94a] and Wagner [Wag98] that the refinement of the boolean hierarchy over Σ_k^{p}, in particular the inclusion structure of the classes shown in Figure 2.3, is strict unless the polynomial hierarchy collapses.

2.4 Query Order

In Chapter 5 we study query order in computational complexity. Since query order has never been studied before we have to define the basic concepts and fix appropriate notations.

Definition 2.4.1 Let $\mathcal{C}$ and $\mathcal{D}$ be complexity classes.

1. $\mathrm{P}^{\mathcal{C}:\mathcal{D}}$ denotes the set of languages that are accepted by some DPTM making on every input at most one query to some oracle from $\mathcal{C}$ *followed* by at most one query to some oracle from $\mathcal{D}$.

2. $\mathrm{P}^{(\mathcal{C},\mathcal{D})}$ denotes the set of languages that are accepted by some DPTM making on every input at most two parallel queries, at most one query to some oracle from $\mathcal{C}$ and at most one query to some oracle from $\mathcal{D}$.

The above defined classes are the basic and most natural query order classes, at most one query to each of the two different oracles. We denote the machines that accept the languages of the above defined query order classes in total analogy to the definition of the classes itself, for instance, $M^{A:B}$ denotes a DPTM that on every input makes at most one query to oracle A followed by at most one query to oracle B.

In Section 5.2 we study query order classes of the form $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$. Some results regarding query order classes $\mathrm{P}^{\Sigma_i^{\mathrm{p}}:\Sigma_\ell^{\mathrm{p}}}$ are stated in Section 5.3.

Two already existing notions are somewhat related to these basic query order classes. First, note that $\mathcal{C}\mathbf{\Delta}\mathcal{D} \subseteq \mathrm{P}^{(\mathcal{C},\mathcal{D})}$. In particular, $\Sigma_j^{\mathrm{p}}\mathbf{\Delta}\Sigma_k^{\mathrm{p}}$, $j \neq k$, is a class almost as powerful as $\mathrm{P}^{(\Sigma_j^{\mathrm{p}},\Sigma_k^{\mathrm{p}})}$ being able to handle 15 of the 16 possible two-ary boolean functions (truth-tables). But an equality $\Sigma_j^{\mathrm{p}}\mathbf{\Delta}\Sigma_k^{\mathrm{p}} = \mathrm{P}^{(\Sigma_j^{\mathrm{p}},\Sigma_k^{\mathrm{p}})}$ is very unlikely since it would imply a collapse of the polynomial hierarchy [Sel94a] (see also [Wag98]). The connection between the "$\mathbf{\Delta}$ classes" which in a more general form appeared already in the work of Selivanov [Sel94a, Sel95] and the query order classes is in detail studied in [Wag98]. The second notion related to query order is based on a generalization of the "advice classes" notion of Karp and Lipton [KL80]. For any function class $\mathcal{F}$ and any complexity class $\mathcal{D}$ define $\mathcal{D}//\mathcal{F} = \{L \mid (\exists D \in \mathcal{D})(\exists f \in \mathcal{F})(\forall x \in \Sigma^*)[x \in L \iff \langle x, f(x)\rangle \in D]\}$ [KT94]. If $\mathcal{F}$ is the set of characteristic functions from all languages of a complexity class $\mathcal{C}$, then $\mathcal{D}//\mathcal{F} = \mathrm{P}^{\mathcal{C}:\mathcal{D}+}$ where $\mathrm{P}^{\mathcal{C}:\mathcal{D}+}$ denotes the set of all languages accepted by some DPTM making on every input one query to an oracle from $\mathcal{C}$ followed by one query to an oracle from $\mathcal{D}$ and having the additional property that it accepts if and only if the $\mathcal{D}$ oracle query receives the answer "yes." Classes of the form $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k+}$ have been studied in [ABT96]. There is a nontrivial relationship between the classes $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k+}$ and our query order classes $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ (see Section 5.2).

The following more general query order classes have also been studied [HHW99, BC, Wag98].

Definition 2.4.2 Let $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{E}$ be complexity classes.

1. $\mathrm{P}^{\mathcal{C}:\mathcal{D},\mathcal{E}}$ denotes the class of languages accepted by deterministic polynomial-time machines making one query to a $\mathcal{C}$ oracle followed, in case of a no answer to this first query, by one query to an oracle from $\mathcal{D}$ and, in case of a yes answer to the first query, by one query to an oracle from $\mathcal{E}$.

2. By $\mathrm{P}^{\mathcal{C}:\mathcal{D}:\mathcal{E}}$ we denote the set of all languages accepted by some DPTM making on every input at most one query to a $\mathcal{C}$ oracle followed by at most one query to an oracle from $\mathcal{D}$ followed by at most one query to an oracle from $\mathcal{E}$.

3. $\mathrm{P}^{\mathcal{C}:\mathcal{D}}_{j,k\text{-tt}}$ denotes the set of languages that are accepted by some DPTM making on every input one round of at most j parallel queries to some oracle from $\mathcal{C}$ *followed* by one round of at most k parallel queries to some oracle from $\mathcal{D}$.

4. $\mathrm{P}^{(\mathcal{C},\mathcal{D})}_{j,k\text{-tt}}$ denotes the set of languages that are accepted by some DPTM making on every input at most $j+k$ parallel queries, at most j of those $j+k$ queries to some oracle from $\mathcal{C}$ and at most k of those $j+k$ queries to some oracle from $\mathcal{D}$.

Classes of the form $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k,\mathrm{BH}_l}$ are studied in Section 5.2. Results on classes of the form $\mathrm{P}^{\Sigma_i^p:\Sigma_\ell^p}_{j,k\text{-tt}}$ can be found in Section 5.3.

2.5 Miscellaneous

For every $k \geq 1$, the k-dimensional hypercube is defined to be the graph with vertex set $\{(a_1, a_2, \ldots, a_k) \mid (\forall i : 1 \leq i \leq k)[a_i \in \{0,1\}]\}$ such that two vertices are adjacent (connected by an edge) if and only if they differ in exactly one position. As is standard in graph theory [BM76], a path in the hypercube is a sequence of distinct vertices such that every pair of consecutive vertices is joined by an edge. For the k-dimensional hypercube, the ith unit vector is the k tuple $(\underbrace{0,\ldots,0}_{k-i},1,\underbrace{0,\ldots,0}_{i-1})$. Vectors, especially unit vectors, are added by position wise addition.

Chapter 3

The Boolean and Polynomial Hierarchies Connection

3.1 Introduction

Does the polynomial hierarchy collapse if the boolean hierarchy collapses? This question arises naturally when studying the boolean hierarchy. Recall that the boolean hierarchy is a refinement of the polynomial hierarchy between the classes P and Δ_2^{p}. An affirmative answer to the above question would provide yet another strong hint (aside from relativized separation) that the boolean hierarchy is infinite since it is widely believed that the polynomial hierarchy does not collapse. Furthermore, an exact analysis to what level the polynomial hierarchy collapses if the boolean hierarchy collapses at its mth level will certainly shed light on the relationship between those two hierarchies.

The first result linking a collapse of the boolean hierarchy to a collapse of the polynomial hierarchy has been obtained by Kadin [Kad88] about 10 years ago. Kadin showed that a collapse of the boolean hierarchy at level m implies a collapse of the polynomial hierarchy to Σ_3^{p}. People have tried since to improve the induced collapse of the polynomial hierarchy. In fact, up to now Kadin's result has been improved four times, by Wagner [Wag87, Wag89], Chang and Kadin [CK96], and Beigel, Chang, and Ogihara [BCO93]. The best known result today allows to conclude a collapse of the polynomial hierarchy to a level just inside $\mathrm{DIFF}_m(\Sigma_2^{\mathrm{p}})$ [BCO93].

In this section we will pursue the question of to what level the polynomial hierarchy collapses if the boolean hierarchy collapses at level m. The goal of this section being closely reflected by its structure is twofold. On one hand we will review the work, results and proof techniques of the above mentioned five papers [Kad88, Wag87, Wag89, CK96, BCO93] in the overview-like Section 3.2. We provide a compact historic outline of this interesting line of research together with a detailed analysis of the evolution of the easy-hard technique which, introduced by Kadin [Kad88], has led to increasingly stronger results in the before mentioned papers. The second goal of this chapter is to prove a deeper collapse of the polynomial hierarchy which is done in Section 3.3. The proof of the main theorem of this chapter, Theorem 3.3.1, is based on a careful analysis of a proof given in [BCO93] together with a double application of one of its key ideas. We show that for all $m \geq 2$, if $\mathrm{BH}_m = \mathrm{coBH}_m$ then the polynomial hierarchy collapses to $\mathrm{BH}_m \boldsymbol{\Delta} \mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$. If $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_k^{\mathrm{p}})$, $k \geq 1$, analogous claims do also hold. Theorem 3.3.1 sheds new light on a question asked by Chang and Kadin [CK96]. Together with the downward collapse results from Chapter 4, this allows to conclude a deeper collapse of the polynomial hierarchy from the assumption that the bounded-truth-table hierarchy over Σ_k^{p}, $k > 1$, collapses.

All theorems of this section are examples of upward collapse results, a collapse of the boolean hierarchy (sandwiched between P and Δ_2^{p}) implies a collapse of the polynomial hierarchy to a level above Σ_2^{p}. Recently, the easy-hard technique (modified as needed) has also been crucial in proving downward collapse results [HHH99, BF98, HHH97b]. For an overview and detailed analysis of this line of research see Chapter 4.

3.2 A Review

Kadin [Kad88] showed that a collapse of the boolean hierarchy implies a collapse of the polynomial hierarchy. He invented the easy-hard technique, a key ingredient in his proof and all those that are built upon it and establish stronger results. Since this chapter studies the collapse of the polynomial hierarchy induced by a collapse of the boolean hierarchy the first section consists of an overview-like analysis of the relevant previous work. In particular, we study in detail the results and proof techniques of the following five papers:

1. J. Kadin. The Polynomial Time Hierarchy Collapses if the Boolean Hierarchy Collapses. *SIAM Journal on Computing,* 17(6):1263-1282, 1988. Erratum appears in the same journal, 20(2):404.

2. K. Wagner. Number-of-Query Hierarchies. Technical Report 158, Institut für Mathematik, Universität Augsburg, Augsburg, Germany, October 1987.

3. K. Wagner. Number-of-Query Hierarchies. Technical Report 4, Institut für Informatik, Universität Würzburg, Würzburg, Germany, February 1989.

4. R. Chang and J. Kadin. The Boolean Hierarchy and the Polynomial Hierarchy: A Closer Connection. *SIAM Journal on Computing,* 25(2):340-354, 1996.

5. R. Beigel, R. Chang, and M. Ogiwara. A Relationship Between Difference Hierarchies and Relativized Polynomial Hierarchies. *Mathematical Systems Theory,* 26(3):293-310, 1993.

We will concentrate on the improvements and the technical contributions each paper made with respect to previous work in this line of research. After a short overview over the main results obtained and an informal discussion of the technical advances made we rigorously prove a special case of each of the main theorems in its original version.

3.2.1 Previous Results

In the following we list the main results obtained together with pointers to the earliest and most recent versions of the five above mentioned papers.

Recall that due to $\mathrm{BH}_1 = \mathrm{NP}$ and the upward collapse property of the polynomial hierarchy we have that $\mathrm{BH}_1 = \mathrm{coBH}_1 \implies \mathrm{PH} = \mathrm{NP}$.

Kadin 1987 [Kad87, Kad88] Kadin started a line of research that studies the question of to what level the polynomial hierarchy collapses if the boolean hierarchy collapses. He showed that a collapse of the boolean hierarchy at level m implies a collapse of the polynomial hierarchy to its third level, Σ_3^{p}.

Theorem 3.2.1 [Kad87, Kad88] For all $m \geq 2$,

$$\mathrm{BH}_m = \mathrm{coBH}_m \implies \mathrm{PH} = \Sigma_3^{\mathrm{p}}.$$

Wagner 1987 [Wag87] Wagner not just extended the result of Kadin to the unbounded boolean hierarchy, but also improved it significantly for the boolean hierarchy itself. Kadin's technique together with oracle replacement enabled Wagner to show that a collapse of the boolean hierarchy over NP at level m implies a collapse of the polynomial hierarchy to Δ_3^{p}.

Theorem 3.2.2 [Wag87] For all $m \geq 2$,

$$\mathrm{BH}_m = \mathrm{coBH}_m \implies \mathrm{PH} = \Delta_3^{\mathrm{p}}.$$

Wagner 1989 [Wag89] Wagner observed that a modified definition of hard strings yields an even stronger collapse of the polynomial hierarchy. In particular, he showed that a collapse of the boolean hierarchy at level m implies a collapse of the polynomial hierarchy to a level within Δ_3^{p}, namely, the boolean closure of Σ_2^{p}, $\mathrm{BH}(\Sigma_2^{\mathrm{p}})$.

Theorem 3.2.3 [Wag89] For all $m \geq 2$,

$$\mathrm{BH}_m = \mathrm{coBH}_m \implies \mathrm{PH} = \mathrm{BH}(\Sigma_2^{\mathrm{p}}).$$

Chang and Kadin 1989 [CK89, CK96] Chang and Kadin refined the method originally used by Kadin to further tighten the connection between the boolean hierarchy and the polynomial hierarchy. Unaware of Wagner's work they improved his results. They showed that a collapse of the boolean hierarchy at level m implies a collapse of the polynomial hierarchy to a level within the boolean closure of Σ_2^{p}, namely, the mth level of the boolean hierarchy over Σ_2^{p}.

Theorem 3.2.4 [CK89, CK96] For all $m \geq 2$,

$$\mathrm{BH}_m = \mathrm{coBH}_m \implies \mathrm{PH} = \mathrm{DIFF}_m(\Sigma_2^{\mathrm{p}}).$$

Beigel, Chang, and Ogihara 1991 [BCO91, BCO93] Beigel, Chang, and Ogihara, while picking up ideas developed by Wagner, were able to draw an even stronger conclusion. In particular, they showed that a collapse of the boolean hierarchy at level m implies a collapse of the polynomial hierarchy to a level within the mth level of the boolean hierarchy over Σ_2^p, namely, to $\left(\mathrm{P}^{\mathrm{NP}}_{m-1\text{-tt}}\right)^{\mathrm{NP}}$, the class of languages accepted by some deterministic polynomial-time machine making at most $m-1$ parallel queries to an $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^p$ oracle and an unlimited number of queries to an NP oracle.

Theorem 3.2.5 [BCO91, BCO93] For all $m \geq 2$,

$$\mathrm{BH}_m = \mathrm{coBH}_m \implies \mathrm{PH} = \left(\mathrm{P}^{\mathrm{NP}}_{m-1\text{-tt}}\right)^{\mathrm{NP}}.$$

3.2.2 The Development of the Easy-Hard Technique

As already mentioned, the easy-hard technique plays a crucial role in each of the above mentioned theorem's proofs. The term easy-hard originates from Kadin's observation that in case the boolean hierarchy collapses at level m the strings of any particular length n in a coNP complete language L divide into easy and hard strings. Hard strings are strings that allow to translate a collapse of the boolean hierarchy from level m to level $m-1$ in a restricted sense. Several hard strings eventually allow to reduce the coNP complete language L to an NP language. In contrast, if no hard strings at length n exist then all strings in L of length n are easy and this allows to directly reduce L to an NP language. Consequently, if we know whether there exist hard strings or not, and if, in case they exist, we are able to efficiently compute them, we can with their help reduce the coNP complete language L to an NP language and eventually collapse the polynomial hierarchy. This approach is central in each of the five papers studied in this section. The major difference among the five papers and the main reason for the difference in their results is the way in which the needed information about the hard strings (their existence and the strings itself) is obtained, and in which way this information is then exploited to collapse the polynomial hierarchy.

Kadin [Kad88] constructed a sparse set S such that S contains information about the existence of hard strings for any length and in case there exist hard strings for some length then a lexicographically extreme hard string for that length can be efficiently extracted from S. It is shown that $\mathrm{coNP} \subseteq \mathrm{NP}^S$, which by a result of Yap [Yap83] implies $\mathrm{PH} \subseteq \Sigma_3^p$.

Wagner used a quite different approach in his two papers, [Wag87, Wag89]. In both papers the polynomial hierarchy is collapsed directly (without constructing a sparse oracle) using oracle replacement and hard strings in form of advice. The main reason for the stronger result in his second paper is a modified definition of easy and hard strings. Thus, instead of hard strings giving a reduction for only the strings of one particular length (implying that one hard string for each length is needed when collapsing the polynomial hierarchy), Wagner's new definition yields that hard strings can give a reduction for all strings having a length below a particular threshold.

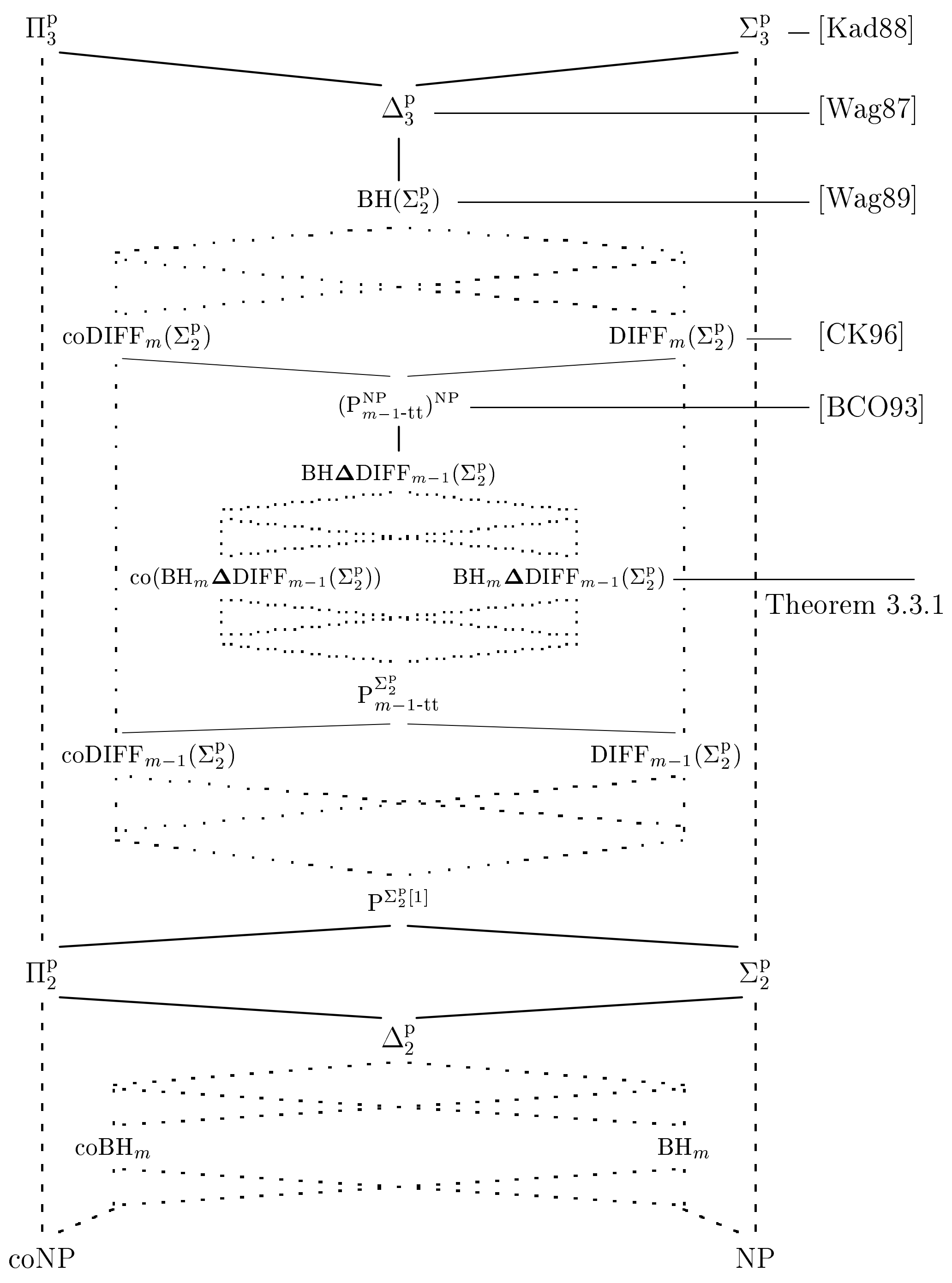


Figure 3.1: Results overview—a collapse $BH_m = coBH_m$, $m \geq 2$, implies a collapse of the polynomial hierarchy to the classes at the horizontal lines (as proven in the appropriate papers).

Chang and Kadin [CK96], independent of Wagner's work, also used the stronger notion of hardness. The observation that hard strings of larger length allow to efficiently gain information about the existence of hard strings at lower length together with an elegant application of the nested difference structure underlying the levels of the boolean hierarchy over Σ_2^p, leads to their final result.

Beigel, Chang, and Ogihara [BCO93] further improved the results of Chang and Kadin. [BCO93] follows the approach of Wagner, but with two major innovations. First, complete languages for the levels of the boolean hierarchy are used that do not force to distinguish between odd and even levels. Second, their proof exploits the mind change technique to efficiently check the existence of hard strings and their effect on the outcome of the reduction using those hard strings. The second innovation goes hand in hand with a modified argumentation for (conditionally) collapsing the polynomial hierarchy.

3.2.3 A Close Look at the Proofs

In the following we will prove for each of the Theorems 3.2.1, 3.2.2, 3.2.3, 3.2.4, and 3.2.5 a special case to illustrate the underlying proof technique and the innovations made with respect to previous work. We will start from the assumption DP = coDP–a collapse of the boolean hierarchy at its second level–and prove the collapse of the polynomial hierarchy obtained by the corresponding theorem as it was done in the original version.

For clarity of presentation we would like to make the following rather technical assumptions. Let s be a polynomial such that for all $x, y, z \in \Sigma^* \cup \{\#\}$, $|\langle x, y\rangle| \leq s(\max\{|x|, |y|\})$ and $|\langle x, y, z\rangle| \leq s(\max\{|x|, |y|, |z|\})$. Furthermore, let us agree on the following convention. Whenever we talk about polynomials in the remainder of this chapter let us assume that those polynomials are of the form $n^a + b$ for some integers $a, b > 0$. Since the polynomials involved in the upcoming proofs always play the role of a function bounding the running time of some Turing machine or the length of some variable and all complexity classes under consideration have certain required properties we can make this assumption without loss of generality. This convention has the advantage that a polynomial p now satisfies $p(n+1) > p(n) > n$ for all n, a condition we will need throughout this section's proofs.

Kadin 1987 [Kad87, Kad88]

Theorem 3.2.6 If DP = coDP then PH = Σ_3^p.

Proof: Recall that DP [PY84] is defined as DP = $\{L_1 - L_2 \mid L_1, L_2 \in \mathrm{NP}\}$.

A Suppose DP = coDP. Let L_{NP} be a many-one complete language for NP. It is not hard to verify that $L_{\mathrm{DP}} = \{\langle x, y\rangle \mid x \in L_{\mathrm{NP}} \wedge y \notin L_{\mathrm{NP}}\}$ is a many-one complete language for DP. According to our assumption, DP = coDP, there is a polynomial-time computable function h reducing L_{DP} to $\overline{L_{\mathrm{DP}}}$, i.e., for all $x_1, x_2 \in \Sigma^*$,

$$\langle x_1, x_2\rangle \in L_{\mathrm{DP}} \iff h(\langle x_1, x_2\rangle) \in \overline{L_{\mathrm{DP}}}.$$

Let h' and h'' be the polynomial-time computable functions such that for all $x_1, x_2 \in \Sigma^*$, $h(\langle x_1, x_2 \rangle) = \langle h'(\langle x_1, x_2 \rangle), h''(\langle x_1, x_2 \rangle) \rangle$. Hence,

$$x_1 \in L_{\mathrm{NP}} \wedge x_2 \notin L_{\mathrm{NP}} \iff h'(\langle x_1, x_2 \rangle) \notin L_{\mathrm{NP}} \vee h''(\langle x_1, x_2 \rangle) \in L_{\mathrm{NP}}.$$

The easy-hard method is based on the fact that h is a many-one reduction from a conjunction to a disjunction.

B The string x_2 is said to be *easy* if and only if $(\exists x_1 : |x_1| = |x_2|)[h''(\langle x_1, x_2 \rangle) \in L_{\mathrm{NP}}]$. Clearly, if x_2 is easy then $x_2 \notin L_{\mathrm{NP}}$. But note that checking whether a particular string is easy can be done with an NP algorithm.

x_2 is said to be *hard* if and only if $x_2 \notin L_{\mathrm{NP}}$ and $(\forall x_1 : |x_1| = |x_2|)[h''(\langle x_1, x_2 \rangle) \notin L_{\mathrm{NP}}]$. Hence, if x_2 is a hard string we have for all x_1, $|x_1| = |x_2|$,

$$x_1 \in L_{\mathrm{NP}} \iff h'(\langle x_1, x_2 \rangle) \notin L_{\mathrm{NP}}.$$

Note that the strings in $\overline{L_{\mathrm{NP}}}$ divide into easy and hard strings.

C Define the set $S' = \{\omega \,|\, \omega$ is the lexicographically smallest hard string of length $|\omega|\}$ and the set S of marked prefixes of S', $S = \{y\#^{|v|} \,|\, yv \in S'\}$. Note that S is sparse.

D *Claim D:* coNP $\subseteq$ (NP)^S.

We will prove the above claim by giving an (NP)^S algorithm for $\overline{L_{\mathrm{NP}}}$:

1. On input x, $|x| = n$, check whether $S^{=n}$ is empty or not. This can be done by querying $0\#^{n-1}$ and $1\#^{n-1}$. Obviously, $S^{=n} = \emptyset$ if and only if both queries are answered no.
2. If $S^{=n} = \emptyset$ then there exists no hard string of length n. Hence, $x \in \overline{L_{\mathrm{NP}}}$ if and only if x is easy. Thus, guess x_1, $|x_1| = n$, compute $h''(\langle x_1, x \rangle)$, and accept if and only if $h''(\langle x_1, x \rangle) \in L_{\mathrm{NP}}$.
3. If $S^{=n} \neq \emptyset$ then there exists a hard string of length n. Retrieve the only string not containing # (recall that this is the lexicographically smallest hard string of length n) from $S^{=n}$, call it ω, with adaptive queries to $S^{=n}$. Compute $h'(\langle x, \omega \rangle)$ and accept if and only if $h'(\langle x, \omega \rangle) \in L_{\mathrm{NP}}$.

According to **B**, this algorithm is correct.

E By a result of Yap [Yap83], coNP $\subseteq$ (NP)^S for a sparse set S implies $\Sigma_3^{\mathrm{p}} = \Pi_3^{\mathrm{p}}$ and hence PH $= \Sigma_3^{\mathrm{p}}$.

∎

Wagner 1987 [Wag87]

Theorem 3.2.7 If DP = coDP then PH = $\mathrm{P}^{\Sigma_2^{\mathrm{p}}}$.

Proof: **A and B** As in the proof of Theorem 3.2.6 (Kadin 1987).

C Let L_{NP}, $L_{\Sigma_2^{\mathrm{p}}}$, and $L_{\Sigma_3^{\mathrm{p}}}$ be many-one complete languages for NP, Σ_2^{p}, and Σ_3^{p}, respectively, and p_2, p_3 be polynomials such that

$$L_{\Sigma_2^{\mathrm{p}}} = \{x \mid (\exists y : |y| \le p_2(|x|))[\langle x, y\rangle \notin L_{\mathrm{NP}}]\}$$

and

$$L_{\Sigma_3^{\mathrm{p}}} = \{x \mid (\exists y : |y| \le p_3(|x|))[\langle x, y\rangle \notin L_{\Sigma_2^{\mathrm{p}}}]\}.$$

D For all $x \in \Sigma^*$ let

$$f(x) = \begin{cases} 1 & \text{if there exists a hard string of length } |x|, \\ 0 & \text{if there exists no hard string of length } |x|. \end{cases}$$

It is not hard to see that $f \in \mathrm{FP}^{\Sigma_2^{\mathrm{p}}[1]}$, where $\mathrm{FP}^{\Sigma_2^{\mathrm{p}}[j]}$ is defined similar to $\mathrm{P}^{\Sigma_2^{\mathrm{p}}[j]}$ with the modification that the base P machine computes a function instead of accepting a language. Note that $f(x) = f(y)$ if $|x| = |y|$.

We call $\langle 1^l, \#\rangle$ a hard pair if and only if $f(1^l) = 0$. $\langle 1^l, y\rangle$, $y \in \Sigma^*$, is called a hard pair if and only if y is a hard string of length l.

E One hard pair suffices to provide a reduction from $\overline{L_{\mathrm{NP}}}$ to an NP language.

Claim E: There exists a set $A \in$ NP such that for all $x \in \Sigma^*$, if $\langle 1^{|x|}, \omega\rangle$ is a hard pair then

$$x \notin L_{\mathrm{NP}} \iff \langle x, \omega\rangle \in A.$$

Let $x \in \Sigma^*$. Let $\langle 1^{|x|}, \omega\rangle$ be a hard pair, note that this implies $\omega \in \Sigma^* \cup \{\#\}$. Suppose $f(1^{|x|}) = 0$. Hence $\omega = \#$ and for every string y such that $|y| = |x|$, $y \notin L_{\mathrm{NP}}$ if and only if y is easy. This holds, in particular, for x itself. According to **B** we thus have

$$x \notin L_{\mathrm{NP}} \iff (\exists x_1 : |x_1| = |x|)[h''(\langle x_1, x\rangle) \in L_{\mathrm{NP}}].$$

Now suppose that $f(1^{|x|}) = 1$. Hence ω is a hard string of length $|x|$. According to **B** we obtain

$$x \notin L_{\mathrm{NP}} \iff h'(\langle x, \omega\rangle) \in L_{\mathrm{NP}}.$$

We define $A = \{\langle x, \omega\rangle \mid (\omega = \# \wedge (\exists x_1 : |x_1| = |x|)[h''(\langle x_1, x\rangle) \in L_{\mathrm{NP}}]) \vee (\omega \in \Sigma^* \wedge h'(\langle x, \omega\rangle) \in L_{\mathrm{NP}})\}$. It is not hard to verify that $A \in$ NP and that A satisfies Claim E.

F Applying Claim E, a series of hard pairs of growing length gives a reduction from $L_{\Sigma_2^{\mathrm{p}}}$ to an NP language.

Claim F: There exist a set $B \in \mathrm{NP}$ and a polynomial q such that for all $x \in \Sigma^*$, if for all $0 \le i \le q(|x|)$, $\langle 1^i, \omega_i\rangle$ is a hard pair then

$$x \in L_{\Sigma_2^{\mathrm{p}}} \iff \langle x, \omega_0, \omega_1, \ldots, \omega_{q(|x|)}\rangle \in B.$$

Let $x \in \Sigma^*$. By definition of $L_{\Sigma_2^{\mathrm{p}}}$ we have

$$x \in L_{\Sigma_2^{\mathrm{p}}} \iff (\exists y : |y| \le p_2(|x|))[\langle x, y\rangle \notin L_{\mathrm{NP}}].$$

According to Claim E there exists a set $A \in \mathrm{NP}$ such that for all $y \in \Sigma^*$, if $\langle 1^{|\langle x,y\rangle|}, \omega\rangle$ is a hard pair then

$$\langle x, y\rangle \notin L_{\mathrm{NP}} \iff \langle\langle x, y\rangle, \omega\rangle \in A.$$

Hence, if $\langle 1^i, \omega_i\rangle$ ($\omega_i \in \Sigma^* \cup \{\#\}$) is a hard pair for all $0 \le i \le s(p_2(|x|))$ (recall that s is a polynomial bounding the size of $\langle . \rangle$) then

$$x \in L_{\Sigma_2^{\mathrm{p}}} \iff (\exists y : |y| \le p_2(|x|))[\langle\langle x, y\rangle, \omega_{|\langle x,y\rangle|}\rangle \in A].$$

Let q be a polynomial such that $q(n) \ge s(p_2(n))$ for all n. Define

$$B = \{\langle x, \omega_0, \omega_1, \ldots, \omega_{q(|x|)}\rangle \mid (\exists y : |y| \le p_2(|x|))[\langle\langle x, y\rangle, \omega_{|\langle x,y\rangle|}\rangle \in A]\}$$

and note that $B \in \mathrm{NP}$. This proves Claim F.

G Taking the result of Claim F one step further, hard pairs of growing length provide a reduction from $L_{\Sigma_3^{\mathrm{p}}}$ to a Σ_2^{p} language.

Claim G: There exist a set $D \in \Sigma_2^{\mathrm{p}}$ and a polynomial p such that for all $x \in \Sigma^*$, if for all $0 \le i \le p(|x|)$, $\langle 1^i, \omega_i\rangle$ is a hard pair then

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff \langle x, \omega_0, \omega_1, \ldots, \omega_{p(|x|)}\rangle \in D.$$

The proof is similar to the proof of Claim F. Let $x \in \Sigma^*$. We have

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff (\exists y : |y| \le p_3(|x|))[\langle x, y\rangle \notin L_{\Sigma_2^{\mathrm{p}}}].$$

According to Claim F there exist a set $B \in \mathrm{NP}$ and a polynomial q such that for all y, $|y| \le p_3(|x|)$, if $\langle 1^i, \omega_i\rangle$ is a hard pair for all $0 \le i \le q(s(p_3(|x|)))$ then

$$\langle x, y\rangle \in L_{\Sigma_2^{\mathrm{p}}} \iff \langle\langle x, y\rangle, \omega_0, \omega_1, \ldots, \omega_{q(|\langle x,y\rangle|)}\rangle \in B,$$

and hence

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff (\exists y : |y| \le p_3(|x|))[\langle\langle x, y\rangle, \omega_0, \omega_1, \ldots, \omega_{q(|\langle x,y\rangle|)}\rangle \notin B].$$

Let p be a polynomial such that $p(n) \ge q(s(p_3(n)))$ for all n. Define

$$D = \{\langle x, \omega_0, \omega_1, \ldots, \omega_{p(|x|)}\rangle \mid (\exists y : |y| \le p_3(|x|))[\langle\langle x, y\rangle, \omega_0, \omega_1, \ldots, \omega_{q(|\langle x,y\rangle|)}\rangle \notin B]\}.$$

Clearly, $D \in \Sigma_2^{\mathrm{p}}$.

H Since the hard pairs needed in Claim G can be computed with queries to a Σ_2^p oracle we finally obtain

Claim H: $L_{\Sigma_3^p} \in \mathrm{P}^{\Sigma_2^p}$.

Let $D \in \Sigma_2^p$ and p be a polynomial, both as defined in **G**. We have the following $\mathrm{P}^{\Sigma_2^p}$ algorithm for $L_{\Sigma_3^p}$:

1. On input x, compute $f(1^0)$, $f(1^1)$, $f(1^2)$, ... , $f(1^{p(|x|)})$. This can be done with $p(|x|)$ parallel queries to a Σ_2^p oracle since $f \in \mathrm{FP}^{\Sigma_2^p[1]}$.
2. Query $\langle x, f(1^0), f(1^1), f(1^2), \ldots, f(1^{p(|x|)})\rangle$. The Σ_2^p oracle queried in this part of the $\mathrm{P}^{\Sigma_2^p}$ algorithm is described by the following algorithm:
   (a) On input $\langle x, y_0, y_1, \ldots, y_k\rangle$ verify that $k = p(|x|)$.
   (b) For all $0 \le i \le p(|x|)$ such that $y_i = 0$ set $\omega_i = \#$.
   (c) For all $0 \le i \le p(|x|)$ such that $y_i = 1$ guess ω_i, $|\omega_i| = i$, and verify that ω_i is a hard string. Recall that this verification can be done with a coNP algorithm. Continue if this verification succeeds, otherwise reject.
   (d) Accept if and only if $\langle x, \omega_0, \omega_1, \ldots, \omega_{p(|x|)}\rangle \in D$.
3. Accept (the input x) if and only if the query from part 2 returns yes.

It is not hard to verify that the above algorithm, in light of Claim G, proves Claim H Note that the oracles queried in parts 1 and 2 differ. Since Σ_2^p is closed under disjoint union this can easily be avoided by using the disjoint union of the oracles and modifying the oracle queries in such a way that they are made to the correct part of the disjoint union.

I Since $L_{\Sigma_3^p}$ is complete for Σ_3^p we conclude $\Sigma_3^p = \mathrm{P}^{\Sigma_2^p}$ and thus $\mathrm{PH} = \mathrm{P}^{\Sigma_2^p}$.

∎

Wagner 1989 [Wag89]

Theorem 3.2.8 If DP = coDP then $\mathrm{PH} = \mathrm{P}^{\Sigma_2^p[3]}$.

Proof: The major difference to the proof of Theorem 3.2.7 lies in **B** of the upcoming proof, a modified definition of easy and hard strings. Note that a straightforward adaption of the proof of Theorem 3.2.7 to this new definition would suffice to prove Theorem 3.2.8. However, Wagner used a slightly different approach and obtained stronger intermediate results than in [Wag87]. Though those stronger intermediate results do not lead to a better overall result here, they appear in the papers of Chang and Kadin [CK89, CK96] and Beigel, Chang, and Ogihara [BCO91, BCO93] again and play an important role there.

A As in the proof of Theorem 3.2.6 (Kadin 1987).

B Let l be an integer. The string x_2 is said to be *easy for length* l if and only if $|x_2| \leq l$ and $(\exists x_1 : |x_1| \leq l)[h''(\langle x_1, x_2 \rangle) \in L_{\mathrm{NP}}]$. Clearly, if x_2 is easy for length l then $x_2 \notin L_{\mathrm{NP}}$.

x_2 is said to be *hard for length* l if and only if $|x_2| \leq l$, $x_2 \notin L_{\mathrm{NP}}$, and $(\forall x_1 : |x_1| \leq l)[h''(\langle x_1, x_2 \rangle) \notin L_{\mathrm{NP}}]$. Hence, if x_2 is a hard string for length l we have for all x_1, $|x_1| \leq l$,

$$x_1 \in L_{\mathrm{NP}} \iff h'(\langle x_1, x_2 \rangle) \notin L_{\mathrm{NP}}.$$

Note that the strings in $(\overline{L_{\mathrm{NP}}})^{\leq l}$ divide into easy and hard strings for length l.

C As in the proof of Theorem 3.2.7 (Wagner 1987).

D For all $x \in \Sigma^*$ let

$$f(x) = \begin{cases} 1 & \text{if there exists a hard string } \textit{for} \text{ length } |x|, \\ 0 & \text{if there exists no hard string } \textit{for} \text{ length } |x|. \end{cases}$$

Note, $f \in \mathrm{FP}^{\Sigma_2^{\mathrm{p}}[1]}$ and $f(x) = f(y)$ if $|x| = |y|$.

We call $\langle 1^l, \# \rangle$ a hard pair if and only if $f(1^l) = 0$. $\langle 1^l, y \rangle$, $y \in \Sigma^*$, is called a hard pair if and only if y is a hard string for length l.

E Similar to **E** in the proof of Theorem 3.2.7 (Wagner 1987) (with the obvious adaptions due to the changed definition of easy and hard strings) one hard pair gives a reduction from $\overline{L_{\mathrm{NP}}}$ to an NP language.

Claim E: There exists a set $A \in \mathrm{NP}$ *such that for all* $x \in \Sigma^*$ *and all* $l \geq |x|$, *if* $\langle 1^l, \omega \rangle$ *is a hard pair then*

$$x \notin L_{\mathrm{NP}} \iff \langle x, 1^l, \omega \rangle \in A.$$

Let $x \in \Sigma^*$ and $l \geq |x|$. Suppose that $\langle 1^l, \omega \rangle$ is a hard pair, hence $\omega \in \Sigma^* \cup \{\#\}$.

If $f(1^l) = 0$ then $\omega = \#$ and for every string y, $|y| \leq l$, $y \in \overline{L_{\mathrm{NP}}}$ if and only if y is easy for length l. This holds, in particular, for x itself. According to **B** we thus have

$$x \notin L_{\mathrm{NP}} \iff (\exists x_1 : |x_1| \leq l)[h''(\langle x_1, x \rangle) \in L_{\mathrm{NP}}].$$

If $f(1^l) = 1$ then ω is a hard string for length l. According to **B** we obtain

$$x \notin L_{\mathrm{NP}} \iff h'(\langle x, \omega \rangle) \in L_{\mathrm{NP}}.$$

We define $A = \{\langle x, 1^l, \omega \rangle \mid (\omega = \# \wedge (\exists x_1 : |x_1| \leq l)[h''(\langle x_1, x \rangle) \in L_{\mathrm{NP}}]) \vee (\omega \in \Sigma^* \wedge h'(\langle x, \omega \rangle) \in L_{\mathrm{NP}})\}$. It is not hard to verify that $A \in \mathrm{NP}$. This completes the proof of Claim E.

F In contrast to **F** in the proof of Theorem 3.2.7 (Wagner 1987), the new definition of easy and hard strings yields that *one* hard pair for sufficiently large length suffices to reduce $L_{\Sigma_2^{\mathrm{p}}}$ to an NP language.

Claim F: There exist a set $B \in \mathrm{NP}$ and a polynomial q such that for all $x \in \Sigma^*$ and all $l \geq q(|x|)$, if $\langle 1^l, \omega\rangle$ is a hard pair then

$$x \in L_{\Sigma_2^{\mathrm{p}}} \iff \langle x, 1^l, \omega\rangle \in B.$$

Let $x \in \Sigma^*$. By definition of $L_{\Sigma_2^{\mathrm{p}}}$ we have

$$x \in L_{\Sigma_2^{\mathrm{p}}} \iff (\exists y : |y| \leq p_2(|x|))[\langle x, y\rangle \notin L_{\mathrm{NP}}].$$

According to Claim E there exists a set $A \in \mathrm{NP}$ such that if $\langle 1^l, \omega\rangle$, $l \geq s(p_2(|x|))$, is a hard pair then for all y, $|y| \leq p_2(|x|)$,

$$\langle x, y\rangle \notin L_{\mathrm{NP}} \iff \langle\langle x, y\rangle, 1^l, \omega\rangle \in A,$$

and hence

$$x \in L_{\Sigma_2^{\mathrm{p}}} \iff (\exists y : |y| \leq p_2(|x|))[\langle\langle x, y\rangle, 1^l, \omega\rangle \in A].$$

Let q be a polynomial such that $q(n) \geq s(p_2(n))$ for all n. Define

$$B = \{\langle x, 1^l, \omega\rangle \mid (\exists y : |y| \leq p_2(|x|))[\langle\langle x, y\rangle, 1^l, \omega\rangle \in A]\}$$

and note that $B \in \mathrm{NP}$. This proves Claim F.

G Applying Claim F twice provides a reduction from $L_{\Sigma_3^{\mathrm{p}}}$ to an NP language requiring two hard pairs.

Claim G: There exist a set $C \in \mathrm{NP}$ and polynomials q_1 and q_2 such that for all $x \in \Sigma^*$, if $\langle 1^{q_1(|x|)}, \omega_1\rangle$ and $\langle 1^{q_2(|x|)}, \omega_2\rangle$ are hard pairs then

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff \langle x, \omega_1, \omega_2\rangle \in C.$$

Let $x \in \Sigma^*$. We have

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff (\exists y : |y| \leq p_3(|x|))[\langle x, y\rangle \notin L_{\Sigma_2^{\mathrm{p}}}].$$

According to Claim F there exist a set $B \in \mathrm{NP}$ and a polynomial q such that if $\langle 1^l, \omega_1\rangle$, $l \geq q(s(p_3(|x|)))$, is a hard pair then

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff (\exists y : |y| \leq p_3(|x|))[\langle\langle x, y\rangle, 1^l, \omega_1\rangle \notin B].$$

Let q_1 be a polynomial such that $q_1(n) \geq q(s(p_3(n)))$ for all n. Define

$$D = \{\langle x, 1^l, \omega_1\rangle \mid (\exists y : |y| \leq p_3(|x|))[\langle\langle x, y\rangle, 1^l, \omega_1\rangle \notin B]\}.$$

Note that $D \in \Sigma_2^{\mathrm{p}}$ and let g be a many-one reduction from D to $L_{\Sigma_2^{\mathrm{p}}}$. Let $\widehat{q}$ be a polynomial such that for all $z \in \Sigma^*$, $|g(z)| \leq \widehat{q}(|z|)$. Hence we have, if $\langle 1^{q_1(|x|)}, \omega_1\rangle$ is a hard pair then

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff g(\langle x, 1^{q_1(|x|)}, \omega_1\rangle) \in L_{\Sigma_2^{\mathrm{p}}}.$$

Applying Claim F again we obtain that if $\langle 1^l, \omega_2\rangle$, $l \geq q(\widehat{q}(s(q_1(|x|))))$, is a hard pair and $|\omega_1| \leq q_1(|x|)$,

$$g(\langle x, 1^{q_1(|x|)}, \omega_1\rangle) \in L_{\Sigma_2^{\mathrm{p}}} \iff \langle g(\langle x, 1^{q_1(|x|)}, \omega_1\rangle), 1^l, \omega_2\rangle \in B.$$

Altogether, if $\langle 1^{q_1(|x|)}, \omega_1\rangle$, and $\langle 1^l, \omega_2\rangle$, $l \geq q(\widehat{q}(s(q_1(|x|))))$, are hard pairs then

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff \langle g(\langle x, 1^{q_1(|x|)}, \omega_1\rangle), 1^l, \omega_2\rangle \in B.$$

Let q_2 be a polynomial such that $q_2(n) \geq q(\widehat{q}(s(q_1(n))))$ for all n. Define $C = \{\langle x, \omega_1, \omega_2\rangle \,|\, \langle g(\langle x, 1^{q_1(|x|)}, \omega_1\rangle), 1^{q_2(|x|)}, \omega_2\rangle \in B\}$. This completes the proof of Claim G.

Observe that, in light of the algorithm given in **H** below, a reduction of $L_{\Sigma_3^{\mathrm{p}}}$ to the Σ_2^{p} language (as it was done in the proof of Theorem 3.2.7 (Wagner 1987)) would suffice. This is implicitly done by the set D in the above proof of Claim G. Since this would clearly require only one hard pair one could similarly to **H** derive even $L_{\Sigma_3^{\mathrm{p}}} \in \mathrm{P}^{\Sigma_2^{\mathrm{p}}[2]}$. However, the possibility of reducing $L_{\Sigma_3^{\mathrm{p}}}$ to an NP language first appeared in [Wag89] and was crucially used in [CK89, CK96] and [BCO91, BCO93].

H In contrast to **H** of the proof of Theorem 3.2.7 (Wagner 1987), in light of Claim G, only two values of f have to be computed.

Claim H: $L_{\Sigma_3^{\mathrm{p}}} \in \mathrm{P}^{\Sigma_2^{\mathrm{p}}[3]}$.

Let $C \in \mathrm{NP}$ and q_1 and q_2 be polynomials, all three as defined in **G**. We give a $\mathrm{P}^{\Sigma_2^{\mathrm{p}}[3]}$ algorithm for $L_{\Sigma_3^{\mathrm{p}}}$.

1. On input x compute $f(1^{q_1(|x|)})$ and $f(1^{q_2(|x|)})$. This amounts for two Σ_2^{p} queries.
2. Query $\langle x, f(1^{q_1(|x|)}), f(1^{q_2(|x|)})\rangle$. The Σ_2^{p} oracle queried in this part of the $\mathrm{P}^{\Sigma_2^{\mathrm{p}}[3]}$ algorithm implicitly does the following:
   (a) On input $\langle x, y_1, y_2\rangle$ compute $q_1(|x|)$ and $q_2(|x|)$.
   (b) If $y_1 = 0$ set $\omega_1 = \#$. If $y_1 = 1$ guess ω_1, $|\omega_1| \leq q_1(|x|)$, and verify that ω_1 is a hard string for length $q_1(|x|)$. Continue if this verification succeeds, otherwise reject.
   (c) If $y_2 = 0$ set $\omega_2 = \#$. If $y_2 = 1$ guess ω_2, $|\omega_2| \leq q_2(|x|)$, and verify that ω_2 is a hard string for length $q_2(|x|)$. Continue if this verification succeeds, otherwise reject.
   (d) Accept if and only if $\langle x, \omega_1, \omega_2\rangle \in C$.
3. Accept (the input x) if and only if the query from part 2 returns yes.

The correctness of this algorithm is obvious. In particular, recall from **H** of the proof of Theorem 3.2.7 (Wagner 1987) that the use of different Σ_2^{p} oracles does no harm to the algorithm.

I Since $L_{\Sigma_3^{\mathrm{p}}}$ is complete for Σ_3^{p} we have $\Sigma_3^{\mathrm{p}} = \mathrm{P}^{\Sigma_2^{\mathrm{p}}[3]}$ and hence $\mathrm{PH} = \mathrm{P}^{\Sigma_2^{\mathrm{p}}[3]}$.

∎

Chang and Kadin 1989 [CK89, CK96]

Theorem 3.2.9 If DP = coDP then PH = $\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})$.

Proof: A As in the proof of Theorem 3.2.6 (Kadin 1987).

B,C,D,E,F, and G As in the proof of Theorem 3.2.8 (Wagner 1989).

H In contrast to **G** one hard pair suffices to reduce a Σ_3^{p} complete language to a Σ_2^{p} language.

Claim H: There exist a set $D_1 \in \Sigma_2^{\mathrm{p}}$ and a polynomial $\widehat{p}_1$ such that for all $x \in \Sigma^*$ and all $l \geq \widehat{p}_1(|x|)$, if $\langle 1^l, \omega\rangle$ is a hard pair then

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff \langle x, 1^l, \omega\rangle \in D_1.$$

Claim H is the analogue of **G** in the proof of Theorem 3.2.7 (Wagner 1987) with the modifications induced by the different hard strings definition and is implicitly contained in **G** of the proof of Theorem 3.2.8 (Wagner 1989). In particular, setting $\widehat{p}_1 = q_1$ and $D_1 = D$, where q_1 and D are as defined in **G** of the proof of Theorem 3.2.8 (Wagner 1989) proves the claim.

I Define $S = \{1^l \mid f(1^l) = 1\}$. Though $f \in \mathrm{FP}^{\Sigma_2^{\mathrm{p}}[1]}$, testing whether $f(1^l) = 1$ can be done with a Σ_2^{p} algorithm, as it is just testing whether there exists a hard string for length l. So $S \in \Sigma_2^{\mathrm{p}}$.

Claim I: There exist a set $T \in \mathrm{NP}$ and a polynomial p_t such that for all $l \in \mathbb{N}$ and all $l' \geq p_t(l)$, if $\langle 1^{l'}, \omega\rangle$ is a hard pair then

$$1^l \in S \iff \langle 1^l, 1^{l'}, \omega\rangle \in T.$$

The claim follows immediately from **F**.

J The above Claim I turns into the key tool to reduce a Π_3^{p} complete language to a Σ_2^{p} language with the help of just one hard pair.

Claim J: There exist a set $D_2 \in \Sigma_2^{\mathrm{p}}$ and a polynomial $\widehat{p}_2$ such that for all $x \in \Sigma^*$ and all $l \geq \widehat{p}_2(|x|)$, if $\langle 1^l, \omega\rangle$ is a hard pair then

$$x \notin L_{\Sigma_3^{\mathrm{p}}} \iff \langle x, 1^l, \omega\rangle \in D_2.$$

It will soon be clear that Claim J is the key trick in the current proof. Let p_t, q_1, and q_2 be polynomials and $C \in \mathrm{NP}$, all four as defined in **G** and **I**. Let $\widehat{p}_2$ be a polynomial such that $\widehat{p}_2(n) \geq p_t(q_2(n))$ for all n. D_2 is defined by the following Σ_2^{p} algorithm.

1. On input $\langle x, 1^l, \omega\rangle$, compute $q_1(|x|)$ and $q_2(|x|)$.

2. Assuming that $\langle 1^l, \omega\rangle$ is a hard pair and $l \geq p_t(q_1(|x|))$ we determine $f(1^{q_1(|x|)})$ by applying Claim I. This is done as follows: Test using an NP oracle query whether $\langle 1^{q_1(|x|)}, 1^l, \omega\rangle \in T$. Set $j_1 = 1$ if this is the case, otherwise $j_1 = 0$.
3. Assuming that $\langle 1^l, \omega\rangle$ is a hard pair and $l \geq p_t(q_2(|x|))$ we determine $f(1^{q_2(|x|)})$ by applying Claim I. This is done similar to step 2: Test using an NP oracle query whether $\langle 1^{q_2(|x|)}, 1^l, \omega\rangle \in T$. Set $j_2 = 1$ if this is the case, otherwise $j_2 = 0$.
4. If $j_1 = 1$ guess a string ω_1, $|\omega_1| \leq q_1(|x|)$, verify that ω_1 is hard for length $q_1(|x|)$, continue if this is the case, and reject otherwise. If $j_1 = 0$ set $\omega_1 = \#$.
5. If $j_2 = 1$ guess a string ω_2, $|\omega_2| \leq q_2(|x|)$, verify that ω_2 is hard for length $q_2(|x|)$, continue if this is the case, and reject otherwise. If $j_2 = 0$ set $\omega_2 = \#$.
6. Assuming that $\langle 1^{q_1(|x|)}, \omega_1\rangle$ and $\langle 1^{q_2(|x|)}, \omega_2\rangle$ are hard pairs we determine whether $x \notin L_{\Sigma_3^p}$ using Claim G. In other words, accept if and only if $\langle x, \omega_1, \omega_2\rangle \notin C$.

Observe that if $\langle 1^l, \omega\rangle$ is a hard pair and $l \geq p_t(q_1(|x|))$, step 2 indeed yields $j_1 = f(1^{q_1(|x|)})$ according to **I**. Similarly, if $\langle 1^l, \omega\rangle$ is a hard pair and $l \geq p_t(q_2(|x|))$, step 3 yields $j_2 = f(1^{q_2(|x|)})$ according to **I**. Furthermore, if $\langle 1^{q_1(|x|)}, \omega_1\rangle$ and $\langle 1^{q_2(|x|)}, \omega_2\rangle$ are hard pairs then we accept in step 6 if and only if $x \notin L_{\Sigma_3^p}$. But, if steps 2 and 3 yield $j_1 = f(1^{q_1(|x|)})$ and $j_2 = f(1^{q_2(|x|)})$, respectively, the algorithm indeed determines hard pairs in steps 4 and 5 and hence the algorithm correctly accepts in step 6.

Overall, the correctness of the above algorithm stands and falls with the correctness of steps 2 and 3. Hence setting $\widehat{p}_2(n) \geq p_t(q_2(n))$ for all n proves the claim (note that, in light of our convention about polynomials, $q_2(n) > q_1(n)$ for all n).

K Combining the results of Claims H and J while exploiting the difference structure of $\mathrm{DIFF}_2(\Sigma_2^p)$ yields

Claim K: $L_{\Sigma_3^p} \in \mathrm{DIFF}_2(\Sigma_2^p)$.

Let the sets $D_1, D_2 \in \Sigma_2^p$ and the polynomials $\widehat{p}_1$ and $\widehat{p}_2$ be as defined in **H** and **J**. Let p be a polynomial such that $p(n) \geq \max\{\widehat{p}_1(n), \widehat{p}_2(n)\}$ for all n. Define

$$E_1 = \{x \mid \langle x, 1^{p(|x|)}, \#\rangle \in D_1\},$$
$$E_2 = \{x \mid (\exists \omega \in \Sigma^*)[\omega \text{ is a hard string for length } p(|x|) \text{ and } \langle x, 1^{p(|x|)}, \omega\rangle \in D_1\},$$
$$E_3 = \{x \mid (\exists \omega \in \Sigma^*)[\omega \text{ is a hard string for length } p(|x|) \text{ and } \langle x, 1^{p(|x|)}, \omega\rangle \in D_2\}.$$

Clearly, $E_1, E_2, E_3 \in \Sigma_2^p$. Since Σ_2^p is closed under union we also have $E_1 \cup E_2 \in \Sigma_2^p$. Hence $(E_1 \cup E_2) - E_3 \in \mathrm{DIFF}_2(\Sigma_2^p)$. We show $L_{\Sigma_3^p} = (E_1 \cup E_2) - E_3$. To see this consider the following case distinction. Let $x \in \Sigma^*$.

Case 1 $f(1^{p(|x|)}) = 0$.

Hence $x \notin E_2$ and $x \notin E_3$. Furthermore, $\langle 1^{p(|x|)}, \#\rangle$ is a hard pair and hence according to Claim H,

$$\begin{aligned} x \in L_{\Sigma_3^{\mathrm{p}}} &\iff \langle x, 1^{p(|x|)}, \# \rangle \in D_1 \\ &\iff x \in E_1 \\ &\iff x \in (E_1 \cup E_2) - E_3. \end{aligned}$$

Case 2 $f(1^{p(|x|)}) = 1$.
Hence there exist hard strings for length $p(|x|)$. If $x \in L_{\Sigma_3^{\mathrm{p}}}$ then we clearly have both, $\langle x, 1^{p(|x|)}, \omega \rangle \in D_1$ and $\langle x, 1^{p(|x|)}, \omega \rangle \notin D_2$ for all hard strings ω for length $p(|x|)$, according to Claims H and J. Hence $x \in (E_1 \cup E_2) - E_3$. If $x \notin L_{\Sigma_3^{\mathrm{p}}}$ then $\langle x, 1^{p(|x|)}, \omega \rangle \notin D_1$ and $\langle x, 1^{p(|x|)}, \omega \rangle \in D_2$ for all hard strings ω for length $p(|x|)$, according to Claims H and J. Independent of whether $x \in E_1$ or $x \notin E_1$ we have $x \notin (E_1 \cup E_2) - E_3$.

L We have shown $L_{\Sigma_3^{\mathrm{p}}} \in \mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})$ and thus $\Sigma_3^{\mathrm{p}} = \mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})$ which immediately implies $\mathrm{PH} = \mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})$.

∎

Beigel, Chang, and Ogihara 1991 [BCO91, BCO93]

Theorem 3.2.10 If DP = coDP then $\mathrm{PH} = \left(\mathrm{P}^{\mathrm{NP}}_{1\text{-tt}}\right)^{\mathrm{NP}}$.

Proof: Recall that $\left(\mathrm{P}^{\mathrm{NP}}_{1\text{-tt}}\right)^{\mathrm{NP}}$ is the class of languages accepted by some DPTM making at most one query to an $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^{\mathrm{p}}$ oracle and polynomially many queries to an oracle from NP.

A As in the proof of Theorem 3.2.6 (Kadin 1987).

B,C,D,E,F, and G As in the proof of Theorem 3.2.8 (Wagner 1989).

H Quite similar to **H** of the proof of Theorem 3.2.9 (Chang and Kadin 1989) one hard pair suffices to reduce a $\mathrm{P}^{\Sigma_2^{\mathrm{p}}}$ language to a P^{NP} language.

Claim H: Let $L \in \mathrm{P}^{\Sigma_2^{\mathrm{p}}}$. There exist a set $D \in \mathrm{P}^{\mathrm{NP}}$ and a polynomial $\widehat{p}$ such that for all $x \in \Sigma^*$, if $\langle 1^{\widehat{p}(|x|)}, \omega \rangle$ is a hard pair then

$$x \in L \iff \langle x, \omega \rangle \in D.$$

The proof is a straightforward application of **F**. Let $L \in \mathrm{P}^{\Sigma_2^{\mathrm{p}}}$, hence $L = L(N_1^{L_{\Sigma_2^{\mathrm{p}}}})$ for some DPTM N_1 running in time p for some polynomial p. According to **F** there exist a language $B \in \mathrm{NP}$ and a polynomial q such that for all $x \in \Sigma^*$ and all $l \geq q(|x|)$, if $\langle 1^l, \omega \rangle$ is a hard pair then

$$x \in L_{\Sigma_2^{\mathrm{p}}} \iff \langle x, 1^l, \omega \rangle \in B.$$

We use this to reduce L to a P^{NP} language with the help of one hard pair. Let $\widehat{p}$ be a polynomial such that $\widehat{p}(n) \geq q(p(n))$ for all n. Define the DPTM N_2^B as follows: $N_2^B(\langle x, \omega\rangle)$ simulates the work of $N_1^{L_{\Sigma_2^{\mathrm{p}}}}(x)$ but replaces every query v to $L_{\Sigma_2^{\mathrm{p}}}$ by a query $\langle v, 1^{\widehat{p}(|x|)}, \omega\rangle$ to B. Let $D = L(N_2^B)$. Clearly, $D \in \mathrm{P}^{\mathrm{NP}}$.

I *Claim I:* $L_{\Sigma_3^{\mathrm{p}}} \in \left(\mathrm{P}^{\mathrm{NP}}_{2\text{-tt}}\right)^{\mathrm{NP}}$.

According to Claim G there exist a language $C \in \mathrm{NP}$ and polynomials q_1 and q_2 such that for all $x \in \Sigma^*$, if $\langle 1^{q_1(|x|)}, \omega_1\rangle$ and $\langle 1^{q_2(|x|)}, \omega_2\rangle$ are hard pairs then

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff \langle x, \omega_1, \omega_2\rangle \in C.$$

If no hard strings for length $q_1(|x|)$ and $q_2(|x|)$ exist then $x \in L_{\Sigma_3^{\mathrm{p}}} \iff \langle x, \#, \#\rangle \in C$. But note that $\chi_C(\langle x, \#, \#\rangle)$ can also be used in general for determining $\chi_{L_{\Sigma_3^{\mathrm{p}}}}(x)$ if we know whether existing hard strings ω_1 and ω_2 for length $q_1(|x|)$ and $q_2(|x|)$, respectively, provide $\chi_C(\langle x, \#, \#\rangle) \neq \chi_C(\langle x, \omega_1, \omega_2\rangle)$ or $\chi_C(\langle x, \#, \#\rangle) = \chi_C(\langle x, \omega_1, \omega_2\rangle)$. This approach is also known as the mind change technique. It enables us to give a $\left(\mathrm{P}^{\mathrm{NP}}_{2\text{-tt}}\right)^{\mathrm{NP}}$ algorithm for $L_{\Sigma_3^{\mathrm{p}}}$.

1. On input x, query $\langle x, \#, \#\rangle$ to C.
2. Query x in parallel to the two under (a) and (b) described Σ_2^{p} oracles.
   (a) Accept the query x if and only if
      i. there exists a hard string ω_1 for length $q_1(|x|)$ such that $\chi_C(\langle x, \omega_1, \#\rangle) \neq \chi_C(\langle x, \#, \#\rangle)$, or
      ii. there exists a hard string ω_2 for length $q_2(|x|)$ such that $\chi_C(\langle x, \#, \omega_2\rangle) \neq \chi_C(\langle x, \#, \#\rangle)$, or
      iii. there exist two hard strings ω_1 and ω_2 for length $q_1(|x|)$ and $q_2(|x|)$, respectively, such that $\chi_C(\langle x, \omega_1, \omega_2\rangle) \neq \chi_C(\langle x, \#, \#\rangle)$.
   (b) Accept the query x if and only if there exist two hard strings ω_1 and ω_2 for length $q_1(|x|)$ and $q_2(|x|)$, respectively, such that either
      i. $\chi_C(\langle x, \omega_1, \omega_2\rangle) \neq \chi_C(\langle x, \omega_1, \#\rangle)$ and $\chi_C(\langle x, \omega_1, \#\rangle) \neq \chi_C(\langle x, \#, \#\rangle)$ or
      ii. $\chi_C(\langle x, \omega_1, \omega_2\rangle) \neq \chi_C(\langle x, \#, \omega_2\rangle)$ and $\chi_C(\langle x, \#, \omega_2\rangle) \neq \chi_C(\langle x, \#, \#\rangle)$.
3. Accept if and only if the three queries from parts 1, 2a, and 2b return in this order either the answers "yes, no, no," or "no, yes, no," or "yes, yes, yes."

The correctness of this algorithm follows immediately from the construction. Note that the use of different Σ_2^{p} oracles in part 2 does not affect the correctness of our algorithm as already pointed out in previous proofs. Note that part 2a corresponds to checking whether the existence of hard strings causes at least one mind change, whereas part 2b corresponds to determining whether the existence of hard strings causes two mind changes.

J The $\left(\mathrm{P}^{\mathrm{NP}}_{2\text{-tt}}\right)^{\mathrm{NP}}$ algorithm for $L_{\Sigma^{\mathrm{p}}_3}$ as given in **I** can be improved to a $\left(\mathrm{P}^{\mathrm{NP}}_{1\text{-tt}}\right)^{\mathrm{NP}}$ algorithm by exploiting Claim H.

Claim J: $\mathrm{P}^{\Sigma^{\mathrm{p}}_2} \subseteq \left(\mathrm{P}^{\mathrm{NP}}_{1\text{-tt}}\right)^{\mathrm{NP}}$.

Let $L \in \mathrm{P}^{\Sigma^{\mathrm{p}}_2}$. Let $D \in \mathrm{P}^{\mathrm{NP}}$ and $\widehat{p}$ be a polynomial, both as defined in **H**. We describe a $\left(\mathrm{P}^{\mathrm{NP}}_{1\text{-tt}}\right)^{\mathrm{NP}}$ algorithm for L using the same trick as in **I**.

1. On input x determine whether $\langle x, \# \rangle \in D$. This can be done with the help of queries to an NP oracle, since $D \in \mathrm{P}^{\mathrm{NP}}$.
2. Determine whether there exists a hard string ω for length $\widehat{p}(|x|)$ such that $\chi_D(\langle x, \omega \rangle) \neq \chi_D(\langle x, \# \rangle)$. This can be done with one query to a Σ^{p}_2 oracle.
3. Accept if and only if the (implicit) query "$\langle x, \# \rangle \in D$?" of part 1 and the query of part 2 return different answers.

Note that step 2 corresponds to determining whether the existence of a hard string causes a mind change.

K Since $\Sigma^{\mathrm{p}}_3 \subseteq \left(\mathrm{P}^{\mathrm{NP}}_{2\text{-tt}}\right)^{\mathrm{NP}}$ (Claim I), $\left(\mathrm{P}^{\mathrm{NP}}_{2\text{-tt}}\right)^{\mathrm{NP}} \subseteq \mathrm{P}^{\Sigma^{\mathrm{p}}_2}$, and $\mathrm{P}^{\Sigma^{\mathrm{p}}_2} \subseteq \left(\mathrm{P}^{\mathrm{NP}}_{1\text{-tt}}\right)^{\mathrm{NP}}$ (Claim J) we have proven $\Sigma^{\mathrm{p}}_3 \subseteq \left(\mathrm{P}^{\mathrm{NP}}_{1\text{-tt}}\right)^{\mathrm{NP}}$ and thus $\mathrm{PH} = \left(\mathrm{P}^{\mathrm{NP}}_{1\text{-tt}}\right)^{\mathrm{NP}}$.

∎

3.3 A New Result

3.3.1 A Deeper Collapse of the Polynomial Hierarchy if the Boolean Hierarchy Collapses

The five papers studied in the previous section obtained deeper and deeper collapses of the polynomial hierarchy if the boolean hierarchy over NP collapses. The strongest result previously known is due to Beigel, Chang, and Ogihara [BCO93], see Theorem 3.2.5. Theorem 3.2.5 says that, given a collapse of the boolean hierarchy at level m, $m \geq 2$, the polynomial hierarchy collapses to $\left(\mathrm{P}^{\mathrm{NP}}_{m-1\text{-tt}}\right)^{\mathrm{NP}}$, a class contained in $\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_2)$.

A careful analysis of the proof of Theorem 3.2.5 as given in [BCO93] in combination with a new trick, namely, applying an idea developed in [BCO91, BCO93] twice, yields the following theorem.

Theorem 3.3.1 For all $m \geq 2$,

$$\mathrm{BH}_m = \mathrm{coBH}_m \implies \mathrm{PH} = \mathrm{BH}_m \boldsymbol{\Delta} \mathrm{DIFF}_{m-1}(\Sigma^{\mathrm{p}}_2).$$

Note that for $m = 1$ the hypothesis of the above Theorem 3.3.1 implies a trivial collapse of the polynomial hierarchy, namely, to NP itself. Theorem 3.3.1 has been independently obtained by Reith and Wagner [RW98].

Let us compare the results of Theorem 3.2.5 and Theorem 3.3.1. Though both theorems collapse the polynomial hierarchy to a class containing $\mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$ and being contained in $\mathrm{DIFF}_m(\Sigma_2^{\mathrm{p}})$, their results differ substantially. It is immediate from a recent paper of Wagner [Wag98] that $\left(\mathrm{P}^{\mathrm{NP}}_{m-1\text{-tt}}\right)^{\mathrm{NP}}$ is a strict superset of $\mathrm{BH}_m\boldsymbol{\Delta}\mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$ unless the polynomial hierarchy collapses. Furthermore, observe that $\left(\mathrm{P}^{\mathrm{NP}}_{m-1\text{-tt}}\right)^{\mathrm{NP}}$ involves $m-1$ parallel queries to a Σ_2^{p} oracle and an unlimited number of queries to an NP oracle. So the P base machine of $\left(\mathrm{P}^{\mathrm{NP}}_{m-1\text{-tt}}\right)^{\mathrm{NP}}$ evaluates $m-1$ bits of information originating from the parallel Σ_2^{p} queries and polynomially many bits of information from the NP queries. In contrast, $\mathrm{BH}_m\boldsymbol{\Delta}\mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$ involves just two bits of information, which are evaluated via a fixed truth-table, namely, the XOR-truth-table. One bit of information comes from the $\mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$ part consisting of $m-1$ underlying parallel queries to a Σ_2^{p} oracle evaluated with one fixed truth-table. The second bit of information, the one from the BH_m part, implicitly contains m parallel queries to an NP oracle, their answers again being evaluated via a fixed truth-table. In a nutshell, we have improved from unlimited many queries to NP and $m-1$-truth-table queries to Σ_2^{p}, to m-fixed-truth-table queries to NP and $m-1$-fixed-truth-table queries to Σ_2^{p}.

Proof of Theorem 3.3.1: The proof is structured in such a way that the analogies to the proofs of the special cases in Section 3.2 can be easily observed. Recall our convention about polynomials from the beginning of Section 3.2.3. Recall that $\Sigma = \{0,1\}$ and $\# \notin \Sigma$. Let s be a polynomial such that for all $1 \le j \le m+1$ and all $x_1, x_2, \ldots, x_j \in \Sigma^* \cup \{\#\}$, $|\langle x_1, x_2, \ldots, x_j\rangle| \le s(\max\{|x_1|, |x_2|, \ldots, |x_j|\})$. Define $s^{(0)}(n) = n$ and $s^{(i)}(n) = \underbrace{s(s(\cdots s}_{i\ times}(n)\cdots))$ for all n and all $i \ge 1$.

Suppose $m \ge 2$.

A Let L_{NP}, $L_{\Sigma_2^{\mathrm{p}}}$, and $L_{\Sigma_3^{\mathrm{p}}}$ be many-one complete languages for NP, Σ_2^{p}, and Σ_3^{p}, respectively, and p_2, p_3 be polynomials such that

$$L_{\Sigma_2^{\mathrm{p}}} = \{x \mid (\exists y : |y| \le p_2(|x|))[\langle x, y\rangle \notin L_{\mathrm{NP}}]\}$$

and

$$L_{\Sigma_3^{\mathrm{p}}} = \{x \mid (\exists y : |y| \le p_3(|x|))[\langle x, y\rangle \notin L_{\Sigma_2^{\mathrm{p}}}]\}.$$

Hence, $L_{\mathrm{coNP}} = \overline{L_{\mathrm{NP}}}$ is a complete language for coNP. Define $L_{\mathrm{DIFF}_1(\mathrm{coNP})} = L_{\mathrm{coNP}}$, and for every $i \ge 2$, $L_{\mathrm{DIFF}_i(\mathrm{coNP})} = \{\langle x, y\rangle \mid x \in L_{\mathrm{coNP}} \wedge y \notin L_{\mathrm{DIFF}_{i-1}(\mathrm{coNP})}\}$. It is not hard to verify that for all $i \ge 1$, $L_{\mathrm{DIFF}_i(\mathrm{coNP})}$ is many-one complete for $\mathrm{DIFF}_i(\mathrm{coNP})$.

Note that $\mathrm{DIFF}_m(\mathrm{coNP}) = \mathrm{BH}_m$ if m is even and $\mathrm{DIFF}_m(\mathrm{coNP}) = \mathrm{coBH}_m$ if m is odd. So in general,

$$\mathrm{BH}_m = \mathrm{coBH}_m \iff \mathrm{DIFF}_m(\mathrm{coNP}) = \mathrm{coDIFF}_m(\mathrm{coNP}).$$

B Suppose $\mathrm{BH}_m = \mathrm{coBH}_m$. Hence $\mathrm{DIFF}_m(\mathrm{coNP}) = \mathrm{coDIFF}_m(\mathrm{coNP})$. Thus there exists a many-one reduction h from $L_{\mathrm{DIFF}_m(\mathrm{coNP})}$ to $\overline{L_{\mathrm{DIFF}_m(\mathrm{coNP})}}$. In other words, there exists a polynomial-time computable function h such that for all $x_1, x_2 \in \Sigma^*$,

$$\langle x_1, x_2\rangle \in L_{\mathrm{DIFF}_m(\mathrm{coNP})} \iff h(\langle x_1, x_2\rangle) \notin L_{\mathrm{DIFF}_m(\mathrm{coNP})}.$$

Let h' and h'' be the polynomial-time computable functions such that for all $x_1, x_2 \in \Sigma^*$, $h(\langle x_1, x_2\rangle) = \langle h'(\langle x_1, x_2\rangle), h''(\langle x_1, x_2\rangle)\rangle$. Hence, we have for all $x_1, x_2 \in \Sigma^*$,

$$(*) \qquad x_1 \in L_{\mathrm{coNP}} \wedge x_2 \notin L_{\mathrm{DIFF}_{m-1}(\mathrm{coNP})} \iff h'(\langle x_1, x_2\rangle) \notin L_{\mathrm{coNP}} \vee h''(\langle x_1, x_2\rangle) \in L_{\mathrm{DIFF}_{m-1}(\mathrm{coNP})}.$$

C Recall that we want to show a collapse of the polynomial hierarchy. Though we do not claim that we can prove NP = coNP we will nevertheless show that an NP algorithm for L_{coNP} exists which requires certain additional input. We will extend this to also give NP algorithms for $L_{\Sigma_2^p}$ and $L_{\Sigma_3^p}$, both algorithms requiring additional input.

Let n be an integer. In light of the equivalence (*), we call the string x_1 m-easy for length n if and only if $|x_1| \leq n$ and $(\exists x_2 : |x_2| \leq s^{(m-2)}(n))[h'(\langle x_1, x_2\rangle) \notin L_{\mathrm{coNP}}]$. Clearly, if x_1 is m-easy for length n then $x_1 \in L_{\mathrm{coNP}}$.

A string x_1 is said to be m-hard for length n if and only if $|x_1| \leq n$, $x_1 \in L_{\mathrm{coNP}}$, and $(\forall x_2 : |x_2| \leq s^{(m-2)}(n))[h'(\langle x_1, x_2\rangle) \in L_{\mathrm{coNP}}]$. It is not hard to verify that the strings in $(L_{\mathrm{coNP}})^{\leq n}$ divide into m-easy and m-hard strings for length n.

Case 1 There are no m-hard strings for length n.
Hence all strings in $(L_{\mathrm{coNP}})^{\leq n}$ are m-easy for length n. Thus deciding whether x, $|x| = n$, is in L_{coNP} is equivalent to deciding whether x is m-easy for length n. Note that the latter can be done by the following NP algorithm:

1. Guess y, $|y| \leq s^{(m-2)}(n)$.
2. Compute $h(\langle x, y\rangle)$.
3. Accept if and only if $h'(\langle x, y\rangle) \notin L_{\mathrm{coNP}}$.

Case 2 There exist m-hard strings for length n.
Let ω_m be an m-hard string for length n, hence $|\omega_m| \leq n$. For every $a \in \Sigma^*$, let $h_{(a)}$ be the function such that for every $u \in \Sigma^*$, $h_{(a)}(u) = h''(\langle a, u\rangle)$. Note that $h_{(\omega_m)}(u)$ is computable in time polynomial in $\max\{n, |u|\}$. According to the definition of m-hard strings and equivalence (*) we have for all u, $|u| \leq s^{(m-2)}(n)$,

$$u \in L_{\mathrm{DIFF}_{m-1}(\mathrm{coNP})} \iff h_{(\omega_m)}(u) \notin L_{\mathrm{DIFF}_{m-1}(\mathrm{coNP})}.$$

Thus we have a situation similar to the one in **B** but m replaced by $m-1$ and also the equivalence holds only for an initial segment. In analogy to the definition of h' and h'' let for every $a \in \Sigma^*$, $h'_{(a)}$ and $h''_{(a)}$ be the functions such that for all $x_1, x_2 \in \Sigma^*$, $h_{(a)}(\langle x_1, x_2\rangle) = \langle h'_{(a)}(\langle x_1, x_2\rangle), h''_{(a)}(\langle x_1, x_2\rangle)\rangle$.

Let $u = \langle u_1, u_2\rangle$. Hence for all u_1, $|u_1| \leq n$, and all u_2, $|u_2| \leq s^{(m-3)}(n)$,

$$u_1 \in L_{\mathrm{coNP}} \wedge u_2 \notin L_{\mathrm{DIFF}_{m-2}(\mathrm{coNP})} \iff \\ h'_{(\omega_m)}(\langle u_1, u_2\rangle) \notin L_{\mathrm{coNP}} \vee h''_{(\omega_m)}(\langle u_1, u_2\rangle) \in L_{\mathrm{DIFF}_{m-2}(\mathrm{coNP})}.$$

We call the string u_1 $m-1$-easy for length n if and only if $|u_1| \leq n$ and $(\exists u_2 : |u_2| \leq s^{(m-3)}(n))[h'_{(\omega_m)}(\langle u_1, u_2\rangle) \notin L_{\mathrm{coNP}}]$. If u_1 is $m-1$-easy for length n then $u_1 \in L_{\mathrm{coNP}}$.
A string u_1 is said to be $m-1$-hard for length n if and only if $|u_1| \leq n$, $u_1 \in L_{\mathrm{coNP}}$, and $(\forall u_2 : |u_2| \leq s^{(m-3)}(n))[h'_{(\omega_m)}(\langle u_1, u_2\rangle) \in L_{\mathrm{coNP}}]$.
It is not hard to verify that, given an m-hard string ω_m for length n, the strings in $(L_{\mathrm{coNP}})^{\leq n}$ divide into $m-1$-easy and $m-1$-hard strings for length n. Note that $m-1$-hardness is only defined with respect to some particular m-hard string ω_m.

Case 2.1 There exist no $m-1$-hard strings for length n.
Hence similar to Case 1, all strings in $(L_{\mathrm{coNP}})^{\leq n}$ are $m-1$-easy for length n, deciding whether x, $|x| = n$, is in L_{coNP} is equivalent to deciding whether x is $m-1$-easy for length n which, with the help of ω_m, can be done with an NP algorithm.

Case 2.2 There exist $m-1$-hard strings for length n.
Let ω_{m-1} be an $m-1$-hard string for length n, $|\omega_{m-1}| \leq n$. For all $a, b \in \Sigma^*$, let $h_{(a,b)}$ be the function such that for all $v \in \Sigma^*$, $h_{(a,b)}(v) = h''_{(a)}(\langle b, v\rangle)$. Note that $h_{(\omega_m,\omega_{m-1})}(v)$ is computable in time polynomial in $\max\{n, |v|\}$. Hence, for all v, $|v| \leq s^{(m-3)}(n)$,

$$v \in L_{\mathrm{DIFF}_{m-2}(\mathrm{coNP})} \iff h_{(\omega_m,\omega_{m-1})}(v) \notin L_{\mathrm{DIFF}_{m-2}(\mathrm{coNP})}.$$

Continuing in that manner we define for $\ell \geq 2$, ℓ-hard and ℓ-easy strings for length n. Note that these terms are defined with respect to some fixed m-hard, $m-1$-hard, ... , $\ell+1$-hard strings. In other words, a string is only ℓ-hard or ℓ-easy with respect to a particular sequence of hard strings $\omega_m, \omega_{m-1}, \ldots, \omega_{\ell+1}$. We define that there are no 1-hard strings for length n, and a string z is called 1-easy for length n if and only if $|z| \leq n$ and $h_{(\omega_m,\omega_{m-1},\ldots,\omega_2)}(z) \notin L_{\mathrm{coNP}}$.

A sequence $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$, $\ell \geq 2$, is called a hard sequence for length n if and only if for all j, $\ell \leq j \leq m$, ω_j is j-hard for length n with respect to ω_m, ω_{m-1}, ... , ω_{j+1}.

We call $m - \ell + 1$ the order of the hard sequence $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$.

A sequence $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$ is called a maximal hard sequence for length n if and only if $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$ is a hard sequence for length n and there are no $(\ell-1)$-hard strings (with respect to ω_m, ω_{m-1}, ... , ω_ℓ) for length n. As a special case, # is called a maximal hard sequence for length n if and only if there exists no m-hard string for length n, # is said to have order zero. Note that deciding whether, given a sequence of strings s and an integer n, s is a hard sequence for length n can be done with a coNP algorithm.

It is clear that for every n, a maximal hard sequence for length n always exists and has order at most $m-1$ since there are no 1-hard strings for length n.

D One maximal hard sequence is needed to reduce L_{coNP} to an NP language.

Claim D: There exists a set $A \in \mathrm{NP}$ such that for all $x \in \Sigma^*$ and all $l \geq |x|$, if $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$ is a maximal hard sequence for length l then

$$x \in L_{\mathrm{coNP}} \iff \langle x, 1^l, \omega_m, \omega_{m-1}, \ldots, \omega_\ell \rangle \in A.$$

Let $x \in \Sigma^*$ and let $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$ be a maximal hard sequence for length l, $l \geq |x|$. Note that $\ell \geq 2$. Since $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$ is maximal hard, no string of length less or equal to l is $(\ell-1)$-hard with respect to $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$. Hence, for every string y, $|y| \leq l$, $y \in (L_{\mathrm{coNP}})^{\leq l}$ if and only if y is $(\ell-1)$-easy for length l. This holds especially for x itself (recall $|x| \leq l$). But testing whether x is $(\ell-1)$-easy for length l can clearly be done by an NP algorithm when receiving x, 1^l, and $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$ as inputs. In particular, define $A = \{\langle x, 1^l, \omega_m, \omega_{m-1}, \ldots, \omega_\ell \rangle \mid (\ell = 2 \wedge h_{(\omega_m, \omega_{m-1}, \ldots, \omega_\ell)}(x) \notin L_{\mathrm{coNP}}) \vee (\ell > 2 \wedge (\exists y : |y| \leq s^{\ell-3}(l))[h'_{(\omega_m, \omega_{m-1}, \ldots, \omega_\ell)}(\langle x, y \rangle) \notin L_{\mathrm{coNP}}]\}$.

E One maximal hard sequence for sufficiently large length also suffices to give a reduction from $L_{\Sigma_2^p}$ to an NP language.

Claim E: There exist a set $B \in \mathrm{NP}$ and a polynomial q such that for all $x \in \Sigma^*$ and all $l \geq q(|x|)$, if $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$ is a maximal hard sequence for length l then

$$x \in L_{\Sigma_2^p} \iff \langle x, 1^l, \omega_m, \omega_{m-1}, \ldots, \omega_\ell \rangle \in B.$$

Recall from **A** that for all $x \in \Sigma^*$,

$$x \in L_{\Sigma_2^p} \iff (\exists y : |y| \leq p_2(|x|))[\langle x, y \rangle \in L_{\mathrm{coNP}}].$$

Applying Claim D we obtain that there is a set $A \in \mathrm{NP}$ such that for all x and all $l \geq s(p_2(|x|))$, if $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$ is a maximal hard sequence for length l then

$$x \in L_{\Sigma_2^p} \iff (\exists y : |y| \leq p_2(|x|))[\langle \langle x, y \rangle, 1^l, \omega_m, \omega_{m-1}, \ldots, \omega_\ell \rangle \in A].$$

Note that the right-hand-side of the above equivalence clearly defines a NP language B. Define q to be a polynomial such that $q(n) \geq s(p_2(n))$ for all n. This proves the claim.

F In contrast to **D** and **E**, two maximal hard sequences for different length are required when reducing $L_{\Sigma_3^p}$ to an NP language.

Claim F: There exist a set $C \in \mathrm{NP}$ and polynomials q_1, q_2 such that for all $x \in \Sigma^*$, if $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$ and $\omega'_m, \omega'_{m-1}, \ldots, \omega'_{\ell'}$ are maximal hard sequences for length $q_1(|x|)$ and $q_2(|x|)$, respectively, then

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff \langle x, \langle \omega_m, \omega_{m-1}, \ldots, \omega_\ell\rangle, \langle \omega'_m, \omega'_{m-1}, \ldots, \omega'_{\ell'}\rangle\rangle \in C.$$

Recall from **A** that for all $x \in \Sigma^*$,

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff (\exists y : |y| \leq p_3(|x|))[\langle x, y\rangle \notin L_{\Sigma_2^{\mathrm{p}}}].$$

Applying Claim E we obtain that there is a set $B \in \mathrm{NP}$ and a polynomial q such that for all $x \in \Sigma^*$ and all $l \geq q(s(p_3(|x|)))$, if $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$ is a maximal hard sequence for length l then

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff (\exists y : |y| \leq p_3(|x|))[\langle\langle x, y\rangle, 1^l, \omega_m, \omega_{m-1}, \ldots, \omega_\ell\rangle \notin B].$$

Define q_1 to be a polynomial such that $q_1(n) \geq q(s(p_3(n)))$ for all n. Define $L' = \{\langle x, 1^l, \omega_m, \omega_{m-1}, \ldots, \omega_\ell\rangle \mid (\exists y : |y| \leq p_3(|x|))[\langle\langle x, y\rangle, 1^l, \omega_m, \omega_{m-1}, \ldots, \omega_\ell\rangle \notin B]\}$. Note that $L' \in \Sigma_2^{\mathrm{p}}$ and let g be a many-one reduction from L' to $L_{\Sigma_2^{\mathrm{p}}}$. Hence we have for all $x \in \Sigma^*$, if $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$ is a maximal hard sequence for length $q_1(|x|)$ then

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff g(\langle x, 1^{q_1(|x|)}, \omega_m, \omega_{m-1}, \ldots, \omega_\ell\rangle) \in L_{\Sigma_2^{\mathrm{p}}}.$$

Applying Claim E for the second time we obtain that for all $x \in \Sigma^*$, if $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$ is a maximal hard sequence for length $q_1(|x|)$ and $\omega'_m, \omega'_{m-1}, \ldots, \omega'_{\ell'}$ is a maximal hard sequence for length l, $l \geq q(|g(\langle x, 1^{q_1(|x|)}, \omega_m, \omega_{m-1}, \ldots, \omega_\ell\rangle)|)$, then

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff \langle g(\langle x, 1^{q_1(|x|)}, \omega_m, \omega_{m-1}, \ldots, \omega_\ell\rangle), 1^l, \omega'_m, \omega'_{m-1}, \ldots, \omega'_{\ell'}\rangle \in B.$$

Let $\widehat{q}$ be a polynomial such that $|g(z)|$ is bounded by $\widehat{q}(|z|)$ for all z. Define q_2 to be a polynomial such that $q_2(n) \geq q(\widehat{q}(s(q_1(n))))$ for all n. Set

$$\begin{aligned} C = \{\langle x, \langle \omega_m, \omega_{m-1}, \ldots, \omega_\ell\rangle, \langle \omega'_m, \omega'_{m-1}, \ldots, \omega'_{\ell'}\rangle\rangle \mid \\ \langle g(\langle x, 1^{q_1(|x|)}, \omega_m, \omega_{m-1}, \ldots, \omega_\ell\rangle), 1^{q_2(|x|)}, \omega'_m, \omega'_{m-1}, \ldots, \omega'_{\ell'}\rangle \in B\} \end{aligned}$$

and note that clearly $C \in \mathrm{NP}$. Furthermore, it is not hard to verify that C indeed satisfies Claim F.

G Applying the mind change technique in light of Claim C, we can conclude that $L_{\Sigma_3^{\mathrm{p}}} \in \mathrm{NP}\boldsymbol{\Delta}\mathrm{DIFF}_{2m-2}(\Sigma_2^{\mathrm{p}})$.

Claim G: $\mathrm{PH} \subseteq \mathrm{NP}\boldsymbol{\Delta}\mathrm{DIFF}_{2m-2}(\Sigma_2^{\mathrm{p}})$.

To prove the claim it suffices to show that $L_{\Sigma_3^{\mathrm{p}}} \in \mathrm{NP}\boldsymbol{\Delta}\mathrm{DIFF}_{2m-2}(\Sigma_2^{\mathrm{p}})$.

A few definitions will be helpful. For sequences of strings $u = (u_1, u_2, \ldots, u_j)$ and $v = (v_1, v_2, \ldots, v_{j'})$, v is called an extension of u if and only if $j \leq j'$ and for all

$1 \leq i \leq j$, $u_i = v_i$. v is called a proper extension of u if and only if v is an extension of u and $j < j'$. A similar definition is made for pairs of sequences of strings. For (u, v) and (u', v'), where u, u', v, v' are sequences of strings we call (u', v') an extension of (u, v) if and only if u' is an extension of u and v' is an extension of v. (u', v') is called a proper extension of (u, v) if and only if (u', v') is an extension of (u, v), and u' or v' is a proper extension of u or v, respectively.

Let $\ell_1(n), \ell_2(n)$ to be the orders of the longest maximal hard sequences for lengths $q_1(n)$ and $q_2(n)$, respectively, where q_1 and q_2 are the polynomials spoken of in Claim F. According to Claim F, for all x, $x \in L_{\Sigma_3^p}$ if and only if there exist two hard sequences s_1 and s_2 for length $q_1(|x|)$ and $q_2(|x|)$ of order $\ell_1(|x|)$ and $\ell_2(|x|)$, respectively, such that $\langle x, s_1, s_2 \rangle \in C$. However, in order to decide whether $x \in L_{\Sigma_3^p}$ or not, first computing s_1 and s_2 and then checking whether $\langle x, s_1, s_2 \rangle \in C$ might exceed the computational power of NP$\boldsymbol{\Delta}$DIFF$_{2m-2}(\Sigma_2^p)$. So we use a different strategy.

On one hand, if no m-hard strings for length $q_1(|x|)$ and $q_2(|x|)$ exist, then $\chi_{L_{\Sigma_3^p}}(x) = \chi_C(\langle x, \#, \# \rangle)$. On the other hand, if m-hard strings for length $q_1(|x|)$ and $q_2(|x|)$ exist it might well be the case that for the maximal hard sequences s_1 and s_2, $\chi_{L_{\Sigma_3^p}}(x) = \chi_C(\langle x, s_1, s_2 \rangle) \neq \chi_C(\langle x, \#, \# \rangle)$. But instead of determining $\chi_C(\langle x, s_1, s_2 \rangle)$ we compute $\chi_C(\langle x, \#, \# \rangle)$ and determine whether $\chi_C(\langle x, \#, \# \rangle) \neq \chi_C(\langle x, s_1, s_2 \rangle)$ or not. With this knowledge it is then easy to compute $\chi_{L_{\Sigma_3^p}}(x)$. This approach is known as the mind change technique.

Define $Q_0 = \{x \mid \langle x, \langle \# \rangle, \langle \# \rangle \rangle \in C\}$. Define for $1 \leq j$,

$Q_j = \{x \mid$ there exist $r_1, s_1, r_2, s_2, \ldots, r_j, s_j$ such that

1. for all $1 \leq i \leq j$, r_i and s_i are hard sequences for length $q_1(|x|)$ and $q_2(|x|)$, respectively,
2. for all $1 \leq i \leq j-1$, (r_{i+1}, s_{i+1}) is a proper extension of (r_i, s_i), and
3. $\chi_C(\langle x, \langle \# \rangle, \langle \# \rangle \rangle) \neq \chi_C(\langle x, \langle r_1 \rangle, \langle s_1 \rangle \rangle$ and for all $1 \leq i \leq j-1$, $\chi_C(\langle x, \langle r_i \rangle, \langle s_i \rangle \rangle) \neq \chi_C(\langle x, \langle r_{i+1} \rangle, \langle s_{i+1} \rangle \rangle)\}$.

Observe that $Q_0 \in$ NP and $Q_j \in \Sigma_2^p$, $j \geq 1$. It follows from the definition of Q_j, $j \geq 1$ that $Q_1 \supseteq Q_2 \supseteq \cdots$. Since all hard sequences have order at most $m-1$ (and thus $\ell_1(n) \leq m-1$ and $\ell_2(n) \leq m-1$) and part 3 of the definition of Q_j requires $(r_1, s_1) \neq (\#, \#)$ we obtain that for all $j > 2m-2$, $Q_j = \emptyset$.

Let $x \in \Sigma^*$. Let c be the largest i such that $x \in Q_i$. Observe that $\chi_C(\langle x, \#, \# \rangle) \neq \chi_{Q_c}(x)$ if and only if c is odd. Hence,

$$x \in L_{\Sigma_3^p} \iff \chi_C(\langle x, \#, \# \rangle) + c \equiv 1 \pmod{2}.$$

But note that $c \pmod{2} \equiv 1 \pmod{2}$ if and only if $x \in Q_1 - (Q_2 - (\cdots - (Q_{2m-3} -$

$Q_{2m-2})\cdots))$. It follows that for all $x \in \Sigma^*$,

$$x \in L_{\Sigma_3^{\mathrm{p}}} \iff x \in Q_0 \Delta(Q_1 - (Q_2 - (\cdots - (Q_{2m-3} - Q_{2m-2})\cdots))).$$

This shows $L_{\Sigma_3^{\mathrm{p}}} \in \mathrm{NP}\mathbf{\Delta}\mathrm{DIFF}_{2m-2}(\Sigma_2^{\mathrm{p}})$.

H Applying the mind change technique to the result of Claim G while exploiting Claim E yields $L_{\Sigma_3^{\mathrm{p}}} \in \mathrm{BH}_{2m-1}\mathbf{\Delta}\mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$.

Claim H: $\mathrm{PH} \subseteq \mathrm{BH}_{2m-1}\mathbf{\Delta}\mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$.

In light of Claim G and the fact that $\mathrm{NP}\mathbf{\Delta}\mathrm{DIFF}_{2m-2}(\Sigma_2^{\mathrm{p}}) \subseteq \mathrm{DIFF}_{2m-1}(\Sigma_2^{\mathrm{p}})$, it suffices to show $\mathrm{DIFF}_{2m-1}(\Sigma_2^{\mathrm{p}}) \subseteq \mathrm{BH}_{2m-1}\mathbf{\Delta}\mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$.

Let $L \in \mathrm{DIFF}_{2m-1}(\Sigma_2^{\mathrm{p}})$, hence there exist sets $L_1, L_2, \ldots, L_{2m-1} \in \Sigma_2^{\mathrm{p}}$ such that $L = L_1 - (L_2 - (\cdots - (L_{2m-2} - L_{2m-1})\cdots))$. According to Claim E (note that Claim E can be easily extended to hold for all Σ_2^{p} languages and not just for $L_{\Sigma_2^{\mathrm{p}}}$) there exist sets $B_1, B_2, \ldots, B_{2m-1} \in \mathrm{NP}$ and polynomials $q'_1, q'_2, \ldots, q'_{2m-1}$ such that for all $x \in \Sigma^*$ and all $1 \le i \le 2m-1$, if $\omega^i_m, \omega^i_{m-1}, \ldots, \omega^i_{\ell_i}$ is a maximal hard sequence for length l_i, $l_i \ge q'_i(|x|)$, then

$$x \in L_i \iff \langle x, 1^{l_i}, \omega^i_m, \omega^i_{m-1}, \ldots, \omega^i_{\ell_i}\rangle \in B_i.$$

Let $\widehat{p}$ be a polynomial such that $\widehat{p}(n) \ge q'_i(n)$ for all n and all $1 \le i \le 2m-1$. Define $D = \{\langle x, \omega_m, \omega_{m-1}, \ldots, \omega_\ell\rangle \mid \langle x, 1^{\widehat{p}(|x|)}, \omega_m, \omega_{m-1}, \ldots, \omega_\ell\rangle \in B_1 - (B_2 - (\cdots - (B_{2m-2} - B_{2m-1})\cdots))\}$. Note that $D \in \mathrm{BH}_{2m-1}$. We have for all $x \in \Sigma^*$, if $\omega_m, \omega_{m-1}, \ldots, \omega_\ell$ is a maximal hard sequence for length $\widehat{p}(|x|)$ then

$$x \in L \iff \langle x, \omega_m, \omega_{m-1}, \ldots, \omega_\ell\rangle \in D.$$

Now we use a similar idea as in the proof of Claim G. In particular, recall the definitions from the beginning of its proof. Define $P_0 = \{x \mid \langle x, \#\rangle \in D\}$. Define for $j \ge 1$,

$P_j = \{x \mid$ there exist $s_1, s_2, \ldots, s_j$ such that

1. for all $1 \le i \le j$, s_i is a hard sequence for length $\widehat{p}(|x|)$,
2. for all $1 \le i \le j-1$, s_{i+1} is a proper extension of s_i, and
3. $\chi_D(\langle x, \#\rangle) \ne \chi_D(\langle x, s_1\rangle$ and for all $1 \le i \le j-1$, $\chi_D(\langle x, s_i\rangle) \ne \chi_D(\langle x, s_{i+1}\rangle)\}$.

Note that $P_0 \in \mathrm{BH}_{2m-1}$ and $P_j \in \Sigma_2^{\mathrm{p}}$, $j \ge 1$. Since all hard sequences have order at most $m-1$ and part 3 of the definition of P_j requires $s_1 \ne \#$ we obtain that for all $j > m-1$, $P_j = \emptyset$. Similar to the proof of Claim G it is not hard to verify that for all $x \in \Sigma^*$,

$$x \in L \iff x \in P_0 \Delta(P_1 - (P_2 - (\cdots - (P_{m-2} - P_{m-1})\cdots))).$$

Hence $L \in \mathrm{BH}_{2m-1}\mathbf{\Delta}\mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$. This completes the proof of Claim H.

I Applying the mind change technique again, this time to the result of Claim H, gives the claim of the theorem being proven.

Claim I: $\mathrm{PH} \subseteq \mathrm{BH}_m \mathbf{\Delta} \mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$.

Observe that $\mathrm{BH}_{2m-1}\mathbf{\Delta}\mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}}) \subseteq \mathrm{DIFF}_m(\Sigma_2^{\mathrm{p}})$. In light of Claim H, it suffices to show $\mathrm{DIFF}_m(\Sigma_2^{\mathrm{p}}) \subseteq \mathrm{BH}_m\mathbf{\Delta}\mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$ which can be done quite analogous to the proof of Claim H.

∎

3.3.2 Concluding Remarks

At the end of this chapter we will make a few remarks on Theorem 3.3.1. Recall that Chang and Kadin [CK96](see Theorem 3.2.4) showed that for all $m \geq 2$,

$$\mathrm{BH}_m = \mathrm{coBH}_m \implies \mathrm{DIFF}_m(\Sigma_2^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_2^{\mathrm{p}}) = \mathrm{PH}.$$

Chang and Kadin concluded from that result that there is a close connection between the boolean hierarchy and the boolean hierarchy over Σ_2^{p}. Relatedly, they asked whether there is a straightforward argument showing that $\mathrm{BH}_m = \mathrm{coBH}_m \implies \mathrm{DIFF}_m(\Sigma_2^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_2^{\mathrm{p}})$. Though the result of Chang and Kadin has later been improved by Beigel, Chang, and Ogihara [BCO93] our Theorem 3.3.1 sheds new light on this question. Observe that it is quite obvious that $\mathrm{BH}_m = \mathrm{coBH}_m = \mathrm{BH}$ implies $\mathrm{BH}_m\mathbf{\Delta}\mathrm{DIFF}_j(\Sigma_2^{\mathrm{p}}) = \mathrm{co}(\mathrm{BH}_m\mathbf{\Delta}\mathrm{DIFF}_j(\Sigma_2^{\mathrm{p}}))$ for all $j \geq 1$, in particular, for $j = m-1$. However, this rather trivial observation does not help in proving Theorem 3.3.1. In fact, the proof consists exactly in showing that $\Sigma_3^{\mathrm{p}} \subseteq \mathrm{BH}_m\mathbf{\Delta}\mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$.

It is not hard to see that the proof of Theorem 3.3.1 can be easily extended to yield a similar result for a collapse of the boolean hierarchy over some Σ_k^{p}, $k \geq 1$.

Corollary 3.3.2 For all $m \geq 2$ and all $k \geq 1$,

$$\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_k^{\mathrm{p}}) \implies \mathrm{PH} = \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})\mathbf{\Delta}\mathrm{DIFF}_{m-1}(\Sigma_{k+1}^{\mathrm{p}}).$$

To generalize this result even further, one can of course extend this result from Σ_k^{p} to arbitrary complexity classes $\mathcal{C}$ having certain properties, such as, for instance, being closed under $\exists$ and conjunctive-truth-table reductions and having many-one complete sets.

In light of the intertwined inclusion structure of the boolean hierarchy and the bounded-truth-table hierarchy, Theorem 3.3.1 also establishes a previously unknown collapse of the polynomial hierarchy if the bounded-truth-table hierarchy collapses.

Corollary 3.3.3 For all $m \geq 1$,

$$\mathrm{P}_{m\text{-tt}}^{\mathrm{NP}} = \mathrm{P}_{m+1\text{-tt}}^{\mathrm{NP}} \implies \mathrm{PH} = \mathrm{BH}_{m+1}\mathbf{\Delta}\mathrm{DIFF}_m(\Sigma_2^{\mathrm{p}}).$$

Clearly, an analogous result for the bounded-truth-table hierarchy over Σ_k^{p}, $k \geq 1$, does also hold. However, Hemaspaandra, Hemaspaandra, and Hempel [HHH97b] have shown the following downward translation of equality (see Theorem 4.2.4): For all $m \geq 1$ and all $k > 1$, $\mathrm{P}_{m\text{-tt}}^{\Sigma_k^{\mathrm{p}}} = \mathrm{P}_{m+1\text{-tt}}^{\Sigma_k^{\mathrm{p}}} \implies \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_k^{\mathrm{p}})$. Combining this result with Corollary 3.3.2 we obtain a improvement over a straightforward generalization of Corollary 3.3.3 for Σ_k^{p}, $k > 1$.

Theorem 3.3.4 For all $m \geq 2$ and all $k > 1$,

$$\mathrm{P}_{m\text{-tt}}^{\Sigma_k^{\mathrm{p}}} = \mathrm{P}_{m+1\text{-tt}}^{\Sigma_k^{\mathrm{p}}} \implies \mathrm{PH} = \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) \boldsymbol{\Delta} \mathrm{DIFF}_{m-1}(\Sigma_{k+1}^{\mathrm{p}}).$$

Observe that for $m = 1$ the hypothesis of the above Theorem 3.3.4 even implies $\mathrm{PH} = \Sigma_k^{\mathrm{p}}$ (see Theorem 4.2.1 and Theorem 4.2.3).

To the end, we mention that there is an alternative way of proving Theorem 3.3.1. A close inspection of the proof of Beigel, Chang, and Ogihara [BCO93] shows that what actually being shown is the following result:

$$\mathrm{BH}_m = \mathrm{coBH}_m \implies \mathrm{PH} = \Delta_2^{\mathrm{p}} \boldsymbol{\Delta} \mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}}).$$

This together with a recent result of Chang [Cha], namely,

$$\mathrm{BH}_m = \mathrm{coBH}_m \implies \Delta_2^{\mathrm{p}} = \mathrm{BH}_m,$$

proves Theorem 3.3.1. However, as our proof of Theorem 3.3.1 shows, Theorem 3.3.1 can be proven with only the tools provided by the papers we have reviewed in Section 3.2.

Chapter 4

Downward Collapse

4.1 Introduction

Does the collapse of lower complexity classes imply a collapse of higher complexity classes? Does the collapse of higher complexity classes cause lower complexity classes to collapse? These question are known as upward and downward collapse, respectively. As a special variant of linking collapses of complexity classes, proving downward and upward collapse is a very elegant and powerful method to tie together the relative powers of various computation models.

Usually, hierarchies in complexity theory that are inductively defined in a bottom-up manner display upward collapse, for instance, the polynomial hierarchy. Examples for upward collapse have long been known, one prominent example being $\mathrm{P} = \mathrm{NP} \implies \mathrm{PH} = \mathrm{P}$ [MS72]. Also, all results mentioned and proven in Chapter 3 are examples of upward collapse, for instance, for all $m \geq 2$, if $\mathrm{BH}_m = \mathrm{coBH}_m$ then $\mathrm{PH} = \mathrm{BH}_m \boldsymbol{\Delta} \mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$.

In contrast, downward collapse results are rarely observed. Though one can find a number of examples with the general flavor of downward collapse in the literature (equivalent notions are downward translation of equality and upward separation) most of them are not quite satisfying with respect to the term "downward." Some results are limited to sparse sets only [Boo74, HIS85, RRW94], while others contain unspecified parameters [Wra77]. Clearly, for complexity classes $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{D}_1$, and $\mathcal{D}_2$ such that $\mathcal{C}_1 \cup \mathcal{C}_2 \subseteq \mathcal{D}_1 \cap \mathcal{D}_2$ and it is not known that $\mathcal{C}_1 \cup \mathcal{C}_2 = \mathcal{D}_1 \cap \mathcal{D}_2$, a result $\mathcal{D}_1 = \mathcal{D}_2 \implies \mathcal{C}_1 = \mathcal{C}_2$ would describe exactly what one might expect from the notion of downward collapse. However, it is known that many potential downward collapse results (in the above sense) require non relativizable proof techniques and are quite unlikely to hold. For instance, it is known that $\mathrm{BH}_4 = \mathrm{coBH}_4 \implies \mathrm{NP} = \mathrm{coNP}$ does not hold in all relativized worlds [CGH+88].

In 1996 Hemaspaandra, Hemaspaandra, and Hempel [HHH96a] obtained the first downward collapse result (in the sense as formally defined above) linking classes of bounded-query hierarchies and classes of the polynomial hierarchy and triggered a recent outburst of downward collapse results [BF96, HHH99, HHH97b]. It was shown in [HHH96a] that if one Turing query to a Σ_k^{p} oracle, $k > 2$, is as powerful as two Turing queries to a Σ_k^{p} oracle then

Σ_k^p is closed under complementation and hence as powerful as the entire polynomial hierarchy. Buhrman and Fortnow [BF98] extended this result to the $k = 2$ case. Hemaspaandra, Hemaspaandra, and Hempel [HHH99] showed that a m versus $m+1$ queries analogue does also hold, displaying downward collapse between the levels of the bounded-truth-table hierarchy and the boolean hierarchy over Σ_k^p, $k > 2$. In [HHH97b] the approaches of [BF98] and [HHH99] were combined with new techniques to extend the results of both papers establishing the $k = 2$ analogue of the main result from [HHH99]. Though all of the followup results are based on the proof technique used in [HHH96a], which itself is a modification of Kadin's easy-hard technique [Kad88] (see also Section 3.2), each new downward collapse result added some crucial and elegant enrichment to the proof technique itself.

In this chapter we will review this interesting development and prove a new downward collapse result. The chapter is divided into two parts. Quite similar to the structure of Chapter 3 the first part, Section 4.2, is devoted to a detailed analysis of the previous work. In Section 4.3 we prove the main theorem of this chapter, Theorem 4.3.3, which strengthens a result from [HHH97b] and removes an asymmetry in that result's hypothesis. In particular, we show that for all $s, m \geq 1$ and all $0 < i < k-1$, if $\mathrm{DIFF}_s(\Sigma_i^p)\mathbf{\Delta}\mathrm{DIFF}_m(\Sigma_k^p)$ is closed under complementation, then $\mathrm{DIFF}_m(\Sigma_k^p) = \mathrm{coDIFF}_m(\Sigma_k^p)$. The Chapter closes with a number of remarks and applications of Theorem 4.3.3.

4.2 A Review

After the recent flow of downward collapse results we feel that the time is right to pause and look back; what has been achieved, what technical advances have been made? Doing exactly that we review the following four papers in this section:

1. E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. An Upward Separation in the Polynomial Hierarchy. Technical Report Math/Inf/96/15, Institut für Informatik, Friedrich-Schiller-Universität Jena, Jena, Germany, June 1996.

2. E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. A Downward Collapse Within the Polynomial Hierarchy. *SIAM Journal on Computing.* To appear.

3. H. Buhrman and L. Fortnow. Two queries. In *Proceedings of the 13th Annual IEEE Conference on Computational Complexity.* IEEE Computer Society Press, June 1998. To appear.

4. E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. Translating Equality Downwards. Technical Report TR-657, Department of Computer Science, University of Rochester, Rochester, NY, April 1997.

Similar to Section 3.2 we first state the main results obtained, second informally discuss the development of the proof technique while emphasizing the new contributions made in each of papers under investigation, and third rigorously prove a special case of each paper's main theorems.

4.2.1 Previous Results

We give a short overview over the results obtained in each of the four papers we are going to review. For completeness we added the appropriate citations of the earliest and most recent versions.

Hemaspaandra, Hemaspaandra, and Hempel 1996 [HHH96a] Motivated by the question of whether certain query order classes (see Chapter 5 for results in this interesting new area in complexity theory) are distinct and whether the collapse of those query order classes induces a collapse of the polynomial hierarchy, Hemaspaandra, Hemaspaandra, and Hempel came up with a surprising downward collapse result. A collapse of the bounded-query hierarchy over Σ_k^{p}, $k > 2$, at its first level implies a collapse of the polynomial hierarchy to Σ_k^{p} itself; informally, the polynomial hierarchy collapses to a level that is below the level of the bounded-query hierarchy at which the initial collapse occurred. This was the the first downward collapse result completely within the polynomial hierarchy.

Theorem 4.2.1 [HHH96a] For $k > 2$,

$$\mathrm{P}^{\Sigma_k^{\mathrm{p}}[1]} = \mathrm{P}^{\Sigma_k^{\mathrm{p}}[2]} \implies \mathrm{PH} = \Sigma_k^{\mathrm{p}} = \Pi_k^{\mathrm{p}}.$$

Hemaspaandra, Hemaspaandra, and Hempel 1996 [HHH96b, HHH99] Generalizing the ideas developed in [HHH96a], the authors extended their results to hold in a similar fashion also for m versus $m+1$ queries. It turned out that a collapse of the bounded-truth-table hierarchy over Σ_k^{p}, $k > 2$, implies a collapse of the boolean hierarchy over Σ_k^{p} one level below the trivially implied collapse of the boolean hierarchy. More precisely, a collapse of the bounded-truth-table hierarchy over Σ_k^{p} at level m implies a collapse of the boolean hierarchy over Σ_k^{p} at level m.

Theorem 4.2.2 [HHH96b, HHH99] For all $m \geq 1$ and all $k > 2$,

$$\mathrm{P}^{\Sigma_k^{\mathrm{p}}}_{m\text{-tt}} = \mathrm{P}^{\Sigma_k^{\mathrm{p}}}_{m+1\text{-tt}} \implies \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_k^{\mathrm{p}}).$$

Observe that the result of Theorem 4.2.1 now appears in a different light. Theorem 4.2.1 might give the impression that the bounded-query hierarchy over Σ_k^{p}, $k > 2$, is tightly connected to Σ_k^{p} itself. But Theorem 4.2.2 clarifies that this is a mere coincidence since on one hand the first level of the bounded-query hierarchy over Σ_k^{p} and the first level of the bounded-truth-table hierarchy over Σ_k^{p} coincide, and on the other hand, the first level of the boolean hierarchy over Σ_k^{p} happens to be the kth level of the polynomial hierarchy.

Theorem 4.2.2 together with the result of Theorem 3.3.1, yields also a collapse of the polynomial hierarchy. In Theorem 3.3.4 we have stated that for all $m \geq 2$ (for the $m = 1$ case see Theorem 4.2.1) and all $k > 2$,

$$\mathrm{P}^{\Sigma_k^{\mathrm{p}}}_{m\text{-tt}} = \mathrm{P}^{\Sigma_k^{\mathrm{p}}}_{m+1\text{-tt}} \implies \mathrm{PH} = \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})\boldsymbol{\Delta}\mathrm{DIFF}_{m-1}(\Sigma_{k+1}^{\mathrm{p}}).$$

Note that the collapse of the polynomial hierarchy occurs, roughly speaking, one level lower in the boolean hierarchy over $\Sigma_{k+1}^{\mathrm{p}}$ than could be concluded from the same hypothesis without Theorem 4.2.2.

Buhrman and Fortnow 1996 [BF96, BF98] Buhrman and Fortnow extended the result of Theorem 4.2.1 to the $k = 2$ case; they proved that if one query to a Σ_2^{p} oracle is as powerful as two queries to a Σ_2^{p} oracle then Σ_2^{p} is closed under complementation, establishing a downward collapse result in the second level of the polynomial hierarchy.

Theorem 4.2.3 [BF96] If $\mathrm{P}^{\Sigma_2^{\mathrm{p}}[1]} = \mathrm{P}^{\Sigma_2^{\mathrm{p}}[2]}$ then $\mathrm{PH} = \Sigma_2^{\mathrm{p}} = \Pi_2^{\mathrm{p}}$.

Hemaspaandra, Hemaspaandra, and Hempel 1997 [HHH97b] In an attempt to also extend the result of Theorem 4.2.2 to the $k = 2$ case Hemaspaandra, Hemaspaandra, and Hempel [HHH97b] combined the approaches of [HHH99] and [BF98] with new ideas to obtain a result that implies Theorems 4.2.1, 4.2.2, 4.2.3, and more.

Theorem 4.2.4 [HHH97b] For all $m \geq 1$ and all $k > 1$,

$$\mathrm{P}_{m\text{-tt}}^{\Sigma_k^{\mathrm{p}}} = \mathrm{P}_{m+1\text{-tt}}^{\Sigma_k^{\mathrm{p}}} \implies \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_k^{\mathrm{p}}).$$

This is a very general downward collapse result, as the mth level of the boolean hierarchy over Σ_k^{p} is contained in $\mathrm{P}_{m\text{-tt}}^{\Sigma_k^{\mathrm{p}}}$. In light of Theorem 3.3.1, the above Theorem 4.2.4 also gives a collapse of the polynomial hierarchy that was previously unknown to hold (see Theorem 3.3.4); for all $m \geq 2$ and all $k > 1$,

$$\mathrm{P}_{m\text{-tt}}^{\Sigma_k^{\mathrm{p}}} = \mathrm{P}_{m+1\text{-tt}}^{\Sigma_k^{\mathrm{p}}} \implies \mathrm{PH} = \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) \boldsymbol{\Delta} \mathrm{DIFF}_{m-1}(\Sigma_{k+1}^{\mathrm{p}}).$$

For the $m = 1$ case see Theorems 4.2.1 and 4.2.3.

4.2.2 The Development of the Proof Method

Common in all of the above theorem's proofs is the use of the easy-hard technique. Originally invented by Kadin [Kad88] and applied for proving a polynomial hierarchy collapse from a collapse of the boolean hierarchy, it has been (modified as needed) crucial in proving the downward collapse results we study.

Applications of the easy-hard technique in its original version (see Section 3.2) require the search for hard sequences (strings). Not just the mere information about the existence or nonexistence of hard sequences was exploited in proving (conditional) collapses of the polynomial hierarchy but rather the hard sequences itself. The line of research that has been reviewed in Chapter 3 has led to more and more sophisticated methods to gain information about the hard sequences and to eventually compute them. The fact that all results of Chapter 3 are upward collapse results is more then everything else due to the common need of hard sequences.

Hemaspaandra, Hemaspaandra, and Hempel [HHH96a] observed that in certain settings every string is exactly one of the following, either hard or easy. Thus, one can completely discard the search for hard strings. One simply looks at the input itself, determines whether it is an easy or hard string, and completes the computation in one or the other way. This approach led to the result

$$\mathrm{P}^{\Sigma_k^{\mathrm{p}}[1]} = \mathrm{P}^{\Sigma_k^{\mathrm{p}}[2]} \implies \Sigma_k^{\mathrm{p}} = \Pi_k^{\mathrm{p}} = \mathrm{PH}$$

for $k > 2$ [HHH96a]. In [HHH99] and [BF98] this result is extended in two different directions. The approach of [HHH96a] works only for 1-vs-2 query access to Σ_k^{p}, $k > 2$, where the easy-hard test for the input is responsible for the $k > 2$ bound. Buhrman and Fortnow [BF98] extended the result to the 1-vs-2 query case for $k = 2$. They did so in modifying the test of whether the input is easy or hard to nondeterministically simply assume both that the input is easy and that the input is hard. Since in any case one of the nondeterministic computation branches starts from the wrong assumption it has to be guaranteed that the branch starting from the wrong assumption does no harm to the overall algorithm. This is achieved by shielding each of the nondeterministic branches against falsely accepting the input. [HHH99], on the other hand, removes the 1-vs-2 restriction. Since languages from $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$ are accepted by a collection of nondeterministic machines one has to ensure that their computations follow a common scheme. This is achieved by implementing "0-bit communication" between machines by having the machines independently latch onto a certain lexicographically extreme string signaled by the input.

However, recall that the two improvements just mentioned—from $k > 2$ to $k > 1$ via [BF98] and from 1-vs-2 to m-vs-$m+1$ via [HHH99]—are incomparable. Neither paper allows both improvements to work simultaneously. However, this was achieved in [HHH97b], via a new twist. [HHH97b] provides an improved way of allowing the underlying Σ_k^{p} machines of $\mathrm{DIFF}(\Sigma_k^{\mathrm{p}})$ languages to work together. In particular, [HHH97b] does so by exploiting the so-called telescoping normal form of boolean (or difference) hierarchies [CGH$^+$88]—a normal form that in concept dates as far back as the work of Hausdorff [Hau14].

4.2.3 A Detailed Analysis of the Proofs

The development of the proof method underlying the Theorems 4.2.1, 4.2.2, 4.2.3, and 4.2.4 can best be seen by proving (in its original version) a special case of each of the before mentioned theorems. Starting from an assumption, $\mathrm{P}^{\Sigma_3^{\mathrm{p}}[1]} = \mathrm{P}^{\Sigma_3^{\mathrm{p}}[2]}$, $\mathrm{P}^{\Sigma_3^{\mathrm{p}}[2]} = \mathrm{P}^{\Sigma_3^{\mathrm{p}}[3]}$, $\mathrm{P}^{\Sigma_2^{\mathrm{p}}[1]} = \mathrm{P}^{\Sigma_2^{\mathrm{p}}[2]}$, or $\mathrm{P}^{\Sigma_2^{\mathrm{p}}[2]} = \mathrm{P}^{\Sigma_2^{\mathrm{p}}[3]}$, respectively, we will prove the downward collapse result obtained and also state the induced collapse of the polynomial hierarchy which follows from Theorem 3.3.1.

For simplicity of representation let s be a polynomial such that for all $x, y, z \in \Sigma^*$, $|\langle x, y\rangle| \leq s(\max\{|x|, |y|\})$ and $|\langle x, y, z\rangle| \leq s(\max\{|x|, |y|, |z|\})$. Furthermore, let us renew the convention on polynomials already stated in Section 3.2. Whenever we talk about polynomials in the remainder of this section let us assume that those polynomials are of the form $n^a + b$ for some integers $a, b > 0$. This convention can be made without loss of generality in our context and has the advantage that a polynomial p now satisfies $p(n+1) > p(n) > n$ for all n, a condition we will need throughout this section's proofs.

Hemaspaandra, Hemaspaandra, and Hempel 1996 [HHH96a]

Theorem 4.2.5 If $\mathrm{P}^{\Sigma_3^{\mathrm{p}}[1]} = \mathrm{P}^{\Sigma_3^{\mathrm{p}}[2]}$ then $\Sigma_3^{\mathrm{p}} = \Pi_3^{\mathrm{p}} = \mathrm{PH}$.

Proof: Main Claim *If* $\mathrm{P}^{(\mathrm{P},\Sigma_3^{\mathrm{p}})} = \mathrm{P}^{(\mathrm{NP},\Sigma_3^{\mathrm{p}})}$ *then* $\Sigma_3^{\mathrm{p}} = \Pi_3^{\mathrm{p}}$.

Recall that $\mathrm{P}^{(\mathrm{P},\Sigma_3^{\mathrm{p}})}$ and $\mathrm{P}^{(\mathrm{NP},\Sigma_3^{\mathrm{p}})}$ are the classes of languages that can be accepted by some DPTM making *in parallel* at most one query to a P or NP oracle, respectively, and at most one query to a Σ_3^{p} oracle (see Section 2.4).

Since $\mathrm{P}^{\Sigma_3^{\mathrm{p}}[1]} = \mathrm{P}^{(\mathrm{P},\Sigma_3^{\mathrm{p}})} \subseteq \mathrm{P}^{(\mathrm{NP},\Sigma_3^{\mathrm{p}})} \subseteq \mathrm{P}^{\Sigma_3^{\mathrm{p}}[2]}$ the theorem follows immediately from the above claim. Thus, it remains to prove **Main Claim**.

Proof of Main Claim: **A** Suppose $\mathrm{P}^{(\mathrm{P},\Sigma_3^{\mathrm{p}})} = \mathrm{P}^{(\mathrm{NP},\Sigma_3^{\mathrm{p}})}$. Let L_{P} and $L_{\mathrm{P}^{\mathrm{NP}[1]}}$ be many-one complete languages for $\mathrm{P}^{\mathrm{P}[1]} = \mathrm{P}$ and $\mathrm{P}^{\mathrm{NP}[1]}$, respectively. Let $L_{\Sigma_3^{\mathrm{p}}}$ be a Σ_3^{p} complete language. In order to prove **Main Claim** it suffices to give a Σ_3^{p} algorithm for $\overline{L_{\Sigma_3^{\mathrm{p}}}}$.

B *Claim B:* $L_{\mathrm{P}}\tilde{\Delta}L_{\Sigma_3^{\mathrm{p}}}$ *and* $L_{\mathrm{P}^{\mathrm{NP}[1]}}\tilde{\Delta}L_{\Sigma_3^{\mathrm{p}}}$ *are many-one complete languages for* $\mathrm{P}^{(\mathrm{P},\Sigma_3^{\mathrm{p}})}$ *and* $\mathrm{P}^{(\mathrm{NP},\Sigma_3^{\mathrm{p}})}$*, respectively.*

We only show the claim for $L_{\mathrm{P}^{\mathrm{NP}[1]}}\tilde{\Delta}L_{\Sigma_3^{\mathrm{p}}}$. Recall that for any two sets A and B, $A\tilde{\Delta}B = \{\langle x,y\rangle \mid x \in A \iff y \notin B\}$. Obviously, $L_{\mathrm{P}^{\mathrm{NP}[1]}}\tilde{\Delta}L_{\Sigma_3^{\mathrm{p}}} \in \mathrm{P}^{(\mathrm{NP},\Sigma_3^{\mathrm{p}})}$. Let L be an arbitrary language from $\mathrm{P}^{(\mathrm{NP},\Sigma_3^{\mathrm{p}})}$. Without loss of generality let L be accepted by a DPTM N making, on every input x, in parallel exactly one query x_A to A and one query x_B to B, where $A \in \mathrm{NP}$ and $B \in \Sigma_3^{\mathrm{p}}$. Hence $L = L(N^{(A,B)})$. Define

$C = \{x \mid N^{(A,B)}(x)$ accepts if x_A is answered correctly and x_B is answered no$\}$ and

$D = \{x \mid$ the acceptance/rejection behavior of $N^{(A,B)}(x)$ after answering the query x_A correctly strictly depends on the answer to x_B, and $x_B \in B\}$.

Note that the set D can also be seen as the set of all x such that $x_B \in B$ and the partial truth-table of $N^{(A,B)}(x)$ with respect to a correct answer to x_A has at least one mind change. Clearly, $C \in \mathrm{P}^{\mathrm{NP}[1]}$ and $D \in \Sigma_3^{\mathrm{p}}$. Furthermore, it is not hard to verify that for all $x \in \Sigma^*$,

$$x \in L \iff \langle x,x\rangle \in C\tilde{\Delta}D.$$

But note that we also have for all $x \in \Sigma^*$,

$$\langle x,x\rangle \in C\tilde{\Delta}D \iff \langle f(x),g(x)\rangle \in L_{\mathrm{P}^{\mathrm{NP}[1]}}\tilde{\Delta}L_{\Sigma_3^{\mathrm{p}}},$$

where f and g are polynomial-time computable functions reducing C to $L_{\mathrm{P}^{\mathrm{NP}[1]}}$ and D to $L_{\Sigma_3^{\mathrm{p}}}$, respectively. This shows that L is many-one reducible to $L_{\mathrm{P}^{\mathrm{NP}[1]}}\tilde{\Delta}L_{\Sigma_3^{\mathrm{p}}}$ which completes the proof of the claim.

C Since by assumption $\mathrm{P}^{(\mathrm{P},\Sigma_3^\mathrm{p})} = \mathrm{P}^{(\mathrm{NP},\Sigma_3^\mathrm{p})}$ we have a many-one reduction from $L_{\mathrm{P}^{\mathrm{NP}[1]}}\tilde{\Delta}L_{\Sigma_3^\mathrm{p}}$ to $L_\mathrm{P}\tilde{\Delta}L_{\Sigma_3^\mathrm{p}}$. In other words, there exists a polynomial-time computable function h such that for all $x_1, x_2 \in \Sigma^*$,

$$\langle x_1, x_2\rangle \in L_{\mathrm{P}^{\mathrm{NP}[1]}}\tilde{\Delta}L_{\Sigma_3^\mathrm{p}} \iff h(\langle x_1, x_2\rangle) \in L_\mathrm{P}\tilde{\Delta}L_{\Sigma_3^\mathrm{p}}.$$

Let h' and h'' be the polynomial-time computable functions such that for all $x_1, x_2 \in \Sigma^*$, $h(\langle x_1, x_2\rangle) = \langle h'(\langle x_1, x_2\rangle), h''(\langle x_1, x_2\rangle)\rangle$ and thus

$$(x_1 \in L_{\mathrm{P}^{\mathrm{NP}[1]}} \iff x_2 \notin L_{\Sigma_3^\mathrm{p}}) \iff (h'(\langle x_1, x_2\rangle) \in L_\mathrm{P} \iff h''(\langle x_1, x_2\rangle) \notin L_{\Sigma_3^\mathrm{p}}).$$

D Let l be an integer. The string x_2 is said to be *easy for length l* if and only if $(\exists x_1 : |x_1| \leq l)[x_1 \in L_{\mathrm{P}^{\mathrm{NP}[1]}} \iff h'(\langle x_1, x_2\rangle) \notin L_\mathrm{P}]$.

x_2 is said to be *hard for length l* if and only if $|x_2| \leq l$ and x_2 is not easy for length l, that is, if and only if $(\forall x_1 : |x_1| \leq l)[x_1 \in L_{\mathrm{P}^{\mathrm{NP}[1]}} \iff h'(\langle x_1, x_2\rangle) \in L_\mathrm{P}]$.

Thus, every string is exactly one of the following, either easy or hard for length l. This observation will be used to divide the problem of giving a Σ_3^p algorithm for $\overline{L_{\Sigma_3^\mathrm{p}}}$ into two sub problems, which we are going to solve in **E** and **F**. Note that testing whether a string x is easy for length $r(|x|)$, where r is some polynomial, can be done by a Σ_2^p algorithm.

E The upcoming Claim E solves the sub problem for the strings being hard for a certain length.

Claim E: There exist a set $A \in \Sigma_3^\mathrm{p}$ and a polynomial q such that for all $x \in \Sigma^*$, if x is hard for length $q(|x|)$ then

$$x \notin L_{\Sigma_3^\mathrm{p}} \iff x \in A.$$

Let p be a polynomial such that for all $x \in \Sigma^*$,

$$x \notin L_{\Sigma_3^\mathrm{p}} \iff (\forall y : |y| \leq p(|x|))(\exists z : |z| \leq p(|x|))[\langle x, y, z\rangle \in L_{\mathrm{P}^{\mathrm{NP}[1]}}].$$

Recall, if x is a hard string for length l, where l is some integer, then

$$(\forall x_1 : |x_1| \leq l)[x_1 \in L_{\mathrm{P}^{\mathrm{NP}[1]}} \iff h'(\langle x_1, x\rangle) \in L_\mathrm{P}].$$

Let q be a polynomial such that $q(n) \geq s(p(n))$ for all n. Suppose that x is a hard string for length $q(|x|)$. Hence for all $y, z \in \Sigma^*$, $|y|, |z| \leq p(|x|)$,

$$\langle x, y, z\rangle \in L_{\mathrm{P}^{\mathrm{NP}[1]}} \iff h'(\langle\langle x, y, z\rangle, x\rangle) \in L_\mathrm{P}$$

and thus

$$x \notin L_{\Sigma_3^\mathrm{p}} \iff (\forall y : |y| \leq p(|x|))(\exists z : |z| \leq p(|x|))[h'(\langle\langle x, y, z\rangle, x\rangle) \in L_\mathrm{P}].$$

Note that $h'(\langle v, x\rangle)$ is computable in time polynomial in $\max\{|v|, |x|\}$. Set

$$A = \{x \mid (\forall y : |y| \leq p(|x|))(\exists z : |z| \leq p(|x|))[h'(\langle\langle x, y, z\rangle, x\rangle) \in L_\mathrm{P}]\}.$$

Note that $A \in \Pi_2^\mathrm{p}$ and thus $A \in \Sigma_3^\mathrm{p}$.

F We now solve the sub problem for the strings x being easy for length $q(|x|)$.

Claim F: Let q be the polynomial defined in **E**. *There exists a set $B \in \Sigma_3^{\mathrm{p}}$ such that for all $x \in \Sigma^*$, if x is easy for length $q(|x|)$ then*

$$x \notin L_{\Sigma_3^{\mathrm{p}}} \iff x \in B.$$

Define $B = \{x \mid \ (\exists x_1 : |x_1| \leq q(|x|))$
$[(x_1 \in L_{\mathrm{P}^{\mathrm{NP}[1]}} \iff h'(\langle x_1, x\rangle) \notin L_{\mathrm{P}}) \wedge h''(\langle x_1, x\rangle) \in L_{\Sigma_3^{\mathrm{p}}}]\}$.

Note that $B \in \Sigma_3^{\mathrm{p}}$. In light of **C** and **D**, this proves the claim.

G Combining Claims E and F with a preliminary test whether the input x is hard or easy for length $q(|x|)$ we obtain a Σ_3^{p} algorithm for $\overline{L_{\Sigma_3^{\mathrm{p}}}}$.

Claim G: $\overline{L_{\Sigma_3^{\mathrm{p}}}} \in \Sigma_3^{\mathrm{p}}$.

Let $A, B \in \Sigma_3^{\mathrm{p}}$ and q be a polynomial, all three as defined in **E** and **F**. In light of Claims E and F, the following algorithm is a Σ_3^{p} algorithm for $\overline{L_{\Sigma_3^{\mathrm{p}}}}$.

1. On input x determine whether the input x is easy or hard for length $q(|x|)$. Recall that this can be done with one Σ_2^{p} oracle query according to **D**.
2. If the input x is hard for length $q(|x|)$ then accept if and only if $x \in A$.
3. If the input x is easy for length $q(|x|)$ then accept if and only if $x \in B$.

Note that the use of different oracles in the above algorithm does not affect its correctness as it can be easily be avoided by using the disjoint union of the oracles instead.

H Since $\overline{L_{\Sigma_3^{\mathrm{p}}}}$ is complete for Π_3^{p} we have shown $\Pi_3^{\mathrm{p}} = \Sigma_3^{\mathrm{p}}$ and hence $\mathrm{PH} = \Sigma_3^{\mathrm{p}}$.

End of Proof of Main Claim ▮

Hemaspaandra, Hemaspaandra, and Hempel 1996 [HHH96b, HHH99]

Theorem 4.2.6 If $\mathrm{P}_{2\text{-tt}}^{\Sigma_3^{\mathrm{p}}} = \mathrm{P}_{3\text{-tt}}^{\Sigma_3^{\mathrm{p}}}$ then $\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}}) = \mathrm{coDIFF}_2(\Sigma_3^{\mathrm{p}})$.

Proof: **Main Claim** *If* $\mathrm{P}_{1,2\text{-tt}}^{(\mathrm{P},\Sigma_3^{\mathrm{p}})} = \mathrm{P}_{1,2\text{-tt}}^{(\mathrm{NP},\Sigma_3^{\mathrm{p}})}$ *then* $\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}}) = \mathrm{coDIFF}_2(\Sigma_3^{\mathrm{p}})$.

Recall that $\mathrm{P}_{1,2\text{-tt}}^{(\mathrm{P},\Sigma_3^{\mathrm{p}})}$ and $\mathrm{P}_{1,2\text{-tt}}^{(\mathrm{NP},\Sigma_3^{\mathrm{p}})}$ are the classes of languages that are accepted by some DPTM making in parallel at most one query to a P or NP oracle, respectively, and at most two queries to a Σ_3^{p} oracle (see Section 2.4). Since $\mathrm{P}_{2\text{-tt}}^{\Sigma_3^{\mathrm{p}}} = \mathrm{P}_{1,2\text{-tt}}^{(\mathrm{P},\Sigma_3^{\mathrm{p}})} \subseteq \mathrm{P}_{1,2\text{-tt}}^{(\mathrm{NP},\Sigma_3^{\mathrm{p}})} \subseteq \mathrm{P}_{3\text{-tt}}^{\Sigma_3^{\mathrm{p}}}$, the theorem follows immediately from the above claim. Thus, it suffices to show the correctness of the main claim.

Proof of Main Claim: **A** Suppose $\mathrm{P}_{1,2\text{-tt}}^{(\mathrm{P},\Sigma_3^{\mathrm{p}})} = \mathrm{P}_{1,2\text{-tt}}^{(\mathrm{NP},\Sigma_3^{\mathrm{p}})}$. Let L_{P} and $L_{\mathrm{P}^{\mathrm{NP}[1]}}$ be many-one complete languages for $\mathrm{P}^{\mathrm{P}[1]} = \mathrm{P}$ and $\mathrm{P}^{\mathrm{NP}[1]}$, respectively. Let $L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}$ be a $\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})$ complete language. We will give a $\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})$ algorithm for $\overline{L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}}$. Let $L_1, L_2 \in \Sigma_3^{\mathrm{p}}$ such that $L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})} = L_1 - L_2$.

B *Claim B:* $L_{\mathrm{P}}\tilde{\Delta}L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}$ *and* $L_{\mathrm{P}^{\mathrm{NP}[1]}}\tilde{\Delta}L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}$ *are many-one complete languages for* $\mathrm{P}_{1,2\text{-tt}}^{(\mathrm{P},\Sigma_3^{\mathrm{p}})}$ *and* $\mathrm{P}_{1,2\text{-tt}}^{(\mathrm{NP},\Sigma_3^{\mathrm{p}})}$*, respectively.*

The proof is similar to **B** in the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996). But now the $L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}$ part of $L_{\mathrm{P}^{\mathrm{NP}[1]}}\tilde{\Delta}L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}$ accounts essentially for determining if there is *exactly* one mind change in the partial truth-table with respect to a correctly answered NP query.

C Since by assumption $\mathrm{P}_{1,2\text{-tt}}^{(\mathrm{P},\Sigma_3^{\mathrm{p}})} = \mathrm{P}_{1,2\text{-tt}}^{(\mathrm{NP},\Sigma_3^{\mathrm{p}})}$ it follows that there is a many-one reduction h from $L_{\mathrm{P}^{\mathrm{NP}[1]}}\tilde{\Delta}L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}$ to $L_{\mathrm{P}}\tilde{\Delta}L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}$. While continuing as in **C** in the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996) and replacing $L_{\Sigma_3^{\mathrm{p}}}$ by $L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}$ we obtain for all $x_1, x_2 \in \Sigma^*$,

$$(x_1 \in L_{\mathrm{P}^{\mathrm{NP}[1]}} \iff x_2 \notin L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}) \iff$$
$$(h'(\langle x_1, x_2\rangle) \in L_{\mathrm{P}} \iff h''(\langle x_1, x_2\rangle) \notin L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}).$$

D As in the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996).

E As in the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996) we solve the sub problem for the strings being hard for a certain length first.

Claim E: There exist sets $A_1, A_2 \in \Sigma_3^{\mathrm{p}}$ *and a polynomial* q *such that for all* $x \in \Sigma^*$*, if* x *is a hard string for length* $q(|x|)$ *then*

$$x \notin L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})} \iff x \in A_1 - A_2.$$

A straightforward application of the key idea of **E** from the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996) leads to a $\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})$ algorithm to test whether $x \in \overline{L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}}$ if x is a hard string. Without loss of generality let p_1 and p_2 be two polynomials and $f_1, f_2 \in \mathrm{FP}$ be two functions such that for all $x \in \Sigma^*$ and all $i = 1, 2$, $|f_i(x)| \le p_i(|x|)$ and

$$x \in L_i \iff (\exists y : |y| \le p_i(|x|))(\forall z : |z| \le p_i(|x|))[\langle f_i(x), y, z\rangle \in L_{\mathrm{P}^{\mathrm{NP}[1]}}].$$

Let q be a polynomial such that $q(n) \ge \max\{s(p_1(n)), s(p_2(n))\}$ for all n. Let x be hard for length $q(|x|)$. Hence we have for all x_1, $|x_1| \le q(|x|)$, $x_1 \in L_{\mathrm{P}^{\mathrm{NP}[1]}} \iff h'(\langle x_1, x\rangle) \in L_{\mathrm{P}}$. Thus, for all $i = 1, 2$,

$$x \in L_i \iff (\exists y : |y| \le p_i(|x|))(\forall z : |z| \le p_i(|x|))[h'(\langle\langle f_i(x), y, z\rangle, x\rangle) \in L_{\mathrm{P}}].$$

Define for $i = 1, 2$, $A'_i = \{x \mid (\exists y : |y| \leq p_i(|x|))(\forall z : |z| \leq p_i(|x|))[h'(\langle\langle f_i(x), y, z\rangle, x\rangle) \in L_{\mathrm{P}}\}$. Note that $A'_1, A'_2 \in \Sigma_2^{\mathrm{p}}$ and that for all $x \in \Sigma^*$, if x is hard for length $q(|x|)$ then $x \in L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})} \iff x \in A'_1 - A'_2$. Since $\mathrm{coDIFF}_2(\Sigma_2^{\mathrm{p}}) \subseteq \mathrm{P}^{\Sigma_2^{\mathrm{p}}[2]} \subseteq \mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})$ there exist sets $A_1, A_2 \in \Sigma_3^{\mathrm{p}}$ such that $A_1 - A_2 = \overline{A'_1 - A'_2}$. Hence, if x is a hard string for length $q(|x|)$ then

$$x \notin L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})} \iff x \in A_1 - A_2.$$

F Now follows the solution of the sub problem for the strings x being easy for length $q(|x|)$.

Claim F: Let q be the polynomial defined in **E**. *There exist sets $B_1, B_2 \in \Sigma_3^{\mathrm{p}}$ such that for all $x \in \Sigma^*$, if x is an easy string for length $q(|x|)$ then*

$$x \notin L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})} \iff x \in B_1 - B_2.$$

Recall $L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})} = L_1 - L_2$, where $L_1, L_2 \in \Sigma_3^{\mathrm{p}}$. Define for $i = 1, 2$,

$$\begin{aligned} B_i = \{x \mid\ & (\exists x_1 : |x_1| \leq q(|x|))[(x_1 \in L_{\mathrm{P}^{\mathrm{NP}[1]}} \iff h'(\langle x_1, x\rangle) \notin L_{\mathrm{P}}) \wedge \\ & (\forall v : v <_{lex} x_1)[v \in L_{\mathrm{P}^{\mathrm{NP}[1]}} \iff h'(\langle v, x\rangle) \in L_{\mathrm{P}}] \wedge h''(\langle x_1, x\rangle) \in L_i]\}. \end{aligned}$$

Obviously, $B_1, B_2 \in \Sigma_3^{\mathrm{p}}$. In light of **C** and the definition of B_1 and B_2, it is not hard to verify that if x is an easy string for length $q(|x|)$ then

$$x \notin L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})} \iff x \in B_1 - B_2.$$

G Combining the results of **E** and **F** with a preliminary test whether the input x is hard or easy for length $q(|x|)$, we obtain a $\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})$ algorithm for $\overline{L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}}$.

Claim G: $\overline{L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}} \in \mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})$.

Let $A_1, A_2, B_1, B_2 \in \Sigma_3^{\mathrm{p}}$ and q be a polynomial, all as defined in **E** and **F**. For $i = 1, 2$ let $\widehat{L_i}$ be the language accepted by the following algorithm:

1. On input x determine whether the input x is easy or hard for length $q(|x|)$. This can be done with one Σ_2^{p} oracle query according to **D**.
2. If the input x is hard for length $q(|x|)$ then accept if and only if $x \in A_i$.
3. If the input x is easy for length $q(|x|)$ accept if and only if $x \in B_i$.

Clearly, $\widehat{L_1}, \widehat{L_2} \in \Sigma_3^{\mathrm{p}}$. Furthermore, for all $x \in \Sigma^*$, $x \in \overline{L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}} \iff x \in \widehat{L_1} - \widehat{L_2}$ due to Claims E and F. Hence $\overline{L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}} \in \mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})$.

H Since $\overline{L_{\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}})}}$ is complete for $\mathrm{coDIFF}_2(\Sigma_3^{\mathrm{p}})$ we obtain $\mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}}) = \mathrm{coDIFF}_2(\Sigma_3^{\mathrm{p}})$.

End of Proof of Main Claim ∎

In light of Theorem 3.3.1, we have the following collapse of the polynomial hierarchy.

Corollary 4.2.7 If $\mathrm{P}_{2\text{-tt}}^{\Sigma_3^{\mathrm{p}}} = \mathrm{P}_{3\text{-tt}}^{\Sigma_3^{\mathrm{p}}}$ then $\mathrm{PH} = \mathrm{DIFF}_2(\Sigma_3^{\mathrm{p}}) \boldsymbol{\Delta} \Sigma_4^{\mathrm{p}}$.

Buhrman and Fortnow 1996 [BF96]

Theorem 4.2.8 If $\mathrm{P}^{\Sigma_2^p[1]} = \mathrm{P}^{\Sigma_2^p[2]}$ then $\Sigma_2^p = \Pi_2^p = \mathrm{PH}$.

Proof: **Main Claim** *If* $\mathrm{P}^{\Sigma_2^p[1]} = \mathrm{NP}\boldsymbol{\Delta}\Sigma_2^p$ *then* $\Sigma_2^p = \Pi_2^p$.

Recall $\mathrm{NP}\boldsymbol{\Delta}\Sigma_2^p = \{A\Delta B \mid A \in \mathrm{NP} \wedge B \in \Sigma_2^p\}$. Since $\mathrm{P}^{\Sigma_2^p[1]} \subseteq \mathrm{NP}\boldsymbol{\Delta}\Sigma_2^p \subseteq \mathrm{P}^{\Sigma_2^p[2]}$ the theorem follows immediately from the above claim. So we will prove the theorem by proving the main claim.

Proof of Main Claim: **A** Assume $\mathrm{P}^{\Sigma_2^p[1]} = \mathrm{NP}\boldsymbol{\Delta}\Sigma_2^p$. Let L_P and $L_{\Sigma_2^p}$ be complete languages for P and Σ_2^p, respectively. $L_\mathrm{P}\tilde{\Delta}L_{\Sigma_2^p}$ is complete for $\mathrm{P}^{(\mathrm{P},\Sigma_2^p)} = \mathrm{P}^{\Sigma_2^p[1]}$. This can be shown quite analogous to **B** from the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996). Furthermore, observe that $\mathrm{SAT}\tilde{\Delta}L_{\Sigma_2^p} \in \mathrm{NP}\boldsymbol{\Delta}\Sigma_2^p$.

B Since $\mathrm{P}^{\Sigma_2^p[1]} = \mathrm{NP}\boldsymbol{\Delta}\Sigma_2^p$ we have $\mathrm{SAT}\tilde{\Delta}L_{\Sigma_2^p} \in \mathrm{P}^{\Sigma_2^p[1]}$. Consequently, there is a many-one reduction h from $\mathrm{SAT}\tilde{\Delta}L_{\Sigma_2^p}$ to $L_\mathrm{P}\tilde{\Delta}L_{\Sigma_2^p}$. Continuing as in **C** in the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996) while replacing $L_{\Sigma_3^p}$ by $L_{\Sigma_2^p}$ and $L_{\mathrm{P}^{\mathrm{NP}[1]}}$ by SAT yields for all $x_1, x_2 \in \Sigma^*$,

$$(x_1 \in \mathrm{SAT} \iff x_2 \notin L_{\Sigma_2^p}) \iff (h'(\langle x_1, x_2\rangle) \in L_\mathrm{P} \iff h''(\langle x_1, x_2\rangle) \notin L_{\Sigma_2^p}).$$

C As **D** in the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996) but replace $L_{\mathrm{P}^{\mathrm{NP}[1]}}$ by SAT.

D Observe that the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996) is not valid for Σ_2^p instead of Σ_3^p. The crucial point in the proof of Theorem 4.2.5 is the test in the final algorithm whether a string is easy or hard for a certain length (done by a Σ_2^p oracle query). Since the final algorithm of the current proof has to be a Σ_2^p algorithm, one has to avoid the preliminary easy-hard test and thus one needs to shield each of the sub algorithms against falsely accepting.

Claim D: There exist a set $A \in \mathrm{coNP}$ and a polynomial q such that

1. $A \subseteq \overline{L_{\Sigma_2^p}}$ *and*
2. *for all $x \in \Sigma^*$, if x is hard for length $q(|x|)$ then*

$$x \notin L_{\Sigma_2^p} \iff x \in A.$$

Without loss of generality let for all $x \in \Sigma^*$,

$$x \notin L_{\Sigma_2^p} \iff (\forall z : |z| \leq p'(|x|))[\langle x, z\rangle \in \mathrm{SAT}],$$

for some polynomial p'. Define q to be a polynomial such that $q(n) \geq s(p'(n))$ for all n. Assume that x is a hard string for length $q(|x|)$. Hence,

$$(*) \qquad (\forall x_1 : |x_1| \leq q(|x|))[x_1 \in \mathrm{SAT} \iff h'(\langle x_1, x\rangle) \in L_\mathrm{P}].$$

Consider the following coNP algorithm:

1. On input x guess z, $|z| \leq p'(|x|)$.
2. Assume (*) and use the self-reduction of SAT to find a potential witness for $\langle x, z\rangle \in \text{SAT}$ with the help of (*) in deterministic polynomial time. This is done as follows: Let $\langle x, z\rangle$ encode the boolean formula F. We construct ω_F an assignment for F. Fix an ordering for the variables in F. Replacing the first variable in F by 0 (1) leads to a boolean formula F_0 (F_1), let the string v_0 encode F_0. Compute $h(\langle v_0, x\rangle)$ and test whether $h'(\langle v_0, x\rangle) \in L_{\mathrm{P}}$. If $h'(\langle v_0, x\rangle) \in L_{\mathrm{P}}$ then, under assumption (*), $F_0 \in \text{SAT}$. Thus, set the value for the first variable in ω_F to 0 and repeat this procedure with F_0 until ω_F assigns a value to each variable in F. If $h'(\langle v_0, x\rangle) \notin L_{\mathrm{P}}$ then, under assumption (*), $F_0 \notin \text{SAT}$ implying that a satisfying assignment for F, if there exists one, assigns 1 to the first variable. So set the value for the first variable in ω_F to 1 and repeat this procedure with F_1 until ω_F assigns a value to each variable in F.
3. Accept if and only if the string constructed in step 2 is a witness for $\langle x, z\rangle \in \text{SAT}$. In other words, accept if and only if the assignment ω_F constructed in step 2 satisfies F.

Let A be the language accepted by this algorithm, $A \in \text{coNP}$. If x is a hard string for length $q(|x|)$ then (*) in fact holds and it is not hard to verify that

$$x \notin L_{\Sigma_2^{\mathrm{p}}} \iff x \in A.$$

But note that even if x is not a hard string for length $q(|x|)$ we have that $x \in A$ implies $x \notin L_{\Sigma_2^{\mathrm{p}}}$. This follows from the fact that the algorithm only accepts (as it is a coNP algorithm) if for all z, $|z| \leq p(|x|)$, the string constructed in step 2 is a witness for $\langle x, z\rangle \in \text{SAT}$.

E The sub algorithm for the strings x being easy for length $q(|x|)$ given in **F** from the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996) has already the required shielding feature as spoken of in the beginning of **D** and can be easily adapted to the current proof.

Claim E: Let q be the polynomial defined in **D**. *There exists a set $B \in \Sigma_2^{\mathrm{p}}$ such that*

1. *$B \subseteq \overline{L_{\Sigma_2^{\mathrm{p}}}}$ and*
2. *for all $x \in \Sigma^*$, if x is an easy string for length $q(|x|)$ then*

$$x \notin L_{\Sigma_2^{\mathrm{p}}} \iff x \in B.$$

It is not hard to see that Claim E can be shown quite analogous to **F** from the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996) with respect to the obvious adaptions as for instance a replacement of $L_{\Sigma_3^{\mathrm{p}}}$ by $L_{\Sigma_2^{\mathrm{p}}}$ and $L_{\mathrm{P}^{\mathrm{NP}[1]}}$ by SAT. Hence, for the modified set B from **F** in the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996) holds, $x \in B$ if and only if

$(\exists x_1 : |x_1| \leq q(|x|))[(x_1 \in \mathrm{SAT} \iff h'(\langle x_1, x\rangle) \notin L_{\mathrm{P}}) \wedge h''(\langle x_1, x\rangle) \in L_{\Sigma_2^{\mathrm{p}}}]$. Note that $x \in B$ if and only if x is easy for length $q(|x|)$ and $x \notin L_{\Sigma_2^{\mathrm{p}}}$. In particular, if x is hard for length $q(|x|)$ then $x \notin B$.

F Running the sub algorithms from Claims D and E in parallel gives a Σ_2^{p} algorithm for $\overline{L_{\Sigma_2^{\mathrm{p}}}}$.

Claim F: $\overline{L_{\Sigma_2^{\mathrm{p}}}} \in \Sigma_2^{\mathrm{p}}$.

Let $A \in \mathrm{coNP}$, $B \in \Sigma_2^{\mathrm{p}}$, and q be a polynomial, all three as defined in **D** and **E**. We have the following Σ_2^{p} algorithm for $\overline{L_{\Sigma_2^{\mathrm{p}}}}$.

1. On input x guess whether the string x is hard or easy for length $q(|x|)$.
2. If the algorithm has guessed that x is a hard string for length $q(|x|)$ accept if and only if $x \in A$.
3. If "x is an easy string for length $q(|x|)$" was guessed in step 1 then accept if and only if $x \in B$.

Recall from **D** and **E** that the sub algorithm emerging from the wrong guess does not accept if $x \notin \overline{L_{\Sigma_2^{\mathrm{p}}}}$.

G Since $\overline{L_{\Sigma_2^{\mathrm{p}}}}$ is many-one complete for Π_2^{p} we obtain $\Pi_2^{\mathrm{p}} = \Sigma_2^{\mathrm{p}}$ and thus $\mathrm{PH} = \Sigma_2^{\mathrm{p}}$.

End of Proof of Main Claim ▮

Hemaspaandra, Hemaspaandra, and Hempel 1997 [HHH97b]

Theorem 4.2.9 If $\mathrm{P}_{2\text{-tt}}^{\Sigma_2^{\mathrm{p}}} = \mathrm{P}_{3\text{-tt}}^{\Sigma_2^{\mathrm{p}}}$ then $\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}}) = \mathrm{coDIFF}_2(\Sigma_2^{\mathrm{p}})$.

Proof:

Main Claim *If* $\mathrm{P}\boldsymbol{\Delta}\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}}) = \mathrm{NP}\boldsymbol{\Delta}\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})$ *then* $\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}}) = \mathrm{coDIFF}_2(\Sigma_2^{\mathrm{p}})$.

Recall the definition of $\mathcal{C}\boldsymbol{\Delta}\mathcal{D} = \{C \Delta D \mid C \in \mathcal{C} \wedge D \in \mathcal{D}\}$ for complexity classes $\mathcal{C}, \mathcal{D}$ from Chapter 2. Since $\mathrm{P}_{2\text{-tt}}^{\Sigma_2^{\mathrm{p}}} = \mathrm{P}\boldsymbol{\Delta}\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}}) \subseteq \mathrm{NP}\boldsymbol{\Delta}\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}}) \subseteq \mathrm{P}_{3\text{-tt}}^{\Sigma_2^{\mathrm{p}}}$ the theorem follows immediately from the above claim. It remains to prove the main claim.

Proof of Main Claim: **A** Suppose $\mathrm{P}\boldsymbol{\Delta}\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}}) = \mathrm{NP}\boldsymbol{\Delta}\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})$. Let L_{P}, L_{NP}, and $L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}$ be complete languages for P, NP, and $\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})$, respectively. Let $L_1, L_2 \in \Sigma_2^{\mathrm{p}}$ such that $L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})} = L_1 - L_2$.

B *Claim B: The languages* $L_{\mathrm{P}}\tilde{\Delta}L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}$ *and* $L_{\mathrm{NP}}\tilde{\Delta}L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}$ *are many-one complete for* $\mathrm{P}\boldsymbol{\Delta}\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})$ *and* $\mathrm{NP}\boldsymbol{\Delta}\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})$*, respectively.*

The proof is straightforward and thus omitted.

C Since by assumption $\mathrm{P}\boldsymbol{\Delta}\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}}) = \mathrm{NP}\boldsymbol{\Delta}\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})$, $L_{\mathrm{NP}}\tilde{\Delta}L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}$ many-one reduces to $L_{\mathrm{P}}\tilde{\Delta}L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}$. Continue as in **C** in the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996) but replace $L_{\Sigma_3^{\mathrm{p}}}$ by $L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}$ and $L_{\mathrm{P}^{\mathrm{NP}[1]}}$ by L_{NP}. Thus for all $x_1, x_2 \in \Sigma^*$,

$$(x_1 \in L_{\mathrm{NP}} \iff x_2 \notin L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}) \iff (h'(\langle x_1, x_2\rangle) \in L_{\mathrm{P}} \iff h''(\langle x_1, x_2\rangle) \notin L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}).$$

D As **D** in the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996) but replace $L_{\mathrm{P}^{\mathrm{NP}[1]}}$ by L_{NP}.

E As **D** of the proof of Theorem 4.2.8 (Buhrman and Fortnow 1996) but replace $L_{\Sigma_2^{\mathrm{p}}}$ by L_1.

Claim E: There exist a set $A \in \mathrm{coNP}$ and a polynomial q such that

1. *$A \subseteq \overline{L_1}$ and*
2. *for all $x \in \Sigma^*$, if x is hard for length $q(|x|)$ then*

$$x \in \overline{L_1} \iff x \in A.$$

F The result of **E** can be extended to yield a Σ_2^{p} algorithm for the strings x in $\overline{L_{\Sigma_2^{\mathrm{p}}}}$ being hard for length $q(|x|)$ that is protected against accepting if the input string x in fact is an easy string for length $q(|x|)$ and $x \notin \overline{L_{\Sigma_2^{\mathrm{p}}}}$.

Claim F: Let q be the polynomial defined in **E**. *There exists a set $A' \in \Sigma_2^{\mathrm{p}}$ such that*

1. *$A' \subseteq \overline{L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}}$ and*
2. *for all $x \in \Sigma^*$, if x is a hard string for length $q(|x|)$ then*

$$x \notin L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})} \iff x \in A'.$$

Let $L_1, L_2 \in \Sigma_2^{\mathrm{p}}$ and $A \in \mathrm{coNP}$ be as defined in **A** and **E**. Define $A' = A \cup L_2$. Clearly, $A' \in \Sigma_2^{\mathrm{p}}$. Note that

$$\begin{aligned}
x \in A' &\iff x \in A \lor x \in L_2 \\
&\implies x \in \overline{L_1} \lor x \in L_2 \\
&\iff x \in \overline{L_1} \cup L_2 \\
&\iff x \notin L_1 - L_2 \\
&\iff x \in \overline{L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}}.
\end{aligned}$$

Hence $A' \subseteq \overline{L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}}$. Furthermore, in case x is hard for length $q(|x|)$ the second line in the above implication chain turns into an equivalence according to **E**.

G There is also a sub algorithm for the strings x being easy for length $q(|x|)$. Observe that a straightforward adaption of **F** from the proof of Theorem 4.2.5 (Hemaspaandra, Hemaspaandra, and Hempel 1996) does not work here, since the sets B_i constructed there would (even with the adaption to the current situation) remain $\Sigma_3^{\rm p}$ sets, but one needs $\Sigma_2^{\rm p}$ sets here.

Claim G: Let q be the polynomial defined in **E**. *There exist sets $B_1, B_2 \in \Sigma_2^{\rm p}$ such that,*

1. *$B_1 - B_2 \subseteq \overline{L_{\mathrm{DIFF}_2(\Sigma_2^{\rm p})}}$ and*
2. *for all $x \in \Sigma^*$, if x is an easy string for length $q(|x|)$ then*

$$x \notin L_{\mathrm{DIFF}_2(\Sigma_2^{\rm p})} \iff x \in B_1 - B_2.$$

Recall that $L_{\mathrm{DIFF}_2(\Sigma_2^{\rm p})} = L_1 - L_2$ where $L_1, L_2 \in \Sigma_2^{\rm p}$. Without loss of generality let $L_1 \supseteq L_2$ [CGH+88]. Define for $i = 1, 2$,

$$B_i = \{x \mid (\exists x_1 : |x_1| \leq q(|x|))[(x_1 \in L_{\mathrm{NP}} \iff h'(\langle x_1, x\rangle) \notin L_{\mathrm{P}}) \wedge h''(\langle x_1, x\rangle) \in L_i]\}.$$

Note that $B_1, B_2 \in \Sigma_2^{\rm p}$ and $B_1 \supseteq B_2$. We will prove the claim by showing that for all $x \in \Sigma^*$, $x \in B_1 - B_2$ if and only if x is easy for length $q(|x|)$ and $x \notin L_1 - L_2$.

Let $x \in \Sigma^*$. Observe that for all $i = 1, 2$, $x \in B_1 - B_2$ implies that x is easy for length $q(|x|)$. So it suffices to show that if x is easy for length $q(|x|)$ then $x \notin L_{\mathrm{DIFF}_2(\Sigma_2^{\rm p})} \iff x \in B_1 - B_2$.

Let x be easy for length $q(|x|)$. Let $t = \max(\{0\} \cup \{i \in \{1, 2\} \mid x \in B_i\})$. Let z_2 be a string such that

$$\begin{aligned}(\exists x_1 : |x_1| \leq q(|x|))\ [z_2 = h''(\langle x_1, x\rangle) \wedge \\ (x_1 \in L_{\mathrm{NP}} \iff h'(\langle x_1, x\rangle) \notin L_{\mathrm{P}}) \wedge \\ (t > 0 \implies z_2 \in L_t)].\end{aligned}$$

Such a string z_2 exists since x is easy for length $q(|x|)$. Note that $x \notin L_1 - L_2 \iff z_2 \in L_1 - L_2$. This follows from the definition of z_2 and the fact the equivalence

$$(x_1 \in L_{\mathrm{NP}} \iff x \notin L_{\mathrm{DIFF}_2(\Sigma_2^{\rm p})}) \iff \\ (h'(\langle x_1, x\rangle) \in L_{\mathrm{P}} \iff h''(\langle x_1, x\rangle) \notin L_{\mathrm{DIFF}_2(\Sigma_2^{\rm p})})$$

does hold for all $x_1 \in \Sigma^*$ according to **C**. Furthermore, $x \in B_1 - B_2$ if and only if $z_2 \in L_1 - L_2$ due to the definition of z_2, B_1, B_2, and t. Thus,

$$\begin{aligned} x \in \overline{L_{\mathrm{DIFF}_2(\Sigma_2^{\rm p})}} &\iff x \notin L_1 - L_2 \\ &\iff z_2 \in L_1 - L_2 \\ &\iff x \in B_1 - B_2. \end{aligned}$$

H Combining Claims F and G while exploiting the structure of $\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})$ shows

Claim H: $\overline{L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}} \in \mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})$.

Let the sets $A', B_1, B_2 \in \Sigma_2^{\mathrm{p}}$ be as defined in **F** and **G**. We show the above claim by proving $\overline{L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}} = (B_1 \cup A') - B_2$.

Suppose $x \in \overline{L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}}$. Note that x is exactly one of the following, either easy or hard for length $q(|x|)$. If x is easy for that length then $x \in B_1 - B_2$ according to Claim G. If x is hard for length $q(|x|)$ then $x \in A'$ according to Claim F and $x \notin B_2$ according to the definition of B_2 in **G**. In both cases we certainly have $x \in (B_1 \cup A') - B_2$.

Now suppose $x \in (B_1 \cup A') - B_2$. Hence $x \in B_1 \cup A'$. If $x \in A'$ then $x \in \overline{L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}}$ according to Claim F. If $x \notin A'$ then $x \in B_1 - B_2$. But this implies $x \in \overline{L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}}$ according to Claim G.

I Since $\overline{L_{\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}})}}$ is complete for $\mathrm{coDIFF}_2(\Sigma_2^{\mathrm{p}})$ we obtain $\mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}}) = \mathrm{coDIFF}_2(\Sigma_2^{\mathrm{p}})$.

End of Proof of Main Claim ▮

Theorem 4.2.9 together with Theorem 3.3.1 allows to conclude a collapse of the polynomial hierarchy.

Corollary 4.2.10 If $\mathrm{P}^{\Sigma_2^{\mathrm{p}}}_{2\text{-tt}} = \mathrm{P}^{\Sigma_2^{\mathrm{p}}}_{3\text{-tt}}$ then $\mathrm{PH} = \mathrm{DIFF}_2(\Sigma_2^{\mathrm{p}}) \boldsymbol{\Delta} \Sigma_3^{\mathrm{p}}$.

4.3 A New Result

The key result of [HHH97b] is based on the following theorem.

Theorem 4.3.1 [HHH97b] For all $m \geq 1$ and all $0 < i < k-1$, if $\Sigma_i^{\mathrm{p}} \boldsymbol{\Delta} \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$ is closed under complementation, then $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_k^{\mathrm{p}})$.

This, clearly, is a true downward collapse result. Removing the asymmetry in its hypothesis we extend this result to also hold for all levels of the boolean hierarchy over Σ_i^{p}. In particular, we show that for all $s, m \geq 1$ and all $0 < i < k-1$, if $\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}}) \boldsymbol{\Delta} \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$ is closed under complementation, then $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_k^{\mathrm{p}})$. Our proof is based on the approach of [HHH97b] combined with the observation that the results on the connection between a collapse of the boolean hierarchy and a collapse of the polynomial hierarchy (see Chapter 3) also apply to prefixes of Σ^*. Hence our proof employs the easy-hard technique in two directions. On one hand, we make use of the recent modifications of the easy-hard technique that allowed to prove the downward collapse results we have reviewed in the previous section [HHH96a, HHH99, BF98, HHH97b]. On the other hand, we exploit the easy-hard technique (in a more technical way) in its original version and extensions [Kad88, Wag87, Wag89, CK96, BCO93] that subsequently tightened the connection between the boolean hierarchy and the polynomial hierarchy. Our result has a number of applications to query order classes.

4.3.1 The Easy-Hard Technique in Double Use

In this sub section we prove the main result of this chapter, Theorem 4.3.3.

Our first lemma establishes that classes of the form $\mathcal{C}\boldsymbol{\Delta}\mathcal{D}$ in some cases contain well structured many-one complete sets.

Lemma 4.3.2 Let $\mathcal{C}$ and $\mathcal{D}$ be complexity classes. If C is $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete for $\mathcal{C}$ and D is $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete for $\mathcal{D}$, then $C\tilde{\Delta}D$ is $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete for $\mathcal{C}\boldsymbol{\Delta}\mathcal{D}$.

Proof: Let C and D be many-one complete for $\mathcal{C}$ and $\mathcal{D}$, respectively. First observe that $C\tilde{\Delta}D = \{\langle x,y\rangle \mid x \in C\}\Delta\{\langle x,y\rangle \mid y \in D\}$ and thus $C\tilde{\Delta}D \in \mathcal{C}\boldsymbol{\Delta}\mathcal{D}$. For the hardness claim, suppose that $A \in \mathcal{C}\boldsymbol{\Delta}\mathcal{D}$. Hence there exist sets $A_1 \in \mathcal{C}$ and $A_2 \in \mathcal{D}$ such that $A = A_1\Delta A_2$. Let f_1 and f_2 be polynomial-time computable functions many-one reducing A_1 to C and A_2 to D respectively. Thus, for all $x \in \Sigma^*$,

$$\begin{aligned} x \in A &\iff (x \in A_1 \iff x \notin A_2) \\ &\iff (f_1(x) \in C \iff f_2(x) \notin D) \\ &\iff \langle f_1(x), f_2(x)\rangle \in C\tilde{\Delta}D. \end{aligned}$$

This shows that A is many-one reducible to $C\tilde{\Delta}D$. ∎

We now state the main theorem of this chapter.

Theorem 4.3.3 Let $s,m \geq 1$ and $0 < i < k-1$. If $\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})\boldsymbol{\Delta}\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$ is closed under complementation, then $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_k^{\mathrm{p}})$.

In Figure 4.1 we give an overview over the previously known downward collapse results. The figure illustrates that our new result not just strengthens a claim from [HHH97b] but also a result from [HHH99].

Proof of Theorem 4.3.3: Since for $s = 1$ the claim to be proven is exactly the claim of Theorem 4.3.1 we henceforward assume $s \geq 2$.

Let $s \geq 2$, $m \geq 1$, and $0 < i < k-1$.

A Let $L_{\Sigma_i^{\mathrm{p}}}$, $L_{\Sigma_{i+1}^{\mathrm{p}}}$, and $L_{\Sigma_{i+2}^{\mathrm{p}}}$ be $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete languages for Σ_i^{p}, $\Sigma_{i+1}^{\mathrm{p}}$, and $\Sigma_{i+2}^{\mathrm{p}}$, respectively, that satisfy[1]

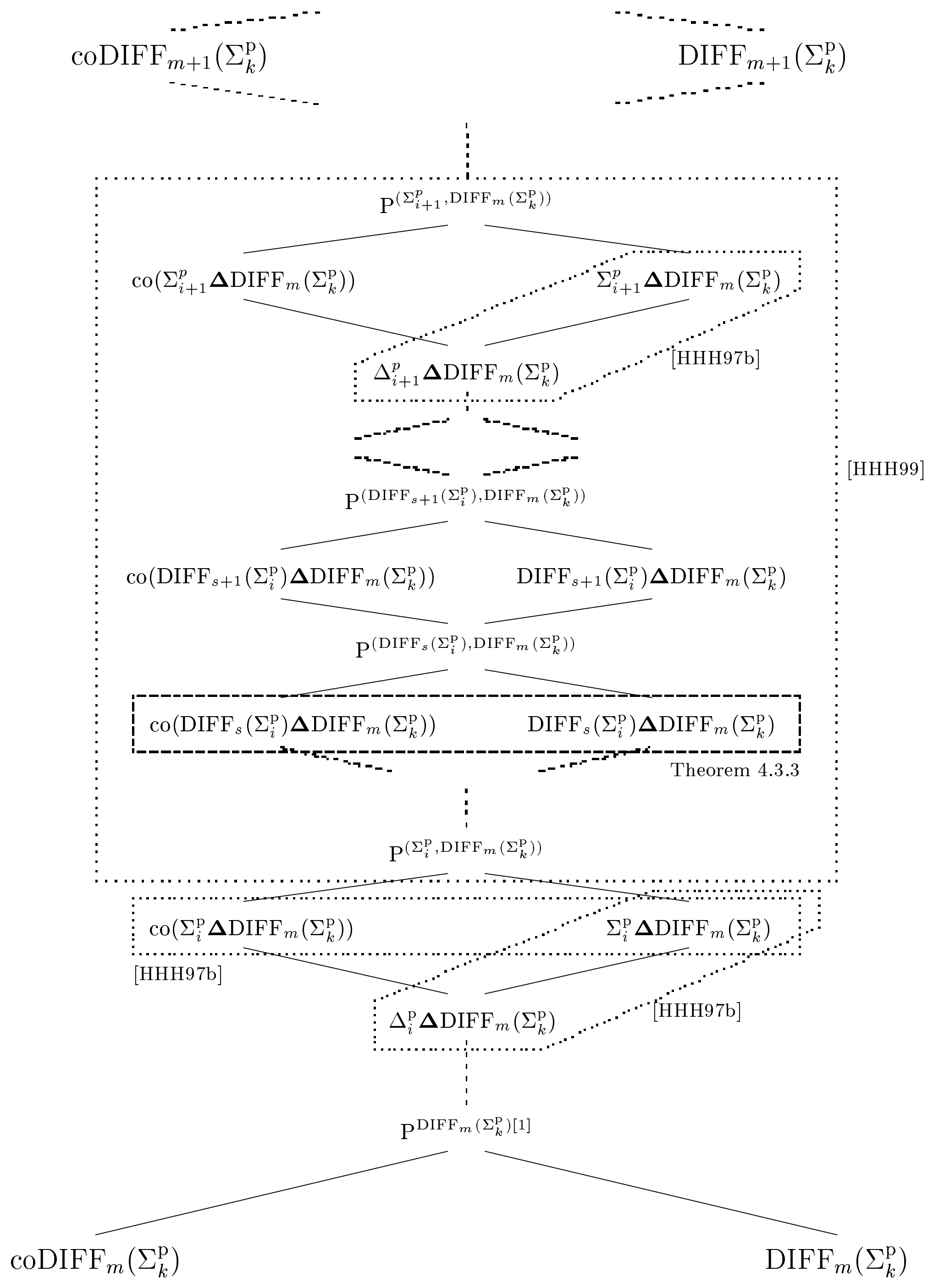
$$L_{\Sigma_{i+1}^{\mathrm{p}}} = \{x \mid (\exists y : |y| = |x|)[\langle x,y\rangle \notin L_{\Sigma_i^{\mathrm{p}}}]\},$$

and

$$L_{\Sigma_{i+2}^{\mathrm{p}}} = \{x \mid (\exists y : |y| = |x|)[\langle x,y\rangle \notin L_{\Sigma_{i+1}^{\mathrm{p}}}]\}.$$

Let $L_{\Sigma_k^{\mathrm{p}}}$ be a $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete language for Σ_k^{p} and let $L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}$ be $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete for $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$. Let $L_{\Pi_i^{\mathrm{p}}} = \overline{L_{\Sigma_i^{\mathrm{p}}}}$ and define $L_{\mathrm{DIFF}_1(\Pi_i^{\mathrm{p}})} = L_{\Pi_i^{\mathrm{p}}}$ and for $j \geq 2$, $L_{\mathrm{DIFF}_j(\Pi_i^{\mathrm{p}})} =$

[1]By Stockmeyer's [Sto77] standard quantifier characterization of the polynomial hierarchy's levels, such sets do exist.



$\mathrm{coDIFF}_{m+1}(\Sigma_k^p)$
$\mathrm{DIFF}_{m+1}(\Sigma_k^p)$
$\mathrm{P}^{(\Sigma_{i+1}^p,\mathrm{DIFF}_m(\Sigma_k^p))}$
$\mathrm{co}(\Sigma_{i+1}^p \mathbf{\Delta} \mathrm{DIFF}_m(\Sigma_k^p))$
$\Sigma_{i+1}^p \mathbf{\Delta} \mathrm{DIFF}_m(\Sigma_k^p)$
$\Delta_{i+1}^p \mathbf{\Delta} \mathrm{DIFF}_m(\Sigma_k^p)$
[HHH97b]
[HHH99]
$\mathrm{P}^{(\mathrm{DIFF}_{s+1}(\Sigma_i^p),\mathrm{DIFF}_m(\Sigma_k^p))}$
$\mathrm{co}(\mathrm{DIFF}_{s+1}(\Sigma_i^p) \mathbf{\Delta} \mathrm{DIFF}_m(\Sigma_k^p))$
$\mathrm{DIFF}_{s+1}(\Sigma_i^p) \mathbf{\Delta} \mathrm{DIFF}_m(\Sigma_k^p)$
$\mathrm{P}^{(\mathrm{DIFF}_s(\Sigma_i^p),\mathrm{DIFF}_m(\Sigma_k^p))}$
$\mathrm{co}(\mathrm{DIFF}_s(\Sigma_i^p) \mathbf{\Delta} \mathrm{DIFF}_m(\Sigma_k^p))$
$\mathrm{DIFF}_s(\Sigma_i^p) \mathbf{\Delta} \mathrm{DIFF}_m(\Sigma_k^p)$
Theorem 4.3.3
$\mathrm{P}^{(\Sigma_i^p,\mathrm{DIFF}_m(\Sigma_k^p))}$
$\mathrm{co}(\Sigma_i^p \mathbf{\Delta} \mathrm{DIFF}_m(\Sigma_k^p))$
$\Sigma_i^p \mathbf{\Delta} \mathrm{DIFF}_m(\Sigma_k^p)$
[HHH97b]
$\Delta_i^p \mathbf{\Delta} \mathrm{DIFF}_m(\Sigma_k^p)$
[HHH97b]
$\mathrm{P}^{\mathrm{DIFF}_m(\Sigma_k^p)[1]}$
$\mathrm{coDIFF}_m(\Sigma_k^p)$
$\mathrm{DIFF}_m(\Sigma_k^p)$

Figure 4.1: Inclusion structure and results overview—a collapse of all classes in a dotted box implies $\mathrm{DIFF}_m(\Sigma_k^p) = \mathrm{coDIFF}_m(\Sigma_k^p)$ $(0 < i < k-1)$.

$\{\langle x,y\rangle \mid x \in L_{\Pi_i^{\mathrm{p}}} \wedge y \notin L_{\mathrm{DIFF}_{j-1}(\Pi_i^{\mathrm{p}})}\}$. Note that $L_{\mathrm{DIFF}_j(\Pi_i^{\mathrm{p}})}$ is many-one complete for $\mathrm{DIFF}_j(\Pi_i^{\mathrm{p}})$ for all $j \geq 1$. Observe that $\mathrm{DIFF}_j(\Sigma_i^{\mathrm{p}}) = \mathrm{DIFF}_j(\Pi_i^{\mathrm{p}})$ if j is even and $\mathrm{DIFF}_j(\Sigma_i^{\mathrm{p}}) = \mathrm{coDIFF}_j(\Pi_i^{\mathrm{p}})$ if j is odd. Let $L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})} = L_{\mathrm{DIFF}_s(\Pi_i^{\mathrm{p}})}$ if s is even and $L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})} = \overline{L_{\mathrm{DIFF}_s(\Pi_i^{\mathrm{p}})}}$ if s is odd. Then $L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})}$ is $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete for $\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})$.

B Recall that $L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})}\tilde{\Delta}L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}$ is many-one complete for $\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})\boldsymbol{\Delta}\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$ by Lemma 4.3.2.

C Since by assumption $\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})\boldsymbol{\Delta}\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$ is closed under complementation, there exists a many-one reduction from $L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})}\tilde{\Delta}L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}$ to its complement. That is, there exists a polynomial-time computable function h such that for all $x_1, x_2 \in \Sigma^*$,

$$\langle x_1, x_2\rangle \in L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})}\tilde{\Delta}L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})} \Leftrightarrow h(\langle x_1,x_2\rangle) \notin L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})}\tilde{\Delta}L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}.$$

Let h' and h'' be two polynomial-time computable functions, such that for all $x_1, x_2 \in \Sigma^*$, $h(\langle x_1,x_2\rangle) = \langle h'(\langle x_1,x_2\rangle), h''(\langle x_1,x_2\rangle)\rangle$. Hence, for all $x_1, x_2 \in \Sigma^*$:

$$(x_1 \in L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})} \Leftrightarrow x_2 \notin L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}) \iff$$
$$(h'(\langle x_1,x_2\rangle) \in L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})} \Leftrightarrow h''(\langle x_1,x_2\rangle) \in L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}).$$

D We say that a string x is *easy for length* n if there exists a string x_1 such that $|x_1| \leq n$ and $(x_1 \in L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})} \Leftrightarrow h'(\langle x_1, x\rangle) \in L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})})$. Note that with a $\Sigma_{i+1}^{\mathrm{p}}$ algorithm one can test whether a string x is easy for length $r(|x|)$, where r is some polynomial.

We say that x is *hard for length* n if $|x| \leq n$ and x is not easy for length n, i.e., if $|x| \leq n$ and, for all x_1 with $|x_1| \leq n$, $(x_1 \in L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})} \Leftrightarrow h'(\langle x_1, x\rangle) \notin L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})})$.

Let for all $a \in \Sigma^*$, h_a denote the function such that for all $y \in \Sigma^*$, $h_a(y) = h'(\langle y, a\rangle)$. Observe that if x is a hard string for length n, then x induces a many-one reduction from $\left(L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})}\right)^{\leq n}$ to $\overline{L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})}}$, namely, h_x; for all x_1 such that $|x_1| \leq n$,

$$x_1 \in L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})} \iff h_x(x_1) \notin L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})}.$$

Note that $h_x(x_1)$ is computable in time polynomial in $\max\{|x|, |x_1|\}$. Since every string x, $|x| \leq n$, is exactly one of the following, either easy or hard for length n, we divide the problem of giving a $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$ algorithm for $\overline{L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}}$ into two sub problems; one being responsible for all strings x that are easy for length $q(|x|)$, and one for all strings x that are hard for length $q(|x|)$, where q is a polynomial we will specify exactly soon.

E We first give a $\mathrm{P}^{\Sigma_{k-1}^{\mathrm{p}}}$ algorithm for all strings x in $\overline{L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}}$ that are hard for length $q(|x|)$.

Claim E: There exist a set $A \in \mathrm{P}^{\Sigma^{\mathrm{p}}_{k-1}}$ and a polynomial q satisfying $(\forall \widehat{m} \geq 0)[q(\widehat{m}+1) > q(\widehat{m}) > 0]$ such that for all $x \in \Sigma^*$, if x is hard for length $q(|x|)$ then

$$x \notin L_{\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k)} \iff x \in A.$$

It is known that a collapse of the boolean hierarchy over Σ^{p}_i implies a collapse of the polynomial hierarchy (see Chapter 3 for references and results). The best known results conclude a collapse of the polynomial hierarchy to a level within the boolean hierarchy over $\Sigma^{\mathrm{p}}_{i+1}$. In other words, if $L_{\mathrm{DIFF}_s(\Sigma^{\mathrm{p}}_i)}$ many-one reduces to $\overline{L_{\mathrm{DIFF}_s(\Sigma^{\mathrm{p}}_i)}}$ then there is (at least) a $\mathrm{P}^{\Sigma^{\mathrm{p}}_{i+1}}$ algorithm for $L_{\Sigma^{\mathrm{p}}_{i+2}}$. Though a hard string for length n only induces a many-one reduction between initial segments of $L_{\mathrm{DIFF}_s(\Sigma^{\mathrm{p}}_i)}$ and $\overline{L_{\mathrm{DIFF}_s(\Sigma^{\mathrm{p}}_i)}}$, we would nevertheless like to derive a $\mathrm{P}^{\Sigma^{\mathrm{p}}_{i+1}}$ algorithm for some (to be specified soon) of $L_{\Sigma^{\mathrm{p}}_{i+2}}$. The following lemma does exactly that.

Lemma 4.3.4 Let $s \geq 2$, $m \geq 1$, and $0 < i < k-1$.
Suppose that $\mathrm{DIFF}_s(\Sigma^{\mathrm{p}}_i)\boldsymbol{\Delta}\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k)$ is closed under complementation. There exist a set $D \in \mathrm{P}^{\Sigma^{\mathrm{p}}_{i+1}}$ and a polynomial r such that for all n, (a) $r(n+1) > r(n) > 0$ and (b) for all $x \in \Sigma^*$, if x is a hard string for length $r(n)$ then for all $y \in (\Sigma^*)^{\leq n}$,

$$y \in L_{\Sigma^{\mathrm{p}}_{i+2}} \iff \langle x, 1^n, y\rangle \in D.$$

We defer the proof of Lemma 4.3.4 and first finish the proof of Claim E and the proof of our theorem.

If x is a hard string for length $q(|x|)$ we will use the result of Lemma 4.3.4 to obtain a $\mathrm{P}^{\Sigma^{\mathrm{p}}_{k-1}}$ algorithm for all strings x in $L_{\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k)}$ that are hard for length $q(|x|)$, and hence (since $\mathrm{P}^{\Sigma^{\mathrm{p}}_{k-1}}$ is closed under complementation) certainly a $\mathrm{P}^{\Sigma^{\mathrm{p}}_{k-1}}$ algorithm for all strings x in $\overline{L_{\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k)}}$ that are hard for length $q(|x|)$.

To be more precise, suppose that x is a hard string for length $r(n)$. According to the above Lemma 4.3.4, x induces a $\mathrm{P}^{\Sigma^{\mathrm{p}}_{i+1}}$ algorithm for all strings in $\left(L_{\Sigma^{\mathrm{p}}_{i+2}}\right)^{\leq n}$ that runs in time polynomial in n. What we would like to conclude is a $\mathrm{P}^{\Sigma^{\mathrm{p}}_{k-1}}$ algorithm for $\left(L_{\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k)}\right)^{=|x|}$. Recall that $L_{\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k)} = L_1 - (L_2 - (L_3 - \cdots (L_{m-1} - L_m)\cdots))$, where $L_j \in \Sigma^{\mathrm{p}}_k$ for all j. Since $L_{\Sigma^{\mathrm{p}}_k}$ is complete for Σ^{p}_k, there exist functions $f_1, \ldots, f_m$ many-one reducing $L_1, \ldots, L_m$ to $L_{\Sigma^{\mathrm{p}}_k}$, respectively. Let the output sizes of all the f_j's be bounded by the polynomial p', which without loss of generality satisfies $(\forall \widehat{m} \geq 0)[p'(\widehat{m}+1) > p'(\widehat{m}) > 0]$. Hence an x-induced $\mathrm{P}^{\Sigma^{\mathrm{p}}_{k-1}}$ algorithm for strings in $\left(L_{\Sigma^{\mathrm{p}}_k}\right)^{\leq p'(|x|)}$ suffices to give us a $\mathrm{P}^{\Sigma^{\mathrm{p}}_{k-1}}$ algorithm for strings in $\left(L_{\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k)}\right)^{=|x|}$.

But Lemma 4.3.4 gives us exactly this, if $k = i+2$ and if x is hard for length $r(p'(|x|))$. More formally, note that if x is a hard string for length $r(p'(|x|))$ we have that

$$\begin{aligned} x \in L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})} &\iff x \in L_1 - (L_2 - (\cdots - (L_{m-1} - L_m)\cdots)) \\ &\iff f_1(x) \in L_{\Sigma_k^{\mathrm{p}}} \wedge \neg(f_2(x) \in L_{\Sigma_k^{\mathrm{p}}} \wedge \neg(\cdots \wedge \neg(f_m(x) \in D')\cdots)) \\ &\iff \langle x, 1^{p'(|x|)}, f_1(x)\rangle \in D' \wedge \neg(\cdots \wedge \neg(\langle x, 1^{p'(|x|)}, f_m(x)\rangle \in D')\cdots). \end{aligned}$$

Set $A' = \{x \mid \langle x, 1^{p'(|x|)}, f_1(x)\rangle \in D' \wedge \neg(\cdots \wedge \neg(\langle x, 1^{p'(|x|)}, f_m(x)\rangle \in D')\cdots)\}$ and note that clearly $A' \in \mathrm{P}^{\Sigma_{k-1}^{\mathrm{p}}}$.

Consequently, if $k = i+2$ set $A = \overline{A'}$ and let q be a polynomial satisfying $(\forall \widehat{m} \geq 0)[q(\widehat{m}+1) > q(\widehat{m}) > 0]$ and for all n, $q(n) \geq r(p'(n))$. Note that this proves Claim E for $k = i+2$.

For the case $k > i+2$, let M be a $\Sigma_{k-(i+2)}^{\mathrm{p}}$ machine recognizing $L_{\Sigma_k^{\mathrm{p}}}$ with oracle queries to $L_{\Sigma_{i+2}^{\mathrm{p}}}$ and running in time q' for some polynomial q' satisfying $(\forall \widehat{m} \geq 0)[q'(\widehat{m}+1) > q'(\widehat{m}) > 0]$. We can certainly replace the $L_{\Sigma_{i+2}^{\mathrm{p}}}$ queries by queries to a $\mathrm{P}^{\Sigma_{i+1}^{\mathrm{p}}}$ oracle and thus obtain a $\Sigma_{k-1}^{\mathrm{p}}$ algorithm (running in time polynomial in $|x|$) for $\left(L_{\Sigma_k^{\mathrm{p}}}\right)^{\leq p'(|x|)}$, if we ensure that Lemma 4.3.4 gives us an x-induced $\mathrm{P}^{\Sigma_{i+1}^{\mathrm{p}}}$ algorithm for all strings in $\left(L_{\Sigma_{i+2}^{\mathrm{p}}}\right)^{\leq q'(p'(|x|))}$. Thus, if $k > i+2$ we need x to be hard for length $r(q'(p'(|x|)))$.

More formally, if $k > i+2$ define D' to be the set accepted by the following algorithm: On input $\langle x, y\rangle$ simulate $M^{L_{\Sigma_{i+2}^{\mathrm{p}}}}(y)$ but replace every query z made by $M^{L_{\Sigma_{i+2}^{\mathrm{p}}}}(y)$ to $L_{\Sigma_{i+2}^{\mathrm{p}}}$ by a query $\langle x, 1^{q'(|y|)}, z\rangle$ to D. Since M is a $\Sigma_{k-(i+2)}^{\mathrm{p}}$ machine and $D \in \mathrm{P}^{\Sigma_{i+1}^{\mathrm{p}}}$ we conclude that $D' \in \Sigma_{k-1}^{\mathrm{p}}$. Recall from Lemma 4.3.4 that for all n, if x is a hard string for length $r(n)$ then for all z, $|z| \leq n$,

$$z \in L_{\Sigma_{i+2}^{\mathrm{p}}} \iff \langle x, 1^n, z\rangle \in D.$$

Since for all n, $M^{L_{\Sigma_{i+2}^{\mathrm{p}}}}(y)$, $|y| \leq n$, can only generate queries z of length at most $q'(n)$ we have that if x is a hard string for length $r(q'(n))$ then for all y, $|y| \leq n$,

$$y \in L_{\Sigma_k^{\mathrm{p}}} \iff \langle x, y\rangle \in D'.$$

Hence, for all n, if x is hard for length $r(q'(p'(n)))$ then for all $1 \leq i \leq m$ and all y, $|y| \leq n$,

$$f_i(y) \in L_{\Sigma_k^{\mathrm{p}}} \iff \langle x, f_i(y)\rangle \in D'.$$

Define $A' = \{x \mid \langle x, f_1(x)\rangle \in D' \wedge \neg(\langle x, f_2(x)\rangle \in D' \wedge \neg(\cdots \wedge \neg(\langle x, f_m(x)\rangle \in D')\cdots))\}$ and note that clearly $A' \in \mathrm{P}^{\Sigma_{k-1}^{\mathrm{p}}}$. It follows that if x is a hard string for length

$r(q'(p'(|x|)))$ then

$$\begin{aligned} x \in L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})} &\iff x \in L_1 - (L_2 - (\cdots - (L_{m-1} - L_m)\cdots)) \\ &\iff f_1(x) \in L_{\Sigma_k^{\mathrm{p}}} \wedge \neg(f_2(x) \in L_{\Sigma_k^{\mathrm{p}}} \wedge \neg(\cdots \wedge \neg(f_m(x) \in D')\cdots)) \\ &\iff \langle x, f_1(x)\rangle \in D' \wedge \neg(\cdots \wedge \neg(\langle x, f_m(x)\rangle \in D')\cdots) \\ &\iff x \in A'. \end{aligned}$$

So, for $k > i+2$ set $A = \overline{A'}$ and let q be a polynomial satisfying $(\forall \widehat{m} \geq 0)[q(\widehat{m}+1) > q(\widehat{m}) > 0]$ and for all n, $q(n) \geq r(q'(p'(n)))$. It is not hard to verify that this proves Claim E if $k > i+2$.

F We now give a $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$ algorithm for all strings x in $\overline{L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}}$ that are easy for length $q(|x|)$.

Claim F: Let q be the polynomial defined in **E**. *There exist sets $B_1, B_2, \ldots, B_m \in \Sigma_k^{\mathrm{p}}$ such that for all $x \in \Sigma^*$, if x is easy for length $q(|x|)$ then*

$$x \notin L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})} \iff x \in B_1 - (B_2 - (\ldots(B_{m-1} - B_m)\ldots)).$$

Let q be the polynomial defined in **E**. Clearly, we have the following algorithm to test whether $x \in \overline{L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}}$ in the case that (our input) x is an easy string for length $q(|x|)$. On input x, guess x_1 with $|x_1| \leq q(|x|)$ and accept if and only if $(x_1 \in L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})} \Leftrightarrow h'(\langle x_1, x\rangle) \in L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})})$ and $h''(\langle x_1, x\rangle) \in L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}$. This algorithm is not necessarily a $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$ algorithm, but it does inspire the following $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$ algorithm to test whether $x \in \overline{L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}}$ in the case that x is an easy string for length $q(|x|)$.

Let $L_1, L_2, \cdots, L_m$ be languages in Σ_k^{p} such that $L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})} = L_1 - (L_2 - (L_3 - \cdots(L_{m-1} - L_m)\cdots))$ and $L_1 \supseteq L_2 \supseteq \cdots \supseteq L_{m-1} \supseteq L_m$ (this can be done, as it is simply the "telescoping" normal form of the levels of the boolean hierarchy over Σ_k^{p}, see [CGH+88, Hau14]). For $1 \leq r \leq m$, define B_r as the language accepted by the following Σ_k^{p} machine: On input x, guess x_1 with $|x_1| \leq q(|x|)$ and accept if and only if $(x_1 \in L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})} \Leftrightarrow h'(\langle x_1, x\rangle) \in L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})})$ and $h''(\langle x_1, x\rangle) \in L_r$.

Note that $B_r \in \Sigma_k^{\mathrm{p}}$ for each r, and that $B_1 \supseteq B_2 \supseteq \cdots \supseteq B_{m-1} \supseteq B_m$. We will show that if x is an easy string for length $q(|x|)$, then $x \in \overline{L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}}$ if and only if $x \in B_1 - (B_2 - (B_3 - \cdots(B_{m-1} - B_m)\cdots))$.

So suppose that x is an easy string for $q(|x|)$. Define r' to be the unique integer such that (a) $0 \leq r' \leq m$, (b) $x \in B_{s'}$ for $1 \leq s' \leq r'$, and (c) $x \notin B_{s'}$ for $s' > r'$. It is immediate that $x \in B_1 - (B_2 - (B_3 - \cdots(B_{m-1} - B_m)\cdots))$ if and only if r' is odd.

Let w be some string such that:

- $(\exists x_1 : |x_1| \leq q(|x|))$
  $[h''(\langle x_1, x\rangle) = w \wedge (x_1 \in L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})} \Leftrightarrow h'(\langle x_1, x\rangle) \in L_{\mathrm{DIFF}_s(\Sigma_i^{\mathrm{p}})})]$, and

- $w \in L_{r'}$ if $r' > 0$.

Note that such a w exists, since x is easy for length $q(|x|)$. By the definition of r' (namely, since $x \notin B_{s'}$ for $s' > r'$), $w \notin L_{s'}$ for all $s' > r'$. It follows that $w \in L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}$ if and only if r' is odd.

It is clear, keeping in mind the definition of h (and h', h''), that

$$\begin{aligned} x \in \overline{L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}} &\iff w \in L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})} \\ &\iff r' \text{ is odd} \\ &\iff x \in B_1 - (B_2 - (B_3 - \cdots (B_{m-1} - B_m) \cdots)). \end{aligned}$$

This completes the proof of Claim F.

G Combining the results of Claims E and F we obtain

Claim G: $\overline{L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}} \in \mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$.

We prove Claim G by showing that $\overline{L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}} = \widehat{L_1} - (\widehat{L_2} - (\widehat{L_3} - (\cdots (\widehat{L_{m-1}} - \widehat{L_m}) \ldots)))$ for some sets $\widehat{L_1}, \widehat{L_2}, \ldots, \widehat{L_m} \in \Sigma_k^{\mathrm{p}}$.

Let the sets $A \in \mathrm{P}^{\Sigma_{k-1}^{\mathrm{p}}}$, $B_1, B_2, \ldots, B_m \in \Sigma_k^{\mathrm{p}}$ and the polynomial q be as defined in **E** and **F**. For $1 \leq r \leq m$, let $\widehat{L_r}$ be the set accepted by the following Σ_k^{p} algorithm:

1. On input x determine whether the input x is an easy string for length $q(|x|)$. This can be done with a one $\Sigma_{k-1}^{\mathrm{p}}$ oracle query, as checking whether the input is an easy string for length $q(|x|)$ can be done by one query to $\Sigma_{i+1}^{\mathrm{p}}$, and $i + 1 \leq k - 1$ by our $i < k - 1$ hypothesis.
2. If the previous step determined that the input is not an easy string, then the input must be a hard string for length $q(|x|)$. If $r = 1$ then accept if and only if $x \in A$. If $r > 1$ then reject.
3. If the first step determined that the input x is easy for length $q(|x|)$, then accept if and only if $x \in B_r$.

Note that the $\Sigma_{k-1}^{\mathrm{p}}$ oracle (implicitly described) in the above algorithm is being used for a number of different sets. However, as $\Sigma_{k-1}^{\mathrm{p}}$ is closed under disjoint union, this presents no problem as we can use the disjoint union of the sets, while modifying the queries so they address the appropriate part of the disjoint union.

It follows that, for all $x \in \Sigma^*$,

$$x \in \overline{L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}} \iff x \in \widehat{L_1} - (\widehat{L_2} - (\widehat{L_3} - \cdots (\widehat{L_{m-1}} - \widehat{L_m}) \cdots)).$$

H Since $\overline{L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}}$ is complete for $\mathrm{coDIFF}_m(\Sigma_k^{\mathrm{p}})$, we conclude that $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$ is closed under complementation, $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_k^{\mathrm{p}})$. ∎

We now give the proof of Lemma 4.3.4. The upcoming proof should be seen in the context with the proof of Theorem 4.3.3 as some notations we are going to use are defined there.

Proof of Lemma 4.3.4: Our proof closely follows the proof of Theorem 3.3.1 in order to emphasize the technical differences that make the current proof valid. Let t be a polynomial such that $|\langle x_1, x_2, \ldots, x_j\rangle| \leq t(\max\{|x_1|, |x_2|, \ldots, |x_j|\})$ for all $1 \leq j \leq 2s+2$ and all $x_1, x_2, \ldots, x_j \in \Sigma^*$. Without loss of generality let t be such that $t(n+1) > t(n) > 0$ for all n. Define $t^{(0)}(n) = n$ and $t^{(j)}(n) = \underbrace{t(t(\cdots t}_{j\ times}(n)\cdots))$ for all n and all $j \geq 1$.

Recall the definitions from **A** of the proof of Theorem 4.3.3 regarding $L_{\Sigma_i^{\rm p}}$, $L_{\Sigma_{i+1}^{\rm p}}$, $L_{\Sigma_{i+2}^{\rm p}}$, $L_{{\rm DIFF}_s(\Pi_i^{\rm p})}$, and $L_{{\rm DIFF}_s(\Sigma_i^{\rm p})}$.

Define r to be a polynomial such that $r(n+1) > r(n) > 0$ and $r(n) \geq t^{(s-1)}(n)$ for all n. Let n be an integer. Suppose that x is a hard string for length $r(n)$ in the sense as defined in **D** of the proof of Theorem 4.3.3. Then, for all y such that $|y| \leq r(n)$,

$$y \in L_{{\rm DIFF}_s(\Sigma_i^{\rm p})} \iff h_x(y) \notin L_{{\rm DIFF}_s(\Sigma_i^{\rm p})},$$

or equivalently

$$y \in L_{{\rm DIFF}_s(\Pi_i^{\rm p})} \iff h_x(y) \notin L_{{\rm DIFF}_s(\Pi_i^{\rm p})}.$$

Recall that $h_x(y)$ can be computed in time polynomial in $\max\{|x|, |y|\}$. Let for all $a \in \Sigma^*$, h'_a and h''_a be the functions such that for all $y \in \Sigma^*$, $h_a(y) = \langle h'_a(y), h''_a(y)\rangle$. Clearly, $h'_x(y)$ and $h''_x(y)$ can be computed in time polynomial in $\max\{|x|, |y|\}$. Thus, for all $y_1, y_2 \in \Sigma^*$ such that $|y_1| \leq n$ and $|y_2| \leq t^{(s-2)}(n)$,

$$(*) \quad y_1 \in L_{\Pi_i^{\rm p}} \wedge y_2 \notin L_{{\rm DIFF}_{s-1}(\Pi_i^{\rm p})} \iff h'_x(\langle y_1, y_2\rangle) \notin L_{\Pi_i^{\rm p}} \vee h''_x(\langle y_1, y_2\rangle) \in L_{{\rm DIFF}_{s-1}(\Pi_i^{\rm p})}.$$

We say that y_1 is s-easy for length n if and only if $|y_1| \leq n$ and $(\exists y_2 : |y_2| \leq t^{(s-2)}(n))[h'_x(\langle y_1, y_2\rangle) \notin L_{\Pi_i^{\rm p}}]$. y_1 is said to be s-hard for length n if and only if $|y_1| \leq n$, $y_1 \in L_{\Pi_i^{\rm p}}$, and $(\forall y_2 : |y_2| \leq t^{(s-2)}(n))[h'_x(\langle y_1, y_2\rangle) \in L_{\Pi_i^{\rm p}}]$. Observe that the above notions are defined with respect to our hard string x, since $h'_x(\langle y_1, y_2\rangle$ depends on x, y_1, and y_2. Furthermore, according to (*), if y_1 is s-easy for length n then $y_1 \in L_{\Pi_i^{\rm p}}$.

Suppose there exists an s-hard string ω_s for length n. Let for all $a, b \in \Sigma^*$, $h_{(a,b)}$ be the function such that for all $z \in \Sigma^*$, $h_{(a,b)}(z) = h''_a(\langle b, z\rangle)$. Note that $h_{(x,\omega_s)}(y)$ can be computed in time polynomial in $\max\{|x|, |\omega_s|, |y|\}$. In analogy to the above we define $s-1$-easy and $s-1$-hard strings. If an $s-1$-hard string exists we can repeat the process and define $s-2$-easy and $s-2$-hard strings and so on. Note that the definition of j-easy and j-hard strings can only be made with respect to our hard string x, some fixed s-hard string ω_s, some fixed $s-1$-hard string ω_{s-1}, ... , some fixed $j+1$-hard string ω_{j+1}. If we have found a sequence of strings $(\omega_s, \omega_{s-1}, \ldots, \omega_2)$ such that every ω_j is j-hard with respect to $(x, \omega_s, \omega_{s-1}, \ldots, \omega_{j+1})$ then we have for all y, $|y| \leq n$,

$$y \in L_{\Pi_i^{\rm p}} \iff h_{(x,\omega_s,\omega_{s-1},\ldots,\omega_2)}(y) \notin L_{\Pi_i^{\rm p}}.$$

We say that a string y is 1-easy for length n if and only if $|y| \leq n$ and $h_{(x,\omega_s,\omega_{s-1},\ldots,\omega_2)}(y) \notin L_{\Pi_i^{\mathrm{p}}}$. We define that no string is 1-hard for length n.

(x) is called a hard sequence for length n. A sequence $(x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell)$ of strings is called a hard sequence for length n if and only if ω_s is s-hard with respect to x and for all j, $\ell \leq j \leq s-1$, ω_j is j-hard with respect to $(x, \omega_s, \omega_{s-1}, \ldots, \omega_{j+1})$. Note that given a hard sequence $(x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell)$, the strings in $(L_{\Pi_i^{\mathrm{p}}})^{\leq n}$ divide into $\ell - 1$-easy and $\ell - 1$-hard strings (with respect to $(x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell)$) for length n.

(x) is called a maximal hard sequence if and only if there exists no s-hard string for length n. A hard sequence $(x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell)$ is called a maximal hard sequence for length n if and only if there exists no $\ell - 1$-hard string for length n with respect to $(x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell)$. If we in the following denote a maximal hard sequence by $(x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell)$ we explicitly include the case that the maximal hard sequence might be (x) or (x, ω_s).

Claim 1: *There exists a set $A \in \Sigma_i^{\mathrm{p}}$ such that if $(x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell)$ is a maximal hard sequence for length n then for all y and n satisfying $|y| \leq n$ it holds that:*

$$y \in L_{\Pi_i^{\mathrm{p}}} \iff \langle x, 1^n, \omega_s, \omega_{s-1}, \ldots, \omega_\ell, y \rangle \in A.$$

Proof of Claim 1: Let $(x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell)$ be a maximal hard sequence for length n. Note that $\ell \geq 2$ and that the strings in $(L_{\Pi_i^{\mathrm{p}}})^{\leq n}$ are exactly the strings of length at most n that are $(\ell-1)$-easy with respect to $(x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell)$. But it is immediate from the definition that testing whether a string y is $(\ell-1)$-easy for length n with respect to $(x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell)$ can be done by a Σ_i^{p} algorithm running in time polynomial in n: If $\ell \geq 3$, check $|y| \leq n$, guess y_2, $|y_2| \leq t^{(\ell-3)}(n)$, compute $h'_{(x,\omega_s,\omega_{s-1},\ldots,\omega_\ell)}$ and accept if and only if $h'_{(x,\omega_s,\omega_{s-1},\ldots,\omega_\ell)} \notin L_{\Pi_i^{\mathrm{p}}}$; If $\ell = 2$, check $|y| \leq n$, and accept if and only if $h_{(x,\omega_s,\omega_{s-1},\ldots,\omega_2)}(y) \notin L_{\Pi_i^{\mathrm{p}}}$.

Claim 2: *There exist a set $B \in \Sigma_i^{\mathrm{p}}$ and a polynomial $\widehat{p}$ such that $(\forall n \geq 0)[\widehat{p}(n+1) > \widehat{p}(n) > 0]$ and if $(x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell)$ is a maximal hard sequence for length $\widehat{p}(n)$ then for all y and n satisfying $|y| \leq n$ it holds that:*

$$y \in L_{\Sigma_{i+1}^{\mathrm{p}}} \iff \langle x, 1^n, \omega_s, \omega_{s-1}, \ldots, \omega_\ell, y \rangle \in B.$$

Proof of Claim 2: Let $A \in \Sigma_i^{\mathrm{p}}$ as in Claim 1. Let y be a string such that $|y| \leq n$. According to the definition of $L_{\Sigma_{i+1}^{\mathrm{p}}}$,

$$y \in L_{\Sigma_{i+1}^{\mathrm{p}}} \iff (\exists z : |z| = |y|)[\langle y, z \rangle \notin L_{\Sigma_i^{\mathrm{p}}}].$$

Recall that $L_{\Pi_i^{\mathrm{p}}} = \overline{L_{\Sigma_i^{\mathrm{p}}}}$. Define $\widehat{p}$ to be a polynomial such that $\widehat{p}(n+1) > \widehat{p}(n) > 0$ and $\widehat{p}(n) \geq t(n)$ for all n. In light of Claim 1, we obtain that if $(x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell)$ is a maximal hard sequence for length $\widehat{p}(n)$ then

$$y \in L_{\Sigma_{i+1}^{\mathrm{p}}} \iff (\exists z : |z| = |y|)[\langle x, 1^{\widehat{p}(n)}, \omega_s, \omega_{s-1}, \ldots, \omega_\ell, \langle y, z \rangle \rangle \in A].$$

We define $B = \{\langle x, 1^n, \omega_s, \omega_{s-1}, \ldots, \omega_j, y \rangle \mid (\exists z : |z| = |y|)[\langle x, 1^{\widehat{p}(n)}, \omega_s, \omega_{s-1}, \ldots, \omega_j, \langle y, z \rangle \rangle \in A]\}$. Clearly $B \in \Sigma_i^{\mathrm{p}}$. This proves the claim.

Claim 3: *There exist a set $C \in \Sigma_{i+1}^{\mathrm{p}}$ and a polynomial $\widehat{p}_1$ such that $(\forall n \geq 0)[\widehat{p}_1(n+1) > \widehat{p}_1(n) > 0$ and if $(x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell)$ is a maximal hard sequence for length $\widehat{p}_1(n)$, then for all y and n satisfying $|y| \leq n$ it holds that:*

$$y \in L_{\Sigma_{i+2}^{\mathrm{p}}} \iff \langle x, 1^n, \omega_s, \omega_{s-1}, \ldots, \omega_\ell, y\rangle \in C.$$

Proof of Claim 3: Let $B \in \Sigma_i^{\mathrm{p}}$ and $\widehat{p}$ be a polynomial, both as defined in Claim 2. Let y be a string such that $|y| \leq n$. According to the definition of $L_{\Sigma_{i+2}^{\mathrm{p}}}$,

$$y \in L_{\Sigma_{i+2}^{\mathrm{p}}} \iff (\exists z : |z| = |y|)[\langle y, z\rangle \notin L_{\Sigma_{i+1}^{\mathrm{p}}}].$$

Define $\widehat{p}_1$ to be a polynomial such that $\widehat{p}_1(n+1) > \widehat{p}_1(n) > 0$ and $\widehat{p}_1(n) \geq \widehat{p}(t(n))$ for all n. In light of Claim 2, we obtain that if $(x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell)$ is a maximal hard sequence for length $\widehat{p}_1(n)$ then

$$y \in L_{\Sigma_{i+2}^{\mathrm{p}}} \iff (\exists z : |z| = |y|)[\langle x, 1^{\widehat{p}_1(n)}, \omega_s, \omega_{s-1}, \ldots, \omega_\ell, \langle y, z\rangle\rangle \notin B].$$

Set $C = \{\langle x, 1^n, \omega_s, \omega_{s-1}, \ldots, \omega_j, y\rangle \mid (\exists z : |z| = |y|)[\langle x, 1^{\widehat{p}_1(n)}, \omega_s, \omega_{s-1}, \ldots, \omega_j, \langle y, z\rangle\rangle \notin B]\}$. Clearly $C \in \Sigma_{i+1}^{\mathrm{p}}$.

Claim 4: *There exists a set $D \in \mathrm{P}^{\Sigma_{i+1}^{\mathrm{p}}}$ such that for all y and n satisfying $|y| \leq n$ it holds that:*

$$y \in L_{\Sigma_{i+2}^{\mathrm{p}}} \iff \langle x, 1^n, y\rangle \in D.$$

Proof of Claim 4: Let $C \in \Sigma_{i+1}^{\mathrm{p}}$ and $\widehat{p}_1$ be a polynomial, both as defined in Claim 3. Note that

$$\{\langle x, 1^n, \omega_s, \omega_{s-1}, \ldots, \omega_\ell\rangle \mid (x, \omega_s, \omega_{s-1}, \ldots, \omega_\ell) \text{ is a hard sequence for length } \widehat{p}_1(n)\}$$

is a Π_i^{p} set. Consequently, the set of strings $\langle x, 1^n, k\rangle$ such that there exists a hard sequence for length $\widehat{p}_1(n)$ of length k is a $\Sigma_{i+1}^{\mathrm{p}}$ set.

The following $\mathrm{P}^{\Sigma_{i+1}^{\mathrm{p}}}$ algorithm accepts $\langle x, 1^n, y\rangle$ if and only if $y \in L_{\Sigma_{i+2}^{\mathrm{p}}}$: On input $\langle x, 1^n, y\rangle$ compute the largest k such that there exists a hard sequence for length $\widehat{p}_1(n)$ of length k. Then guess strings $\omega_s, \omega_{s-1}, \ldots, \omega_{s-k+2}$ of length at most $\widehat{p}_1(n)$. Verify that $(x, \omega_s, \omega_{s-1}, \ldots, \omega_{s-k+2})$ is a hard sequence for length $\widehat{p}_1(n)$ and accept if and only if $\langle x, 1^n, \omega_s, \omega_{s-1}, \ldots, \omega_{s-k+2}, y\rangle \in C$. ▮

4.3.2 Applications and Concluding Remarks

It is evident that Theorem 4.3.3 misses one interesting case, namely, the case $i = k-1$. Note that the $i = k$ case has been considered in detail in Chapter 3. Why is our proof unable to also establish the $i = k-1$ case? First of all, we crucially rely on a preliminary easy-hard test for every input in the final $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})$ algorithm for $\overline{L_{\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}})}}$. As has been noted already in Section 4.2 this test is responsible for the $i < k-1$ constraint. But besides that, there is a second place where we can not overcome $i < k-1$. Recall that Lemma 4.3.4, very informally put, shows that if x is a hard string for sufficiently large length there is a $\mathrm{P}^{\Sigma^{\mathrm{p}}_{i+1}}$ algorithm for some strings in $L_{\Sigma^{\mathrm{p}}_{i+2}}$. Observe that the proof of Lemma 4.3.4 does not require $i < k-1$. Though with a bit more effort one can even in the $i = k-1$ case show that there exists a $\mathrm{DIFF}_s(\Sigma^{\mathrm{p}}_i)\mathbf{\Delta}\mathrm{DIFF}_{s-1}(\Sigma^{\mathrm{p}}_{i+1})$ algorithm for some of $L_{\Sigma^{\mathrm{p}}_{i+2}}$ it is not at all clear how to combine the resulting $\mathrm{DIFF}_s(\Sigma^{\mathrm{p}}_i)\mathbf{\Delta}\mathrm{DIFF}_{s-1}(\Sigma^{\mathrm{p}}_k)$ algorithm for $\overline{L_{\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k)}}$ with the $\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k)$ algorithm for $\overline{L_{\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k)}}$ emerging from the "x is easy" case, if no preliminary easy-hard test is made.

However, proving the $i = k-1$ analogue of Theorem 4.3.3 would instantaneously improve Theorem 3.3.1. Suppose the statement of Theorem 4.3.3 could be extended to remain valid also for $0 < i = k-1$. Furthermore, assume that $\mathrm{BH}_m = \mathrm{coBH}_m$, $m \geq 2$. According to Theorem 3.3.1 we conclude $\mathrm{BH}_m\mathbf{\Delta}\mathrm{DIFF}_{m-1}(\Sigma^{\mathrm{p}}_2) = \mathrm{co}(\mathrm{BH}_m\mathbf{\Delta}\mathrm{DIFF}_{m-1}(\Sigma^{\mathrm{p}}_2)) = \mathrm{PH}$. By the above mentioned hypothetic strengthened downward collapse claim this would immediately imply $\mathrm{DIFF}_{m-1}(\Sigma^{\mathrm{p}}_2) = \mathrm{coDIFF}_{m-1}(\Sigma^{\mathrm{p}}_2) = \mathrm{PH}$, a truly surprising result.

Theorem 4.3.3 has a number of interesting applications. Query order classes of the form $\mathrm{P}^{(\Sigma^{\mathrm{p}}_i,\Sigma^{\mathrm{p}}_k)}_{s,m\text{-tt}}$, $i < k$, have been studied in [BC] and [Wag98]. Beigel and Chang [BC] mention that $\mathrm{P}^{(\Sigma^{\mathrm{p}}_i,\Sigma^{\mathrm{p}}_k)}_{s,m\text{-tt}} = \mathrm{P}^{(\Sigma^{\mathrm{p}}_i,\Sigma^{\mathrm{p}}_k)}_{s+1,m\text{-tt}}$ implies that the polynomial hierarchy collapses.

In light of Theorem 4.3.3 and Corollary 3.3.2, we can make this claim for $i < k-1$ more precise.

Theorem 4.3.5 For all $s, m \geq 1$ and all $0 < i < k-1$, if $\mathrm{P}^{(\Sigma^{\mathrm{p}}_i,\Sigma^{\mathrm{p}}_k)}_{s,m\text{-tt}} = \mathrm{P}^{(\Sigma^{\mathrm{p}}_i,\Sigma^{\mathrm{p}}_k)}_{s+1,m\text{-tt}}$ then $\mathrm{PH} = \mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k)\mathbf{\Delta}\mathrm{DIFF}_{m-1}(\Sigma^{\mathrm{p}}_{k+1})$.

Proof: It is not hard to verify that for all $s, m \geq 1$ and all $i, k > 0$,

$$\mathrm{P}^{(\Sigma^{\mathrm{p}}_i,\Sigma^{\mathrm{p}}_k)}_{s,m\text{-tt}} \subseteq \mathrm{DIFF}_{s+1}(\Sigma^{\mathrm{p}}_i)\mathbf{\Delta}\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k) \subseteq \mathrm{P}^{(\Sigma^{\mathrm{p}}_i,\Sigma^{\mathrm{p}}_k)}_{s+1,m\text{-tt}}.$$

In fact, one can via standard mind-change manipulations show (see also [Wag98]) that

$$\mathrm{P}^{(\Sigma^{\mathrm{p}}_i,\Sigma^{\mathrm{p}}_k)}_{s,m\text{-tt}} = \mathrm{P}^{(\mathrm{DIFF}_s(\Sigma^{\mathrm{p}}_i),\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k))}.$$

Thus,

$$\mathrm{P}^{(\Sigma^{\mathrm{p}}_i,\Sigma^{\mathrm{p}}_k)}_{s,m\text{-tt}} = \mathrm{P}^{(\Sigma^{\mathrm{p}}_i,\Sigma^{\mathrm{p}}_k)}_{s+1,m\text{-tt}} \implies \mathrm{DIFF}_{s+1}(\Sigma^{\mathrm{p}}_i)\mathbf{\Delta}\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k) = \mathrm{co}(\mathrm{DIFF}_{s+1}(\Sigma^{\mathrm{p}}_i)\mathbf{\Delta}\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k)).$$

For $0 < i < k-1$ we conclude $\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k) = \mathrm{coDIFF}_m(\Sigma^{\mathrm{p}}_k)$ from our Theorem 4.3.3. In light of Corollary 3.3.2, this collapse in the boolean hierarchy over Σ^{p}_k implies a collapse of the polynomial hierarchy to $\mathrm{DIFF}_m(\Sigma^{\mathrm{p}}_k)\mathbf{\Delta}\mathrm{DIFF}_{m-1}(\Sigma^{\mathrm{p}}_{k+1})$. ∎

Chapter 5

Query Order

5.1 Introduction

The order in which information sources are accessed is important in every aspect of our life. Does this observation carry over into complexity theory? In complexity theory information sources are usually modeled by oracles. Oracles are used for a number of purposes, one of them being relativization with full complexity classes. Relativization with full complexity classes is routinely done in structural complexity theory. The most well known application can be found in the definition of the polynomial hierarchy. However, until now relativization with full complexity classes has been limited to allow just one oracle from one single complexity class per machine. Allowing a base machine to access two different oracles from different complexity classes naturally leads to the question of whether the order in which the two different oracles are accessed is crucial for the computational power of the so defined class. Query order is a task that has never before been studied in complexity theory.

In this chapter we study the importance of query order. In Section 5.2 we ask whether the order of queries is crucial if the oracle sets are taken from different levels of the boolean hierarchy. We show that the order of queries to levels of the boolean hierarchy is important for the computational power of the resulting class unless the polynomial hierarchy collapses. We achieve this by characterizing query order classes in terms of reducibility closures of NP. In particular, we prove that for $j, k \geq 1$, $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} = \mathrm{R}^{\mathrm{p}}_{j+2k-1\text{-tt}}(\mathrm{NP})$ if j is even and k is odd, and $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} = \mathrm{R}^{\mathrm{p}}_{j+2k\text{-tt}}(\mathrm{NP})$ otherwise. This result has applications for the case of more complicated tree-like query structures, where similar characterizations are obtained.

In Section 5.3 we study query order in the polynomial hierarchy. Section 5.3 is divided into two parts. The first part gives an overview over query order results in the polynomial hierarchy that have been obtained since the results of Section 5.2 first appeared in [HHW95]. The most interesting query order classes, one query to each of two oracles from different levels of the polynomial have been studied by Hemaspaandra, Hemaspaandra, and Hempel [HHH98b]. In a nutshell, query order does not matter in the polynomial hierarchy. Generalizing query order classes to more than one query in each round, Beigel and Chang [BC] have shown an analogous result when the two oracles are from different levels of the polynomial hierarchy. The order of (parallel) query rounds does not matter. They left open the case that the two oracles are taken from the same level, a question we will solve in the second part of Section 5.3. Wagner [Wag98] has proven several results that show the close connection between query order classes in the polynomial hierarchy and Selivanov's generalized boolean hierarchies [Sel94a, Sel95].

In the second part of Section 5.3 we show that also in the case that two consecutive rounds of parallel queries are made to oracles from the same level of the polynomial hierarchy, the order of query rounds is irrelevant, for all $i, j, k \geq 1$, $\mathrm{P}_{j,k\text{-tt}}^{\Sigma_i^p:\Sigma_i^p} = \mathrm{P}_{k,j\text{-tt}}^{\Sigma_i^p:\Sigma_i^p}$. However, in general the two rounds of queries can not be made simultaneously unless the polynomial hierarchy collapses. Our result follows from a general characterization that we prove; for all $i, j, k \geq 1$, $\mathrm{P}_{j,k\text{-tt}}^{\Sigma_i^p:\Sigma_i^p} = \mathrm{R}_{j+jk+k\text{-tt}}^{p}(\Sigma_i^p)$.

In Section 5.4 we give a short overview over results that grew out of the study of query order classes or are related to query order.

5.2 Query Order in the Boolean Hierarchy

We ask whether the order of queries matters in the boolean hierarchy. In particular, for classes BH_j and BH_k from the boolean hierarchy [CGH+88, CGH+89], we ask whether one question to a BH_j oracle followed by one question to a BH_k oracle is more powerful than one question to a BH_k oracle followed by one question to a BH_j oracle. That is, we seek the relative powers of the classes $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ and $\mathrm{P}^{\mathrm{BH}_k:\mathrm{BH}_j}$.

All the results of Section 5.2 follow from a general characterization of classes of the form $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$. We prove via the mind change technique that, for $j, k \geq 1$:

$$\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} = \begin{cases} \mathrm{R}_{j+2k-1\text{-tt}}^{p}(\mathrm{NP}) & \text{if } j \text{ is even and } k \text{ is odd,} \\ \mathrm{R}_{j+2k\text{-tt}}^{p}(\mathrm{NP}) & \text{otherwise.} \end{cases}$$

This shows that in almost all cases, $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ is so powerful that it can do anything that can be done with $j + 2k$ truth-table queries to NP. Since, based on the answer to the first BH_j query, there are two possible BH_k queries that might follow, $j + 2k$ is exactly the number of queries asked in a brute force truth-table simulation of $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$. Thus, our result shows that (in almost all cases) the power of the class is not reduced by the nonlinear structure of the $j + 2k$ queries underlying $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$—that is, the power is not reduced by the fact that in any given run only $j + k$ underlying NP queries will be even implicitly asked (via the BH_j query and the one asked BH_k query). We say "in almost all cases" as if j is even and k is odd, we prove there is a power reduction of exactly one level.

In Subsection 5.2.2 we consider classes of the form $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k,\mathrm{BH}_l}$ and also classes with a more complicated query structure. We generalize the above characterization to apply broadly to classes with tree-like query structure.

5.2.1 The Base Case

In this section we will handle classes only of the form $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$. We show that for no j, k, j', and k' are $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ and $\mathrm{P}^{\mathrm{BH}_{j'}:\mathrm{BH}_{k'}}$ incomparable. The main theorem of this section, Theorem 5.2.6 shows that for all $j,k \geq 1$, $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} = \mathrm{R}^{\mathrm{p}}_{j+2k-1\text{-tt}}(\mathrm{NP})$ if j is even and k is odd, and $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} = \mathrm{R}^{\mathrm{p}}_{j+2k\text{-tt}}(\mathrm{NP})$ otherwise. Our proof employs the mind change technique, which predates complexity theory. In particular, we show that $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ has at most $j+2k$ ($j+2k-1$ if j is even and k is odd) mind changes, and that BH_{j+2k} (BH_{j+2k-1} if j is even and k is odd) is contained in $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$.

The mind change technique or equivalent manipulation (see [Wag79] for an early application in automata theory) was applied to complexity theory in each of the early papers on the boolean hierarchy, including the work of Cai et al. [CGH+88] (see also [Wec85, CH86]), Köbler et al. [KSW87], Wagner [Wag90], and Beigel [Bei91]. These papers use mind changes for a number of purposes. Most crucially they use the maximum number of mind changes (what a mind change is will soon be made clear) of a class as an upper bound that can be used to prove that the class is contained in some other class. In the other direction, they also use the number of mind changes that certain classes—especially the classes of the boolean hierarchy due to their normal form as nested subtractions of telescoping sets [CGH+88]—possess to show that they can simulate other classes. Even for classes that have the same number of mind changes, relativized separations are obtained via showing that the mind changes are of different character (mind change sequences are of two types, depending on whether they start with acceptance or rejection). The technique has also proven useful in many other more recent papers, e.g., [CK96, Cha91, BCO93] (see also Chapter 3).

To make clear the basic nature of mind change arguments, in a simple form, we give an example. We informally argue that each set that is m-truth-table reducible to NP is in fact in $\mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_m)$.

Lemma 5.2.1 For every $m \geq 1$, $\mathrm{R}^{\mathrm{p}}_{m\text{-tt}}(\mathrm{NP}) = \mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_m)$.

Proof: This fact (stated slightly differently) is due to Köbler et al. [KSW87], and the proof flavor presented here is most akin to the approach of Beigel [Bei91]. Consider an m-truth-table reduction to an NP set, F. Let L be the language accepted by the m-truth-table reduction to F. Consider some input x and without loss of generality assume m queries are generated. Let us suppose for the moment that the reduction rejects when all m queries receive the answer "no." Consider the m-dimensional hypercube such that one dimension is associated with each query (0 in that dimension means the query is answered no and 1 means it is answered yes). So the origin is associated with all queries getting the answer "no" and the point (1,1,...,1) is associated with all queries getting the answer "yes." Now, also label each vertex with either A (accept) or R (reject) based on what the truth-table would do given the answers represented by that vertex. So under our supposition, the origin has the label R. Finally, label each vertex with an integer as follows. Label the origin with 0. Inductively label each remaining vertex with the *maximum* integer induced by the vertices that immediately precede it (i.e., those that are the same as it except one yes answer has been changed to a no answer). A preceding vertex v with integer label i induces in a successor v' the integer $i+1$ if v and v' have different A/R labels, and i if they have the same label. Note that vertices given even labels correspond to rejection and those given odd labels correspond to acceptance. Informally, a mind change is just changing one or more strings from no to yes in a way that moves us from a vertex labeled i to one labeled $i+1$. For $1 \leq i \leq m$, let B_i be the NP set that accepts x if (in the queries/labeling generated by the action of the truth-table on input x) for some vertex v labeled i all the queries v claims are yes are indeed in the NP set F. Note that $B_1 \supseteq B_2 \supseteq B_3 \supseteq \cdots$, as if a node labeled v is in B_j, $j \geq 2$, then certainly its predecessor node with label $j-1$ must be in B_{j-1}, as that predecessor represents a subset of the strings v represents. But now note that L is exactly $B_1 - (B_2 - (B_3 - (\cdots - (B_{m-1} - B_m)\cdots)))$. Why? Let the vertex w (say with integer label i_w) represent the true answers to the queries. Note that by construction, $x \in B_q$ for all $q \leq i_w$ but $x \notin B_q$ for any $q > i_w$. As the B_i were alternating in terms of representing acceptance and rejection, and given the format $B_1 - (B_2 - (B_3 - (\cdots - (B_{m-1} - B_m)\cdots)))$, the set $B_1 - (B_2 - (B_3 - (\cdots - (B_{m-1} - B_m)\cdots)))$, will do exactly what B_{i_w} represents, namely, the action on the correct answers. Thus, we have just given a proof that an m-truth-table reduction that rejects whenever all answers are no can be simulated by a set in BH_m. Of course, one cannot validly assume that the reduction rejects whenever all answers are no. But it is not hard to see (analogously to the above) that the case of inputs where the reduction accepts when all answers are no can (analogously to the above) be handled via the complement of a BH_m set, and that (since what the truth-table reduction does when all answers are no is itself polynomial-time computable) via a set in $\mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_m)$ we can accept an arbitrary set in $\mathrm{R}^{\mathrm{p}}_{m\text{-tt}}(\mathrm{NP})$. Of course, it is clear by brute force simulation that $\mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_m) \subseteq \mathrm{R}^{\mathrm{p}}_{m\text{-tt}}(\mathrm{NP})$, and so it holds that $\mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_m) = \mathrm{R}^{\mathrm{p}}_{m\text{-tt}}(\mathrm{NP})$. ∎

What actually is being shown above is that $\mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_m)$ can handle m appropriately structured mind changes, starting either from reject or accept. In the following theorem, the crucial things we show are that (a) $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ can simulate, starting at either accept or reject, $j+2k$ (respectively, $j+2k-1$) mind changes if j is odd or k is even (respectively, if j is even and k is odd), and (b) for j even and k odd, $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ can *never* have more than $j+2k-1$ mind changes. We achieve (b) by examining the possible mind change flow of a $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ machine, j even and k odd, and showing that either a mind change is flagrantly wasted, or a certain underlying graph has an odd length directed cycle (which thus is not two-colorable, and from this will lose one mind change).

Since our arguments in the proofs of this section use paths in hypercubes, we will find useful the concept of an ascending path in a hypercube. Let $K = \{0,1\}^d$ be the d-dimensional hypercube. Then every path p in K can be described as a linear combination of unit vectors $u_1, \ldots, u_d$, where u_i is the ith unit vector. We call p an ascending path in K leading from $(0,0,\ldots,0)$ to v if and only if it can be identified with a sum

$$u_{i_1} + u_{i_2} + \cdots + u_{i_n}$$

of distinct unit vectors u_ν, such that the vertices of this path p are

$$v_0 = (0,\ldots,0), v_1 = u_{i_1}, v_2 = u_{i_1} + u_{i_2}, \ldots, v = u_{i_1} + u_{i_2} + \cdots + u_{i_n}.$$

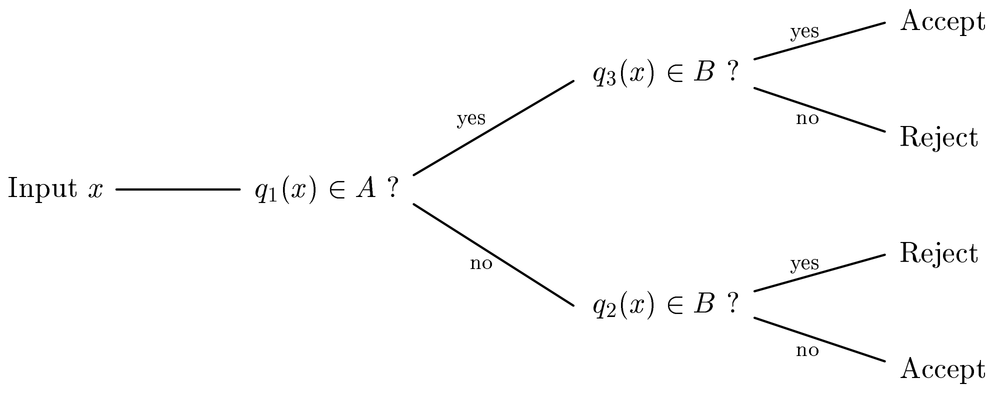


Figure 5.1: Query Tree of a Machine Having Acceptance Scheme (2)

We will call this sum the description of p. Note that the order of the u's matters, as a permutation of the u's results in another path. We call p an ascending path (without specifying starting point and endpoint) if p is an ascending path leading from $(0, 0, \ldots, 0)$ to $(1, 1, \ldots, 1)$.

Before turning to results, we will first study the structure of ascending paths in labeled hypercubes and give some necessary definitions. Building upon them, we will then prove Lemma 5.2.5, which states that $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ can handle exactly $j + 2k$ ($j + 2k - 1$ if j is even and k is odd) mind changes.

Let M be a $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ machine with oracles $A \in \mathrm{BH}_j$ and $B \in \mathrm{BH}_k$ and let $x \in \Sigma^*$. On input x, M first makes a query $q_1(x)$ to A and then if the answer to the first query was no asks query $q_2(x)$ to B and if the answer to the first query was yes asks query $q_3(x)$ to B (see also Figure 5.1). Without loss of generality assume that on every input x exactly two queries are asked.

Every set $C \in \mathrm{BH}_l$ can be written as the nested difference of sets $C_1, C_2, \ldots, C_l \in \mathrm{NP}$

$$C = C_1 - (C_2 - (\cdots - (C_{l-1} - C_l) \cdots))$$

and following Cai et al. [CGH+88] we even can assume that

$$C_l \subseteq C_{l-1} \subseteq \cdots \subseteq C_2 \subseteq C_1.$$

Hence a query "$q \in C$?" can certainly be solved via l queries "$q \in C_1$?," "$q \in C_2$?," ... , "$q \in C_l$?"

In light of this comment, we let

$$A = A_1 - (A_2 - (\cdots - (A_{j-1} - A_j) \cdots)) \quad \text{where} \quad A_i \in \mathrm{NP} \quad \text{for} \quad i = 1, 2, \ldots, j \quad \text{and} \quad A_j \subseteq \cdots \subseteq A_1, \text{ and}$$

$$B = B_1 - (B_2 - (\cdots - (B_{k-1} - B_k) \cdots)) \quad \text{where} \quad B_i \in \mathrm{NP} \quad \text{for} \quad i = 1, 2, \ldots, k \quad \text{and} \quad B_k \subseteq \cdots \subseteq B_1.$$

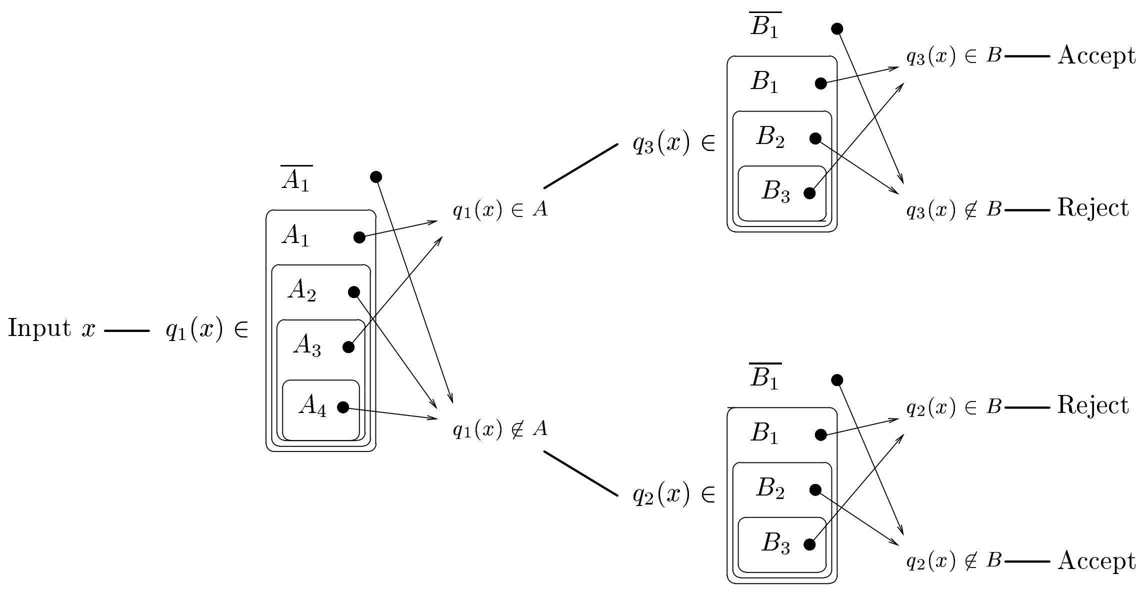


Figure 5.2: Refined Query Tree of a $\mathrm{P}^{\mathrm{BH}_4:\mathrm{BH}_3}$ Machine Having Acceptance Scheme (2): The inclusion structure of the underlying NP sets is shown. A black dot indicates a potential membership situation for the asked query whereas its attached arrow points to the resulting oracle answer received by the base machine.

An example displaying the connection between the membership of a query with respect to the underlying NP sets and the actual oracle answer received by the base machine is given in Figure 5.2.

For the sake of definiteness let us assume that the queries

$$q_1(x) \in A_1, \ldots, q_1(x) \in A_j, q_2(x) \in B_1, \ldots, q_2(x) \in B_k, q_3(x) \in B_1, \ldots, q_3(x) \in B_k$$

correspond in this order to the $j + 2k$ dimensions of the $(j + 2k)$-dimensional hypercube $H = \{0,1\}^{j+2k}$. More precisely, a vector $(a_1, \ldots, a_{j+2k}) \in H$ is understood to consist of the answers to the above-mentioned queries, where 0 means no and 1 means yes.

Since a query "$q \in C$?" for some $C \in \mathrm{BH}_l$ and $C = C_1 - (C_2 - (\cdots(C_{l-1} - C_l)\cdots))$ can be solved by evaluating the answers to "$q \in C_1$?," "$q \in C_2$?," ..., "$q \in C_l$?" every node $v \in H$ gives us answers to "$q_1(x) \in A$?" (by evaluating the first j components of v), to "$q_2(x) \in B$?" (by evaluating the k components of v that immediately follow the first j components of v) and to "$q_3(x) \in B$?" (by evaluating the last k of v's components). This gives us a labeling of all vertices of H. We simply assign label A (Accept) to vertex $v \in H$ if $M^{A:B}(x)$ accepts if the answers to the two asked questions are as determined by v. If $M^{A:B}(x)$ rejects in this case we assign label R (Reject) to v.

So let $H_M(x)$ be the $(j+2k)$-dimensional hypercube labeled according to $M^{A:B}(x)$. The number of mind changes on an ascending path p of $H_M(x)$ leading from $(0, 0, \ldots, 0)$ to a vertex t is by definition the number of label changes when moving from $(0,0,\ldots,0)$ to t along p. The number of mind changes of an internal node v of $H_M(x)$ is the maximum number of mind changes on an ascending path leading from $(0,0,\ldots,0)$ to v. And finally, the number of mind changes of a $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ machine M is by definition the maximum number (we take the maximum over all $x \in \Sigma^*$) of mind changes of the vertex $(1,1,\ldots,1)$ in $H_M(x)$; in other words, this number is the maximum number of label changes on an ascending path in $H_M(x)$ for some $x \in \Sigma^*$.

We say we lose a mind change (between two adjacent vertices v_i and v_{i+1}) along an ascending path if when moving from v_i to v_{i+1} the machine does not change its acceptance behavior.

One can easily verify the following fact:

Fact 5.2.2 If M is a $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ machine such that on input x the acceptance behavior is independent of the answer to one or more of the two possible second queries (that is, if for at least one of the second queries both a yes and a no answer yield the same acceptance or rejection behavior), then we lose at least one mind change on every path in $H_M(x)$.

So from now on let $M^{A:B}(x)$ be a $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ machine that has on input x one of the following four acceptance schemes (for an example see Figure 5.1).

(1) M accepts if and only if exactly one of the two sequential queries is answered yes.

(2) M accepts if and only if either both or neither of the two asked queries is answered yes (see also Figure 5.1).

(3) M accepts if and only if the second query is answered yes.

(4) M accepts if and only if the second query is answered no.

Fact 5.2.3 If p is an ascending path in $H_M(x)$ such that p contains adjacent vertices v and $v + u_d$ such that

$$d \leq j \text{ and the } (d')\text{th component of } v \text{ is } 0 \text{ for some } d' < d,$$

then p loses a mind change.

Proof: Since $A \in \mathrm{BH}_j$ and thus $A = A_1 - (A_2 - (\cdots - (A_{j-1} - A_j)\cdots))$ and there is a 0 in the (d')th component of v and $v + u_d$, both vertices yield the same answer to "$q_1(x) \in A$?" The 1 in the dth component of $v + u_d$ has no affect at all on the answer to "$q_1(x) \in A$?" and so on the outcome of $M^{A:B}(x)$. Hence, both vertices have the same label and p loses a mind change. ∎

Similarly, one can prove that if p is an ascending path and p contains two adjacent vertices v and $v + u_d$ such that $j < d' < d \leq j + k$ and the (d')th component of v is 0 or $j + k < d' < d \leq j + 2k$ and the (d')th component of v is 0 then p also loses one mind change.

So from now on let us focus only on paths p that change their first j, second k and last k dimensions from the smallest to the highest dimension in each group. This allows us to simplify the description of paths as follows. Let e_1 be the following operator on H:

$$e_1((a_1,\ldots,a_{j+2k})) = \begin{cases} (a_1,\ldots,a_{i-1},1,\ldots,a_{j+2k}) & \text{if } i \le j,\ a_i = 0 \wedge (\forall j : j < i)[a_j = 1], \\ (a_1,\ldots,a_{j+2k}) & \text{otherwise.} \end{cases}$$

The operators e_2 and e_3 act on the index groups $(j+1,\ldots,j+k)$ and $(j+k+1,\ldots,j+2k)$, respectively, in the same manner: the zero component with smallest index among the zero components is incremented by 1. The only reasonable paths to consider are those emerging from repeated applications of e_1, e_2 and e_3 to $(0,\ldots,0)$. We will use $(e_{i_1}, e_{i_2},\ldots,e_{i_{j+2k}})$ to denote the path with vertices $v_0 = (0,\ldots,0)$, $v_1 = e_{i_1}(v_0), v_2 = e_{i_2}(v_1),\ldots$, $v_{j+2k} = e_{i_{j+2k}}(v_{j+2k-1}) = (1,1,\ldots,1)$.
The next fact gives sufficient conditions for an ascending path to lose a mind change, namely,

Fact 5.2.4 On any ascending path p a mind change loss occurs if:

Case 1.1 there is an e_2 after an odd number of e_1's in the description of p, or

Case 1.2 there is an e_3 after an even number of e_1's in the description of p, or

Case 2 the description of p contains a sequence of odd length at least 3 that starts and ends with e_1 and contains no other e_1's.

Proof: We will call the occurrence of Case 1.1 (Case 1.2) in p an "e_2-loss" ("e_3-loss") and the occurrence of Case 2 an "odd episode." In general we call a subpath of p of length at least 3 that starts and ends with e_1 and contains no other e_1 an episode.

Intuitively p loses a mind change in the case of Case 1.1 (1.2), since in the actual computation $M(x)$ does not really ask query $q_2(x)$ ($q_3(x)$) and so a change in the answers to the k underlying NP queries of $q_2(x)$ ($q_3(x)$) does not affect the outcome of the overall computation.

Intuitively in Case 2 the following argument holds. If the description of p contains an odd episode, say starting with $e_{i_l} = e_1$ and ending with $e_{i_{l'}} = e_1$, then $v_{l-1}, v_l,\ldots,v_{l'}$ form an even-length subpath p' of p. If the odd episode contains both e_2's and e_3's then note that Case 1 applies and we are done. In fact due to Case 1, we may hence forward assume the odd episode, between the starting and the ending e_1's, has only e_2's (respectively e_3's), if we have an even (respectively odd) number of e_1's up to and including the e_1 starting the odd episode. So in this case v_{l-1} and $v_{l'}$ have the same label Accept/Reject. The acceptance behavior of $M^{A:B}(x)$ due to v_{l-1} and $v_{l'}$ is the same, because after two e_1's the answer to "$q_1(x) \in A$?" is the same as it was before the two e_1's, and the e_2's (e_3's) have not influenced the answer to $q_3(x)$ ($q_2(x)$). Thus we have a subpath of even length, namely, $v_{l-1}, v_l,\ldots,v_{l'}$, whose starting point and endpoint have the same Accept/Reject label. To assign to each vertex of this path an Accept/Reject label in such a way that no mind changes are lost is equivalent to the impossible task of 2-coloring an odd cycle. Hence we lose at least one mind change for every occurrence of an odd episode. ▮

Before stating and proving the main theorem of this section, we show the following lemma, Lemma 5.2.5, which tells how many mind changes $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ can handle. We say a complexity class $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ can handle exactly m mind changes if and only if (a) no $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ machine has more than m mind changes and (b) there is a specific $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ machine that has m mind changes. It is known (see, e.g., [CGH$^+$88, KSW87, Bei91]) that $\mathrm{R}^{\mathrm{p}}_{k\text{-tt}}(\mathrm{NP})$ can handle exactly k mind changes.

Lemma 5.2.5 The class $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ can handle exactly m mind changes, where

$$m = \begin{cases} j+2k-1 & \text{if } j \text{ is even and } k \text{ is odd,} \\ j+2k & \text{otherwise.} \end{cases}$$

Proof: We first consider the case in which j is even and k is odd.

We want to argue that for every $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ machine M and every $x \in \Sigma^*$, on every ascending path in the $j+2k$ dimensional, appropriately labeled, hypercube $H_M(x)$ there are at most $j+2k-1$ mind changes. Let $x \in \Sigma^*$ and M be a $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ machine with the oracles A and B. Due to Facts 5.2.3 and 5.2.2, it suffices to consider a $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ machine M with one of the four previously mentioned acceptance schemes on input x and to show that every path p having the introduced description loses at least one mind change. Let $M(x)$ be such a machine and p be such a path. There are two possibilities.

Case A The description of p contains an e_2-loss or an e_3-loss.

According to Fact 5.2.4, p loses at least one mind change.

Case B The description of p contains neither an e_2- loss nor an e_3-loss.

Hence the description of p consists of blocks of consecutive e_2's and e_3's separated by blocks of e_1's. Since the description of p contains k e_3's and k is odd, there is a block of e_3's of odd size in p. Since we have no e_3-loss and j is even this block is surrounded by e_1's. Thus we have an odd episode in the description of p and, according to Fact 5.2.4, p loses a mind change.

So no $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ machine can realize more than $j+2k-1$ mind changes.

It remains to show that there is a $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ machine and an input $x \in \Sigma^*$ such that in the associated hypercube $H_M(x)$ there is a path having exactly $j+2k-1$ mind changes.

Let us consider the path p_0,

$$p_0 = (\underbrace{e_2, e_2, \ldots, e_2}_{k}, \underbrace{e_1, e_1, \ldots, e_1}_{j-1}, \underbrace{e_3, e_3, \ldots, e_3}_{k}, e_1).$$

Consider the deterministic oracle machine W that asks two sequential queries (all three underlying queries differ pairwise) and accepts an input x if and only if the second query of $W(x)$ was answered yes (acceptance scheme (3)). We know as just shown that all ascending path of $H_W(x)$ have at most $j+2k-1$ mind changes. Note that for every $x \in \Sigma^*$ the path p_0 loses only one mind change and thus $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ can handle exactly $j+2k-1$ mind changes.

We now turn to the j is odd or k is even case of the lemma being proven.

Since our hypercube has in all three cases $j+2k$ dimensions $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ can handle at most $j+2k$ mind changes.

If j is odd, we consider the path

$$p_1 = (\underbrace{e_2, e_2, \ldots, e_2}_{k}, \underbrace{e_1, e_1, \ldots, e_1}_{j}, \underbrace{e_3, e_3, \ldots, e_3}_{k})$$

and the machine with three pairwise different underlying queries having for every input x acceptance scheme (3) or (1) for k odd or even, respectively. If j is even and k even we consider path p_0 and the machine having acceptance scheme (1) for every input.

In each of these cases the considered machine changes its mind along the associated path exactly $j+2k$ times. Hence for j odd or k even the class $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ can handle exactly $j+2k$ mind changes. ▮

Now we are ready to prove our main theorem of this section.

Theorem 5.2.6 For $j, k \geq 1$,

$$\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} = \begin{cases} \mathrm{R}^{\mathrm{p}}_{j+2k-1\text{-tt}}(\mathrm{NP}) & \text{if } j \text{ is even and } k \text{ is odd,} \\ \mathrm{R}^{\mathrm{p}}_{j+2k\text{-tt}}(\mathrm{NP}) & \text{otherwise.} \end{cases}$$

Proof: In order to avoid unnecessary case distinctions we prove the fact for arbitrary j and k and simply denote the appropriate number of mind changes by m, namely, $j+2k-1$ if j is even and k is odd and $j+2k$ otherwise (see Lemma 5.2.5). First, we would like to show that $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} \subseteq \mathrm{R}^{\mathrm{p}}_{m\text{-tt}}(\mathrm{NP})$. We show this by explicitly giving the appropriate truth-table-reduction.

Let $A \in \mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ and let m be the number of mind changes (see Lemma 5.2.5) the class $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ can handle. Let M be a deterministic oracle machine, witnessing $A \in \mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$, via the sets $S_1 \in \mathrm{BH}_j$ and $S_2 \in \mathrm{BH}_k$. It is not hard to verify that the set $Q = \{\langle x, k\rangle \mid M(x) \text{ has at least } k \text{ mind changes}\}$ is an NP set. Note that if $M(x)$ on a particular input x rejects (respectively accepts) if both queries have the answer "no" then $M^{S_1:S_2}(x)$ accepts if and only if the node (of the implicit hypercube) associated with the actual answers has an odd (respectively even) number of mind changes.

Define the variables $o, y_1, y_2, \ldots, y_m$ and the m-ary boolean function α:

$o = 0$ if $M^{S_1:S_2}(x)$ rejects if both queries are answered no,
$o = 1$ if $M^{S_1:S_2}(x)$ accepts if both queries are answered no,

$$\begin{aligned} y_1 &= \langle x, 1\rangle, \\ y_2 &= \langle x, 2\rangle, \\ y_3 &= \langle x, 3\rangle, \\ &\vdots \\ y_m &= \langle x, m\rangle, \end{aligned}$$

and $$\alpha(z_1, z_2, \ldots, z_m) = 1 \iff (\max\{l \mid z_l = 1\} + o) \equiv 1 \pmod 2).$$

Clearly we can compute the just defined variables for a given x and also evaluate the function α at $(\chi_Q(y_1), \chi_Q(y_2), \dots, \chi_Q(y_m))$ in polynomial time. And finally, we have $x \in A \iff \alpha(\chi_Q(y_1), \chi_Q(y_2), \dots, \chi_Q(y_m)) = 1$. Thus $A \in \mathrm{R}^{\mathrm{p}}_{m\text{-tt}}(\mathrm{NP})$.

It remains to show that $\mathrm{R}^{\mathrm{p}}_{m\text{-tt}}(\mathrm{NP}) \subseteq \mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$. Recall $\mathrm{R}^{\mathrm{p}}_{k\text{-tt}}(\mathrm{NP}) = \mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_k)$ from Lemma 5.2.1. Since the class $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ is closed under $\leq^{\mathrm{p}}_{1\text{-tt}}$ reductions it suffices to prove $\mathrm{BH}_m \subseteq \mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$.

So let $B \in \mathrm{BH}_m$. Following Cai et al. [CGH+88] we may assume that the set B is of the form $B = B_1 - (B_2 - (B_3 - (\cdots - (B_{m-1} - B_m)\cdots)))$ with $B_1, B_2, \dots, B_m \in \mathrm{NP}$ and $B_1 \supseteq B_2 \supseteq \cdots \supseteq B_m$.

We show $B \in \mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ by using ideas of the second part of the proof of Lemma 4.1, namely, by implementing the specific good path p_0, respectively p_1. B is accepted by a $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ machine $M^{O_1:O_2}$ as follows:

Case 1 j is odd.
Define the two oracle sets O_1 and O_2,

$$O_1 = B_{k+1} - (B_{k+2} - (\cdots - (B_{k+j-1} - B_{k+j})\cdots)),$$

and

$$\begin{aligned} O_2 = \{\langle y, 2\rangle \mid y \in B_1 - (B_2 - (\cdots - (B_{k-1} - B_k)\cdots))\} \cup \\ \{\langle y, 3\rangle \mid y \in B_{j+k+1} - (B_{j+k+2} - (\cdots - (B_{j+2k-1} - B_{j+2k})\cdots))\}. \end{aligned}$$

Note that $O_1 \in \mathrm{BH}_j$ and $O_2 \in \mathrm{BH}_k$. On input x M first queries "$x \in O_1$." In case of a no answer $M(x)$ queries $\langle x, 2\rangle \in O_2$ and in case of a yes answer to the first query $M(x)$ asks $\langle x, 3\rangle \in O_2$.

Case 1.1 k is odd.
$M(x)$ accepts if and only if the second query is answered yes.

Case 1.2 k is even.
$M(x)$ accepts if and only if exactly one of the two queries is answered yes.

Case 2 j is even.
Define the two oracle sets O_1 and O_2,

$$O_1 = B_{k+1} - (B_{k+2} - (\cdots - (B_{k+j-1} - B_m)\cdots)),$$

and

$$\begin{aligned} O_2 = \{\langle y, 2\rangle \mid y \in B_1 - (B_2 - (\cdots - (B_{k-1} - B_k)\cdots))\} \cup \\ \{\langle y, 3\rangle \mid y \in B_{j+k} - (B_{j+k+1} - (\cdots - (B_{m-2} - B_{m-1})\cdots))\}. \end{aligned}$$

Note that $O_1 \in \mathrm{BH}_j$ and $O_2 \in \mathrm{BH}_k$. On input x M first queries "$x \in O_1$." In case of a no answer $M(x)$ queries $\langle x, 2\rangle \in O_2$ and in case of a yes answer to the first query $M(x)$ asks $\langle x, 3\rangle \in O_2$.

Case 2.1 k is odd.
$M(x)$ accepts if and only if the second query is answered yes.

Case 2.2 k is even.
$M(x)$ accepts if and only if exactly one of the two queries is answered yes.

∎

It is interesting to note which properties of NP are actually required in the above proof for the result to hold. The proof essentially rests on the fact that the key set Q (describing that, for given x and m, the $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ machine M on input x has at least m mind changes) is an NP set. So considering an arbitrary underlying class $\mathcal{C}$, for proving $Q \in \mathcal{C}$ it suffices to note that Q is in the class $\exists^b \cdot \mathrm{R}^{\mathrm{p}}_{\text{c-btt}}(\mathcal{C})$,[1] and to assume that $\mathcal{C}$ be closed under $\exists^b$ and conjunctive bounded-truth-table reductions. Indeed, the $\exists^b$ quantifier describes that there is a path in the boolean hypercube $H_M(x)$, and via the $\leq^{\mathrm{p}}_{\text{c-btt}}$-reduction it can be checked that this path is an ascending path and all the answers the vertices on that path claim to be yes answers indeed correspond to query strings that belong to the class $\mathcal{C}$. Similar observations have been stated in earlier papers [Bei91, BCO93]. In terms of the present paper, note in particular that the assertion of Theorem 5.2.6 holds true for all classes $\mathcal{C}$ closed under union, intersection, and polynomial-time many-one reductions. $\mathrm{C}_{=}\mathrm{P}$, R, and FewP all have these closure properties, to name just a few examples. If the underlying class $\mathcal{C}$ is closed under polynomially bounded $\exists$ quantification and unbounded conjunctive truth-table reductions, it is not hard to see that this analysis can even be done safely up to the case of logarithmically bounded query classes, as the number of paths in the hypercube is polynomial and thus generates a polynomial-sized disjunction.

From Theorem 5.2.6 we can immediately conclude that order matters for queries to the boolean hierarchy unless the boolean hierarchy itself collapses.

Corollary 5.2.7

1. If $(j = k) \vee (j \text{ is even and } k = j+1)$, $1 \leq j \leq k$, then $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} = \mathrm{P}^{\mathrm{BH}_k:\mathrm{BH}_j}$.

2. Unless the boolean hierarchy (and thus the polynomial hierarchy) collapses: for any $1 \leq j \leq k$, $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} \neq \mathrm{P}^{\mathrm{BH}_k:\mathrm{BH}_j}$ unless $(j = k) \vee (j \text{ is even and } k = j+1)$.

The corollary holds, in light of the theorem, simply because the boolean hierarchy and the bounded-truth-table hierarchy are interleaved [KSW87] in such a way that the boolean hierarchy levels are sandwiched between levels of the bounded-truth-table hierarchy, and thus if two different levels of the bounded-truth-table hierarchy are the same (say levels r and s, $r < s$), then some level (in particular, BH_{r+1}) of the boolean hierarchy is closed under complementation, and thus, by the downward separation property of the boolean hierarchy [CGH+88], the boolean hierarchy would collapse. Furthermore, Kadin [Kad88] has shown that if the boolean hierarchy collapses then the polynomial hierarchy collapses,

[1]Here, $\leq^{\mathrm{p}}_{\text{c-btt}}$ denotes the conjunctive bounded-truth-table reducibility, and for any class $\mathcal{K}$, $\exists^b \cdot \mathcal{K}$ is defined to be the class of languages A for which there exists a set $B \in \mathcal{K}$ and a constant bound m such that $x \in A$ if and only if there exists a string y of length at most m with $\langle x, y\rangle \in B$.

and Wagner, Chang and Kadin, and Beigel, Chang, and Ogihara have improved the strength of this connection [Wag87, Wag89, CK96, BCO93] (see Chapter 3). The strongest known connection is established in Theorem 3.3.1; for $m \geq 2$, if $\mathrm{BH}_m = \mathrm{coBH}_m$, then $\mathrm{PH} = \mathrm{BH}_m \boldsymbol{\Delta} \mathrm{DIFF}_{m-1}(\Sigma_2^{\mathrm{p}})$.

In light of this discussion, we can make more clear exactly what collapse is spoken of in the second part of the above corollary. In particular, the collapse of the polynomial hierarchy is (at least) to $\mathrm{BH}_{k+2j+1} \boldsymbol{\Delta} \mathrm{DIFF}_{k+2j}(\Sigma_2^{\mathrm{p}})$. Though one level is gained by the $m-1$ in the Theorem 3.3.1 connection between the boolean hierarchy and the polynomial hierarchy, one level is lost in the collapse of the boolean hierarchy that follows from a given collapse in the bounded-truth-table hierarchy. Observe that in light of the results of Chapter 4, especially Theorem 4.2.4, a slightly deeper collapse of the polynomial hierarchy can be concluded if query order classes in the boolean hierarchy over some Σ_k^{p}, $k > 1$, are equal.

5.2.2 The General Case

In the previous subsection, we studied classes of the form $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$. We completely characterized them in terms of reducibility hulls of NP and noted that in this setting the order of access to different oracles matters quite a bit. What can be said about, for example, the class $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k:\mathrm{BH}_l}$? Is it equal to $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k,\mathrm{BH}_l}$? We will see that the answer is no in certain cases. Even more generally, what can be said about the classes of languages that are accepted by deterministic oracle machines with tree-like query structures and with each query being made to a (potentially) different oracle from a (potentially) different level of the boolean hierarchy? Is it possible that with a more complicated query structure we might lose even more than the one mind change lost in the case of $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ with j even and k odd? From the results of the section, it will be clear that the answer to this question is yes; mind changes can, in certain specific circumstances, accumulate.

First of all, we can immediately derive a characterization of the class $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k,\mathrm{BH}_l}$ from the results of the previous section, namely,

Theorem 5.2.8 For $j, k, l \geq 1$,

$$\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k,\mathrm{BH}_l} = \begin{cases} \mathrm{R}^{\mathrm{p}}_{j+k+l-1\text{-tt}}(\mathrm{NP}) & \text{if } j \text{ is even and } l \text{ is odd,} \\ \mathrm{R}^{\mathrm{p}}_{j+k+l\text{-tt}}(\mathrm{NP}) & \text{otherwise.} \end{cases}$$

Proof: Note that in Lemma 5.2.5 we handle the special case of $k = l$. However, notice that the mind change loss for j even and k odd is due only to the fact that the query made after the first query is answered yes is made to an oracle from an odd level, namely, k, of the boolean hierarchy. In particular, the mind change loss is not tied to the query we ask in case the first query is answered no. Thus we have that the class $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k,\mathrm{BH}_l}$ can handle exactly m mind changes where

$$m = \begin{cases} j+k+l-1 & \text{if } j \text{ is even and } l \text{ is odd,} \\ j+k+l & \text{otherwise.} \end{cases}$$

Similarly to the proof of Theorem 5.2.6 one can now show the equality we claim. ∎

Note that for every $j, k, l \geq 1$, we obviously have

$$\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k,\mathrm{BH}_l} = \mathrm{P}^{\mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_j):\mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_k),\mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_l)}$$

and thus the following corollary holds.

Corollary 5.2.9 For $j, k, l \geq 1$,

$$\mathrm{P}^{\mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_j):\mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_k),\mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_l)} = \begin{cases} \mathrm{R}^{\mathrm{p}}_{j+k+l-1\text{-tt}}(\mathrm{NP}) & \text{if } j \text{ is even and } l \text{ is odd,} \\ \mathrm{R}^{\mathrm{p}}_{j+k+l\text{-tt}}(\mathrm{NP}) & \text{otherwise.} \end{cases}$$

The last corollary is the key tool to use in evaluating any class of languages that are accepted by deterministic oracle machines with tree-like query structures and with each query being made to a (potentially) different oracle from a (potentially) different level of the boolean hierarchy.

We formalize some notions to use in studying this. Let T be a binary tree, not necessarily complete, such that each internal node v_i (a) has exactly two children, and (b) is labeled by a natural number n_i (whose purpose will be explained below). For such a tree T, define f_T by $f_T(v_i) = n_i$. Henceforward, we will write f for f_T in contexts in which T is clear. Let $root_T$ be the root of the tree (we will assign to this node the name v_1) and let LT_T and RT_T respectively be the left and right subtrees of the root. We will denote the class of sets that are accepted by a deterministic oracle machine with a T-like query structure by $\mathrm{P}^{(T)}$. Here the structure of the tree T gives the potential computation tree of every $\mathrm{P}^{(T)}$ machine in the sense that inductively if a query at node v is answered no (yes) we keep on moving through the tree in the left (right) subtree of v. And at each internal node v_i of T the natural number n_i gives the level of the boolean hierarchy from which the oracle queried at that node is taken.

For example consider the tree $\mathcal{T}$ (see Figure 5.3), in which $f(v_1) = 2$, $f(v_2) = 2$, $f(v_3) = 4$, $f(v_4) = 1$, and $f(v_5) = 3$. A $\mathrm{P}^{(\mathcal{T})}$ machine works as follows. The first query is made to a DP oracle. If the answer to that first query is no a second query is made to the DP oracle associated with v_2, and if the answer to the first query is yes the second query is made to the BH_4 oracle associated with v_3. A third query is made only if the answer to the first query is yes; in this case, the oracle set of the third query is in NP if the answer to the second query is no, and is in BH_3 if the answer to the second query is yes. Note that for every input $x \in \Sigma^*$ every $\mathrm{P}^{(\mathcal{T})}$ machine $M(x)$ assigns a label A (Accept) or R (Reject) to each leaf of $\mathcal{T}$ with its own specific acceptance behavior (which, in particular, may depend on x).

If T is the complete tree of depth 1 (i.e., a root plus two leaves), then by definition $m(T) = f(root_T)$, and otherwise define

$$m(T) = \begin{cases} f(root_T) + m(LT_T) + m(RT_T) - 1 & \text{if } f(root_T) \equiv 0 \pmod 2 \text{ and} \\ & m(RT_T) \equiv 1 \pmod 2, \\ f(root_T) + m(LT_T) + m(RT_T) & \text{otherwise.} \end{cases}$$

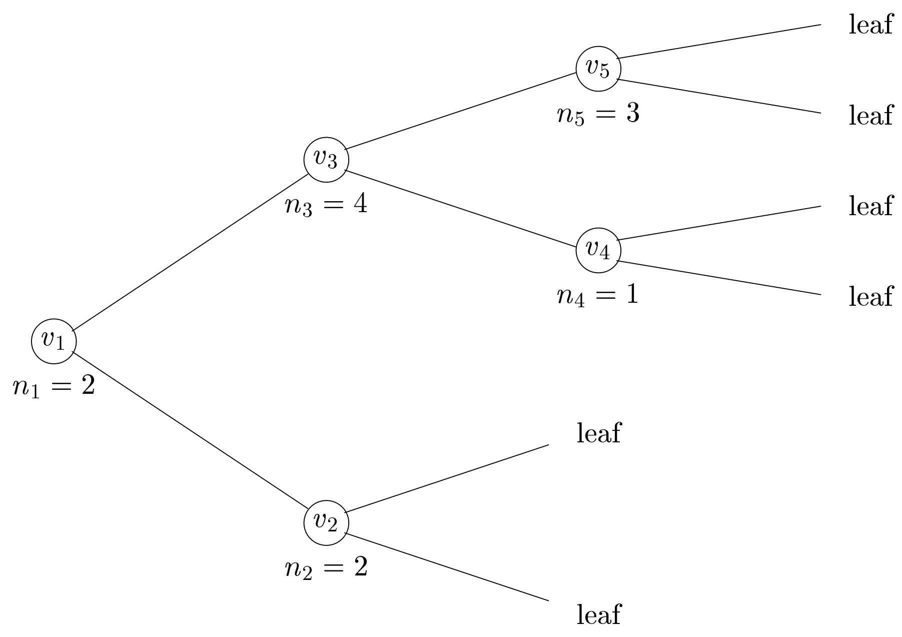


Figure 5.3: Example Tree $\mathcal{T}$

For our example tree $\mathcal{T}$ we have $m(\mathcal{T}) = 10$. The main theorem of this section will prove $m(T)$ determines the number of bounded-truth-table accesses to NP that completely characterizes the class $\mathrm{P}^{(T)}$. It follows from the main theorem that, for example, $\mathrm{P}^{(\mathcal{T})} = \mathrm{R}^{\mathrm{p}}_{10\text{-tt}}(\mathrm{NP})$.

Theorem 5.2.10 $\mathrm{P}^{(T)} = \mathrm{R}^{\mathrm{p}}_{m(T)\text{-tt}}(\mathrm{NP})$.

Proof: The proof consists of an obvious induction over the depth d of the tree. Note that the correctness of the base case of the induction, $d = 2$, is given by Theorem 5.2.8. The proof of the inductive step follows immediately from the obvious fact that

$$\mathrm{P}^{(T)} = \mathrm{P}^{\mathrm{BH}_{f(root_T)}:\mathrm{P}^{(LT_T)},\mathrm{P}^{(RT_T)}},$$

combined with Lemma 5.2.1 ($\mathrm{R}^{\mathrm{p}}_{k\text{-tt}}(\mathrm{NP}) = \mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_k)$) and Corollary 5.2.9. ∎

5.2.3 Remarks

Of course, the main theorem of this section, Theorem 5.2.6, applies far more generally. From it, for any j, k, j', and k', one can either immediately conclude $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} = \mathrm{P}^{\mathrm{BH}_{j'}:\mathrm{BH}_{k'}}$, or can immediately conclude that the classes $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ and $\mathrm{P}^{\mathrm{BH}_{j'}:\mathrm{BH}_{k'}}$ are not equal unless the polynomial hierarchy collapses to $\mathrm{BH}_{\min\{\alpha(j,k),\,\alpha(j',k')\}+1}\boldsymbol{\Delta}\mathrm{DIFF}_{\min\{\alpha(j,k),\,\alpha(j',k')\}}(\Sigma_2^{\mathrm{p}})$, where $\alpha(a,b)$ equals $a+2b-1$ if a is even and b is odd and $a+2b$ otherwise.

The point of Theorem 5.2.6 is that from the even/odd structure of $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ classes one can immediately tell their number of mind changes, and thus their strength, without having to do a separate, detailed, mind change analysis for each j and k pair. For example, one can quickly see that one query to DP followed by one query to BH_4 yields exactly the languages in $\mathrm{R}^{\mathrm{p}}_{10\text{-tt}}(\mathrm{NP})$.

Theorem 5.2.6 should be compared with the work of Agrawal, Beigel, and Thierauf [ABT96]. They prove (using different notation):

$$\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k+} = \begin{cases} \mathrm{BH}_{j+2k-1} & \text{if } j \not\equiv k \pmod 2, \\ \mathrm{BH}_{j+2k} & \text{otherwise.} \end{cases}$$

Note that this result is incomparable with the results of Theorem 5.2.6, as their result deals with a different and seemingly more restrictive acceptance mechanism. Some insight into the degree of restrictiveness of their acceptance mechanism, and its relationship to ours, is given by the following Corollary 5.2.11, which follows immediately from Theorem 5.7 and Lemma 5.9 of [ABT96] and Theorem 5.2.6.

Corollary 5.2.11 For every $j,k \geq 1$,

$$\mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k+}) = \begin{cases} \mathrm{P}^{\mathrm{BH}_{j-1}:\mathrm{BH}_k} & \text{if } j \text{ is odd and } k \text{ is even,} \\ \mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} & \text{otherwise.} \end{cases}$$

5.3 Query Order in the Polynomial Hierarchy

In this section we will study results regarding query order in the polynomial hierarchy. While stating and discussing previously obtained results that essentially establish that query order does not matter in the polynomial hierarchy we observe that the results regarding general query order classes with more than one query in each round miss one interesting case. We ask whether for $j<k$, one round of j parallel queries to a Σ_i^{p} oracle followed by one round of k parallel queries to a Σ_i^{p} oracle is stronger than one round of k parallel queries to a Σ_i^{p} oracle followed by one round of j parallel queries to a Σ_i^{p} oracle. So we study classes of the form $\mathrm{P}_{j,k\text{-tt}}^{\Sigma_i^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}}$. On one hand, $j+2^jk$ different Σ_i^{p} queries can potentially be generated by a $\mathrm{P}_{j,k\text{-tt}}^{\Sigma_i^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}}$ machine. On the other hand, only $j+k$ queries are asked in any run of a $\mathrm{P}_{j,k\text{-tt}}^{\Sigma_i^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}}$ machine. We show that for all $i,j,k \geq 1$, $\mathrm{P}_{j,k\text{-tt}}^{\Sigma_i^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}}$ is as powerful as $j+jk+k$ parallel queries to a Σ_i^{p} oracle, $\mathrm{P}_{j,k\text{-tt}}^{\Sigma_i^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}} = \mathrm{R}^{\mathrm{p}}_{j+jk+k\text{-tt}}(\Sigma_i^{\mathrm{p}})$. It follows that also in the case that all queries are made to oracles from the same level of the polynomial hierarchy the order of query rounds does not matter. However, in contrast to the case that the query rounds are made to oracles from different levels of the polynomial hierarchy, the query rounds can not be made in parallel unless the polynomial hierarchy collapses.

5.3.1 Previous Results–An Overview

Query order in the polynomial hierarchy was first studied by Hemaspaandra, Hemaspaandra, and Hempel [HHH98b]. They studied classes of the form $\mathrm{P}^{\mathcal{C}:\mathcal{D}}$ where $\mathcal{C}$ and $\mathcal{D}$ are classes from the polynomial hierarchy. It was shown that query order never matters in the polynomial hierarchy.

Theorem 5.3.1 [HHH98b]

1. For all $i, \ell \geq 1$,
$$\mathrm{P}^{\Sigma_i^{\mathrm{p}}:\Sigma_\ell^{\mathrm{p}}} = \mathrm{P}^{\Sigma_\ell^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}}.$$

2. For all $i, \ell \geq 1$ such that $i \neq \ell$,
$$\mathrm{P}^{\Sigma_i^{\mathrm{p}}:\Sigma_\ell^{\mathrm{p}}} = \mathrm{P}^{\Sigma_\ell^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}} = \mathrm{P}^{(\Sigma_i^{\mathrm{p}},\Sigma_\ell^{\mathrm{p}})}.$$

Observe that we truly need the $i \neq \ell$ restriction in the second statement of the above theorem. Otherwise we would have included the claim $\mathrm{P}^{\Sigma_i^{\mathrm{p}}[2]} = \mathrm{P}^{\Sigma_i^{\mathrm{p}}}_{2\text{-tt}}$ which due to the closely related structure of the bounded-query, bounded-truth-table and difference hierarchies over Σ_i^{p} would imply a collapse of the polynomial hierarchy to (at least) $\mathrm{DIFF}_3(\Sigma_i^{\mathrm{p}})\mathbf{\Delta}\mathrm{DIFF}_2(\Sigma_{i+1}^{\mathrm{p}})$ (see Corollary 3.3.3 and Theorem 3.3.4). Results similar to those of Theorem 5.3.1 do also hold for the classes Δ_i^{p} and $\Sigma_i^{\mathrm{p}} \cap \Pi_i^{\mathrm{p}}$ from the polynomial hierarchy.

The result of Theorem 5.3.1 has been generalized to the case that more than one query is asked to each oracle by Beigel and Chang [BC].

Theorem 5.3.2 [BC] For all $\ell > i \geq 1$ and all $j, k \geq 1$,
$$\mathrm{P}^{\Sigma_i^{\mathrm{p}}:\Sigma_\ell^{\mathrm{p}}}_{j,k\text{-tt}} = \mathrm{P}^{\Sigma_\ell^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}}_{k,j\text{-tt}} = \mathrm{P}^{(\Sigma_i^{\mathrm{p}},\Sigma_\ell^{\mathrm{p}})}_{j,k\text{-tt}}.$$

Note that no statement is made for $i = \ell$ in the above theorem. We will study and solve the $i = \ell$ case in the next section. Beigel and Chang also studied the effect of query order on the computational power of computing functions. They obtained that in the function setting the order of oracle access for oracles from the polynomial hierarchy is important.

Theorem 5.3.3 [BC]. For all $\ell > i \geq 1$ and all $j, k \geq 1$,

1. $\mathrm{FP}^{\Sigma_i^{\mathrm{p}}:\Sigma_\ell^{\mathrm{p}}}_{j,k\text{-tt}} = \mathrm{FP}^{(\Sigma_i^{\mathrm{p}},\Sigma_\ell^{\mathrm{p}})}_{j,k\text{-tt}} \subseteq \mathrm{FP}^{\Sigma_\ell^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}}_{k,j\text{-tt}}$.

2. $\mathrm{FP}^{\Sigma_i^{\mathrm{p}}:\Sigma_\ell^{\mathrm{p}}}_{j,k\text{-tt}} = \mathrm{FP}^{\Sigma_\ell^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}}_{k,j\text{-tt}} \implies \mathrm{PH} = \Sigma_{i+1}^{\mathrm{p}}$.

Wagner [Wag98] also studied multiple rounds of multiple queries to different oracles from the polynomial hierarchy. He characterized those query order classes in terms of Selivanov's plus hierarchy [Sel94a, Sel95], the generalized boolean hierarchy over $\{\mathrm{NP}, \Sigma_2^{\mathrm{p}}, \Sigma_3^{\mathrm{p}}, \dots\}$.

5.3.2 The Missing Case

Though generalized query order classes of the form $\mathrm{P}_{j,k\text{-tt}}^{\Sigma_i^{\mathrm{p}}:\Sigma_\ell^{\mathrm{p}}}$ have been studied in [BC] all results being obtained assume $i < \ell$. It is not at all clear whether the result of Theorem 5.3.2 also holds when $i = \ell$. As $\mathrm{P}_{j\text{-tt}}^{\Sigma_i^{\mathrm{p}}} = \mathrm{P}^{\mathrm{DIFF}_j(\Sigma_i^{\mathrm{p}})[1]}$ [KSW87] (see also Lemma 5.2.5) it might well be that similarly $\mathrm{P}_{j,k\text{-tt}}^{\Sigma_i^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}} = \mathrm{P}^{\mathrm{DIFF}_j(\Sigma_i^{\mathrm{p}}):\mathrm{DIFF}_k(\Sigma_i^{\mathrm{p}})}$, in which case, due to the results of Section 5.2, the order of query rounds would be crucial. However, we will show in this section that $\mathrm{P}_{j,k\text{-tt}}^{\Sigma_i^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}} = \mathrm{P}^{\mathrm{DIFF}_j(\Sigma_i^{\mathrm{p}}):\mathrm{DIFF}_k(\Sigma_i^{\mathrm{p}})}$ does not hold unless the polynomial hierarchy collapses. This follows from a characterization of $\mathrm{P}_{j,k\text{-tt}}^{\Sigma_i^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}}$ that also establishes that $\mathrm{P}_{j,k\text{-tt}}^{\Sigma_i^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}} = \mathrm{P}_{k,j\text{-tt}}^{\Sigma_i^{\mathrm{p}}:\Sigma_i^{\mathrm{p}}}$ for all $i, j, k \geq 1$.

For clarity of presentation we will from now on restrict ourself to $\Sigma_i^{\mathrm{p}} = \mathrm{NP}$. However, it will be clear from the proofs that similar results hold for Σ_i^{p}, $i \geq 1$.

The main theorem of this section gives a characterization of query order classes $\mathrm{P}_{j,k\text{-tt}}^{\mathrm{NP}:\mathrm{NP}}$ in terms of bounded-truth-table closures of NP. Our approach closely follows the approach of Section 5.2. In particular, we will make use of all concepts and notations defined in Section 5.2. We will use the mind change technique while arguing about paths in hypercubes much in the same way as it was done while proving Theorem 5.2.6. In particular, we will make use of all concepts and notations defined in Section 5.2.

Lemma 5.3.4 For all $j, k \geq 1$, $\mathrm{P}_{j,k\text{-tt}}^{\mathrm{NP}:\mathrm{NP}}$ can handle exactly $j + jk + k$ mind changes.

Proof: Similar to the proof of Lemma 5.2.5 we will first argue that every $\mathrm{P}_{j,k\text{-tt}}^{\mathrm{NP}:\mathrm{NP}}$ machine can have at most $j + jk + k$ mind changes. Second we will give a $\mathrm{P}_{j,k\text{-tt}}^{\mathrm{NP}:\mathrm{NP}}$ machine that on some input has exactly $j + jk + k$ mind changes.

Without loss of generality assume that all queries are made to the same oracle, for instance, a language being many-one complete for NP, and that exactly j queries are made in the first round and exactly k queries are made in the second round. So let $M_{j,k\text{-tt}}^{L:L}$ be a $\mathrm{P}_{j,k\text{-tt}}^{\mathrm{NP}:\mathrm{NP}}$ machine, $L \in \mathrm{NP}$. Observe that on every input x, (potentially) $j + 2^j k$ different NP queries can generated. Let $x \in \Sigma^*$. Denote the queries asked by $M_{j,k\text{-tt}}^{L:L}(x)$ in the first query round by $q_1(x), q_2(x), \ldots, q_j(x)$. For every answer string to this first set of queries, k different second queries are asked in parallel.

In order to enumerate all potential second queries the following concepts will be useful. Let $bin(i)$ denote the binary representation of the integer i. For a string $z \in \Sigma^j$ let $num(z)$ denote the integer such that $z = 0\ldots0bin(num(z))$. Informally, $num(z)$ is the unique integer such that the binary representation of that integer padded with leading zeros equals z. For $z \in \Sigma^j$, we say that $q_1(x), q_2(x), \ldots, q_j(x)$ are answered according to z if and only if for all ℓ, $1 \leq \ell \leq j$, query $q_\ell(x)$, is answered yes if the ℓth bit (counting bits from most significant to least significant) of z is 1 and answered no if the ℓth bit of z is 0.

Now we are prepared to enumerated all (potential) second queries. Let for every $0 \leq i \leq 2^j - 1$, $q_{j+ik+1}(x), q_{j+ik+2}, \ldots, q_{j+ik+k}$ denote the queries asked by $M(x)$ if the queries of the first round have been answered according to z and $num(z) = i$.

Consider the $j + 2^j k$-dimensional hypercube H. Let for every i, $1 \leq i \leq j + 2^j k$, dimension i of the $j+2^j k$-dimensional hypercube H correspond to query $q_i(x)$. Hence every vertex $v \in H$ corresponds to an answer string to all potential queries, when interpreting a 1 (0) in the ith dimension of v as a yes (no) answer to query $q_i(x)$. This allows to label each vertex $v \in H$ with Reject/Accept according to what $M^{L:L}_{j,k\text{-tt}}(x)$ would do when all queries are answered as given by v. Similar to the proof of Lemma 5.2.5 we obtain the labeled hypercube $H_M(x)$.

Claim: *Every ascending path in $H_M(x)$ has at most $j + jk + k$ mind changes.*

To see this let p be an ascending path in $H_M(x)$. Observe that exactly j times we change a 0 to a 1 in the first j dimensions when moving along p. Hence, every vertex on p has one of $j + 1$ different sequences of first j components. Observe that for every vertex $v = (a_1, a_2, \ldots, a_j, \ldots, a_{j+2^j k})$ on p we loose a mind change if the vertex v' following v on p differs from v in a dimension i, $i \notin \{1, 2, \ldots, j\}$ and $i \notin \{j+num(a_1a_2\ldots a_j)k+1, j+num(a_1a_2\ldots a_j)k+2, \ldots, j+num(a_1a_2\ldots a_j)k+k\}$. This follows from the fact that in that case (observe that both vertices correspond to the same set of second k queries as they do not differ in the first j components) the query $q_i(x)$ has no effect on the labeling of the two vertices.

This implies that for every prefix $a_1, a_2, \ldots, a_j$ for a vertex on p at most k changes in the remaining $2^j k$ components can be made without losing a mind change. It follows that p has at most $j + (j + 1)k = j + jk + k$ mind changes.

This shows that every $\mathrm{P}^{\mathrm{NP:NP}}_{j,k\text{-tt}}$ machine can have at most $j+jk+k$ mind changes which implies that the class $\mathrm{P}^{\mathrm{NP:NP}}_{j,k\text{-tt}}$ can handle at most $j + jk + k$ mind changes.

It remains to show that there is a $\mathrm{P}^{\mathrm{NP:NP}}_{j,k\text{-tt}}$ machine that, for some input x, has exactly $j + jk + k$ mind changes. Let M be a $\mathrm{P}^{\mathrm{NP:NP}}_{j,k\text{-tt}}$ machine such that for every pair of different answer strings for the first j queries, the corresponding sets of second k parallel queries are disjoint. In other words, we require M to not ask the same query in the second round for different outcomes of the first j parallel queries.

We distinguish two cases. If k is even, let $M(x)$ accept if and only if the sum of the number of yes answers received in the first and second rounds of queries is odd. If k is odd, let $M(x)$ accept if and only if the sum of yes answers received in the second round of queries is odd. Recall that u_i denotes the ith unit vector. For both cases, observe that the ascending path p starting with

$$(u_{j+1} + \cdots + u_{j+k}) + u_j + (u_{j+k+1} + \cdots + u_{j+2k}) + u_{j-1} + (u_{j+3k+1} + \cdots + u_{j+4k})$$
$$+u_{j-2} + (u_{j+7k+1} + \cdots + u_{j+8k}) + u_{j-3} + \cdots + u_1 + (u_{j+(2^j-1)k+1} + \cdots + u_{j+2^j k})$$

already has $j + jk + k$ mind changes on this initial part. Clearly, by the above fact, the remaining part of the ascending path will not add a single mind change.

This shows that $\mathrm{P}^{\mathrm{NP:NP}}_{j,k\text{-tt}}$ can handle exactly $j + jk + k$ mind changes. ∎

Similar to the proof of Theorem 5.2.6 we will exploit the preceding mind change Lemma to prove the main result of this section.

Theorem 5.3.5 For all $j, k \geq 1$,

$$\mathrm{P}_{j,k\text{-tt}}^{\mathrm{NP:NP}} = \mathrm{R}_{j+jk+k\text{-tt}}^{\mathrm{p}}(\mathrm{NP}).$$

Proof: It follows immediately from Lemma 5.3.4 that $\mathrm{P}_{j,k\text{-tt}}^{\mathrm{NP:NP}} \subseteq \mathrm{R}_{j+jk+k\text{-tt}}^{\mathrm{p}}(\mathrm{NP})$ since $\mathrm{R}_{j+jk+k\text{-tt}}^{\mathrm{p}}(\mathrm{NP})$ is capable of determining the exact number of mind changes of any $\mathrm{P}_{j,k\text{-tt}}^{\mathrm{NP:NP}}$ machine on any input.

To show $\mathrm{R}_{j+jk+k\text{-tt}}^{\mathrm{p}}(\mathrm{NP}) \subseteq \mathrm{P}_{j,k\text{-tt}}^{\mathrm{NP:NP}}$, let $L \in \mathrm{R}_{j+jk+k\text{-tt}}^{\mathrm{p}}(\mathrm{NP})$. Let $M_{j+jk+k\text{-tt}}^{\mathrm{SAT}}$ be a $\mathrm{R}_{j+jk+k\text{-tt}}^{\mathrm{p}}(\mathrm{NP}) = \mathrm{P}_{j+jk+k\text{-tt}}^{\mathrm{NP}}$ machine such that $L = L(M_{j+jk+k\text{-tt}}^{\mathrm{SAT}})$. Observe that $Q = \{\langle x, i\rangle \mid M_{j+jk+k\text{-tt}}^{\mathrm{SAT}}(x)$ has at least i mind changes$\}$ is clearly an NP set. We now describe a $\mathrm{P}_{j,k\text{-tt}}^{\mathrm{NP:NP}}$ machine $\widehat{M}_{j,k\text{-tt}}^{Q:Q}$ that accepts L.

1. On input x compute what $M_{j+jk+k\text{-tt}}^{\mathrm{SAT}}(x)$ would do if all queries to SAT are answered no. Set $a = 1$ if $M_{j+jk+k\text{-tt}}^{\mathrm{SAT}}(x)$ accepts in that case and set $a = 0$ otherwise.

2. Generate the queries $\langle x, k+1\rangle$, $\langle x, 2(k+1)\rangle$, ... , $\langle x, j(k+1)\rangle$ and ask them in parallel to Q. Let $a_1 a_2 \ldots a_j \in \Sigma^j$ be the string corresponding to the received answers, where for all $1 \leq i \leq j$, $a_i = 1$ if and only if query $\langle x, i(k+1)\rangle$ has been answered yes.

3. Observe that we have $a_1 \geq a_2 \geq \cdots \geq a_j$. Let i be the smallest integer such that $a_i = 0$. Note that this means that $M_{j+jk+k\text{-tt}}^{\mathrm{SAT}}(x)$ has at least $(i-1)(k+1)$ mind changes and at most $i(k+1) - 1 = (i-1)(k+1) + k$ mind changes.

4. In order to determine the exact number of mind changes of $M_{j+jk+k\text{-tt}}^{\mathrm{SAT}}(x)$ generate the queries $\langle x, (i-1)(k+1)+1\rangle$, $\langle x, (i-1)(k+1)+2\rangle$, ... , $\langle x, (i-1)(k+1)+k\rangle$ and submit them in parallel to Q. Let $b_1 b_2 \ldots b_k \in \Sigma^k$ be the string corresponding to the received answers, where for all $1 \leq \ell \leq k$, $b_\ell = 1$ if and only if query $\langle x, (i-1)(k+1)+\ell\rangle$ has been answered yes.

5. Observe that we have $b_1 \geq b_2 \geq \cdots \geq b_k$. Let ℓ be the largest integer such that $b_\ell = 1$. Note that this means that $M_{j+jk+k\text{-tt}}^{\mathrm{SAT}}(x)$ has exactly $(i-1)(k+1)+\ell$ mind changes. Accept if and only if $a + (i-1)(k+1) + \ell$ is odd.

It follows from the construction that indeed $L(M_{j+jk+k\text{-tt}}^{\mathrm{SAT}}) = L(\widehat{M}_{j,k\text{-tt}}^{Q:Q})$. This shows that $\mathrm{R}_{j+jk+k\text{-tt}}^{\mathrm{p}}(\mathrm{NP}) \subseteq \mathrm{P}_{j,k\text{-tt}}^{\mathrm{NP:NP}}$ and completes our proof. ∎

It follows immediately from Theorem 5.3.5 that the order of query rounds of different size when all queries are made to the same oracle does not matter. However, in contrast to the case of Theorem 5.3.2 these query rounds can not be made simultaneously unless the polynomial hierarchy collapses.

Corollary 5.3.6 1. For all $j, k \geq 1$, $\mathrm{P}_{j,k\text{-tt}}^{\mathrm{NP:NP}} = \mathrm{P}_{j,k\text{-tt}}^{\mathrm{NP:NP}}$.

2. For all $j, k \geq 1$, $\mathrm{P}_{j,k\text{-tt}}^{\mathrm{NP:NP}} \neq \mathrm{P}_{j,k\text{-tt}}^{(\mathrm{NP,NP})}$ unless the polynomial hierarchy collapses.

Proof: Part one follows directly from Theorem 5.3.5. Regarding part two, note that $\mathrm{P}^{(\mathrm{NP},\mathrm{NP})}_{j,k\text{-tt}} = \mathrm{R}^{\mathrm{p}}_{j+k\text{-tt}}(\mathrm{NP})$. In light of Theorem 5.3.5, we have that $\mathrm{P}^{\mathrm{NP}:\mathrm{NP}}_{j,k\text{-tt}} \neq \mathrm{P}^{(\mathrm{NP},\mathrm{NP})}_{j,k\text{-tt}}$ if and only if $\mathrm{R}^{\mathrm{p}}_{j+jk+k\text{-tt}}(\mathrm{NP}) = \mathrm{R}^{\mathrm{p}}_{j+k\text{-tt}}(\mathrm{NP})$. For $j,k \geq 1$ this implies a collapse of the bounded-truth-table hierarchy at level $j+k$. It is known that a collapse of the bounded-truth-table hierarchy at level $j+k$ implies a collapse of the boolean hierarchy at level $j+k+1$ which in turn implies a collapse of the polynomial hierarchy. This follows from the intertwined structure of the bounded-truth-table hierarchy and the boolean hierarchy and a result by Kadin [Kad88] (see Theorem 3.2.1). In order to specify the collapse of the polynomial hierarchy, we mention that in light of Theorem 3.3.1, we can conclude a collapse of the polynomial hierarchy to $\mathrm{BH}_{j+k+1}\mathbf{\Delta}\mathrm{DIFF}_{j+k}(\Sigma^{\mathrm{p}}_2)$. ∎

Theorem 5.3.5 has an interesting corollary with respect to results obtained in Section 5.2. In Section 5.2 (see Lemma 5.2.1) we have established that for all $m \geq 1$, $\mathrm{R}^{\mathrm{p}}_{m\text{-tt}}(\mathrm{NP}) = \mathrm{R}^{\mathrm{p}}_{1\text{-tt}}(\mathrm{BH}_m)$, or equivalently, $\mathrm{P}^{\mathrm{NP}}_{m\text{-tt}} = \mathrm{P}^{\mathrm{BH}_m}_{1\text{-tt}}$. Inside the proof of Theorem 4.3.5 we mentioned that $\mathrm{P}^{(\Sigma^{\mathrm{p}}_i,\Sigma^{\mathrm{p}}_\ell)}_{j,k\text{-tt}} = \mathrm{P}^{(\mathrm{DIFF}_j(\Sigma^{\mathrm{p}}_i),\mathrm{DIFF}_k(\Sigma^{\mathrm{p}}_\ell))}$. One might be tempted to claim that similarly for $j,k \geq 1$, $\mathrm{P}^{\mathrm{NP}:\mathrm{NP}}_{j,k\text{-tt}} = \mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$. However, this is in general not the case unless the polynomial hierarchy collapses.

Corollary 5.3.7 For all $j,k \geq 1$, if $\mathrm{P}^{\mathrm{NP}:\mathrm{NP}}_{j,k\text{-tt}} = \mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$ then either $j = 1$ or the polynomial hierarchy collapses.

Proof: Suppose $\mathrm{P}^{\mathrm{NP}:\mathrm{NP}}_{j,k\text{-tt}} = \mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k}$. In light of Theorem 5.3.5, we have $\mathrm{P}^{\mathrm{NP}:\mathrm{NP}}_{j,k\text{-tt}} = \mathrm{R}^{\mathrm{p}}_{j+jk+k\text{-tt}}(\mathrm{NP})$. In Section 5.2 (see Theorem 5.2.6) we have established that $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} = \mathrm{R}^{\mathrm{p}}_{j+2k-1\text{-tt}}(\mathrm{NP})$ if j is even and k is odd and $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_k} = \mathrm{R}^{\mathrm{p}}_{j+2k\text{-tt}}(\mathrm{NP})$ otherwise. Suppose j is even and k is odd. Then our assumption implies $\mathrm{R}^{\mathrm{p}}_{j+jk+k\text{-tt}}(\mathrm{NP}) = \mathrm{R}^{\mathrm{p}}_{j+2k-1\text{-tt}}(\mathrm{NP})$. Since $j + jk + k > j + 2k - 1$ for all $j,k \geq 1$, we obtain that the bounded-truth-table hierarchy collapses at level $j + 2k - 1$. As in the proof of Corollary 5.3.6 we conclude a collapse of the polynomial hierarchy to $\mathrm{BH}_{j+2k}\mathbf{\Delta}\mathrm{DIFF}_{j+2k-1}(\Sigma^{\mathrm{p}}_2)$. If j is odd or k is even we have that $\mathrm{R}^{\mathrm{p}}_{j+jk+k\text{-tt}}(\mathrm{NP}) = \mathrm{R}^{\mathrm{p}}_{j+2k\text{-tt}}(\mathrm{NP})$. Hence either $j = 1$ or the bounded-truth-table hierarchy collapses at level $j + 2k$ implying a collapse of the polynomial hierarchy to $\mathrm{BH}_{j+2k+1}\mathbf{\Delta}\mathrm{DIFF}_{j+2k}(\Sigma^{\mathrm{p}}_2)$. ∎

As already mentioned, all results of the current section immediately carry over to Σ^{p}_i, $i \geq 1$. But note that for Σ^{p}_i, $i > 1$, we can, in light of the downward collapse between the levels of the bounded-truth-table hierarchy over Σ^{p}_i and the levels of the boolean hierarchy over Σ^{p}_i (see Theorem 4.2.4), claim a slightly deeper collapse of the polynomial hierarchy than stated in the proofs of Corollary 5.3.6 and Corollary 5.3.7.

Theorem 5.3.8

1. For all $i,j,k \geq 1$, $\mathrm{P}^{\Sigma^{\mathrm{p}}_i:\Sigma^{\mathrm{p}}_i}_{j,k\text{-tt}} = \mathrm{R}^{\mathrm{p}}_{j+jk+k\text{-tt}}(\Sigma^{\mathrm{p}}_i)$.
2. For all $i,j,k \geq 1$, $\mathrm{P}^{\Sigma^{\mathrm{p}}_i:\Sigma^{\mathrm{p}}_i}_{j,k\text{-tt}} = \mathrm{P}^{\Sigma^{\mathrm{p}}_i:\Sigma^{\mathrm{p}}_i}_{k,j\text{-tt}}$.
3. For all $i,j,k \geq 1$, $\mathrm{P}^{\Sigma^{\mathrm{p}}_i:\Sigma^{\mathrm{p}}_i}_{j,k\text{-tt}} \neq \mathrm{P}^{(\Sigma^{\mathrm{p}}_i,\Sigma^{\mathrm{p}}_i)}_{j,k\text{-tt}}$ unless the polynomial hierarchy collapses.

Part 1 of Theorem 5.3.8 can be generalized to more than two rounds of queries. With the same method as in the proof of Theorem 5.3.5 one is able to show that the class of languages being accepted by some DPTM making consecutive rounds of $n_1, n_2, \ldots, n_k$ parallel queries to some Σ_i^{p} oracle, respectively, is exactly the $(n_1+1)(n_2+1)\cdots(n_k+1)-1$-truth-table closure of Σ_i^{p}. This shows that the order of query rounds for more than two rounds does not matter.

5.4 Applications of Query Order and Related Results

Since query order has been studied for the first time in [HHW99] a number of results have been obtained that either directly grew out of the study of query order classes or are related to query order. In this section we will give a short overview over some of these results.

5.4.1 Base Classes Other than P

It has been observed by Hemaspaandra, Hemaspaandra, and Hempel [HHH98b] that most query order results being valid for the base class P do also hold for a large variety of other base classes.

Theorem 5.4.1 [HHH98b] Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be complexity classes and let $\mathcal{D} \in \{\mathrm{R}, \mathrm{coR}, \mathrm{UP}, \mathrm{coUP}, \mathrm{NP}, \mathrm{coNP}, \mathrm{BPP}, \mathrm{PP}, \oplus\mathrm{P}\}$.

$$\mathrm{P}^{\mathcal{C}_1:\mathcal{C}_2} \subseteq \mathrm{P}^{\mathcal{C}_2:\mathcal{C}_1} \implies \mathcal{D}^{\mathcal{C}_1:\mathcal{C}_2} \subseteq \mathcal{D}^{\mathcal{C}_2:\mathcal{C}_1}.$$

The above theorem shows, in light of Corollary 5.2.7, Theorem 5.3.1, and Corollary 5.3.6, that for instance,

1. $\mathrm{UP}^{\mathrm{DP}:\mathrm{BH}_3} = \mathrm{UP}^{\mathrm{BH}_3:\mathrm{DP}} = \mathrm{UP}^{\mathrm{NP}:\mathrm{BH}_3}$.
2. $\mathrm{PP}^{\mathrm{NP}:\Sigma_2^{\mathrm{p}}} = \mathrm{PP}^{\Sigma_2^{\mathrm{p}}:\mathrm{NP}}$.
3. $\oplus\mathrm{P}^{\Sigma_2^{\mathrm{p}}:\Sigma_2^{\mathrm{p}}}_{2,7\text{-tt}} = \oplus\mathrm{P}^{\Sigma_2^{\mathrm{p}}:\Sigma_2^{\mathrm{p}}}_{7,2\text{-tt}}$.

Though Theorem 5.4.1 says that *all* order exchanges of P apply to essentially all standard complexity classes, it of course remains possible that certain path-based classes may possess additional order exchanges. Relatedly, classes may also trivially exhibit certain equalities based on class-specific features. For example, it follows trivially from $\mathrm{NP} \subseteq \mathrm{PP}$ and the (nontrivial) result of Fortnow and Reingold [FR91] regarding the $\leq_{\mathrm{tt}}^{\mathrm{p}}$ closure of PP that $\mathrm{PP} = \mathrm{PP}^{\mathrm{NP}:\mathrm{PP}} = \mathrm{PP}^{\mathrm{PP}:\mathrm{NP}}$.

5.4.2 Results Related to Query Order

The notion of query order, established and studied for the first time in [HHW95, HHW99], has influenced research in complexity theory. Not only that researchers started to study query order in other settings than the boolean hierarchy, the very concept of query order and the resulting new insights into the structure of complexity classes has led to a number of results regarding topics that at first blush might seem totally unrelated, such as bottleneck computation and downward translation of equality. We will in the following list some of the research areas that have been influenced by the research on query order.

Downward Translation of Equality In [HHH98b] Hemaspaandra, Hemaspaandra, and Hempel studied query order in the polynomial hierarchy. It turned out that though query order never matters in the polynomial hierarchy, query order classes seemingly form a refinement of the levels of the polynomial hierarchy. While trying to give evidence that this refinement is strict unless the polynomial hierarchy itself collapses, Hemaspaandra, Hemaspaandra, and Hempel established the first downward collapse result in the polynomial hierarchy (see Theorem 4.2.1). The search for other downward collapses that has been ignited by this result has been studied in detail in Chapter 4.

Bottleneck Computations In a quite different direction, bottleneck machines are a model used to study whether a computational problem can be decomposed into a large number of simple, sequential, tasks, each of which passes on only a very limited amount of information to the next task, and all of which differ only in that input and in a "task number" input [CF91]. A recent paper of Hertrampf [Her97] uses ordered access involving multiple queries, combined with quantifier-based and modulo-based computation, to completely characterize the languages accepted by certain bottleneck machine classes—classes that had long eluded crisp characterization.

Self-Specifying Machines Hemaspaandra, Hempel, and Wechsung [HHW97] have studied self-specifying machines—nondeterministic machines that dynamically specify the path sets on which they will accept. They completely characterize the two most natural such classes in terms of query-order classes with a "positive final query" restriction. They show that the classes have equivalent characterizations as the #P-closures of P and NP, respectively, and they establish a query order result mixing function and language classes: $\mathrm{P}^{\#\mathrm{P}[1]} = \mathrm{P}^{\#\mathrm{P}:\mathrm{NP}} \iff \mathrm{P}^{\#\mathrm{P}[1]} = \mathrm{P}^{\#\mathrm{P}:\mathrm{NP}}_{1,\mathcal{O}(1)\text{-tt}}$ They also show that the classes have characterizations in terms of the "input-specific advice" notation of Köbler and Thierauf [KT94].

Robust Completeness A long line of research has studied the question of whether $\leq_{\mathrm{m}}^{\mathrm{p}}$-completeness and $\leq_{\mathrm{T}}^{\mathrm{p}}$-completeness stand or fall together for classes that potentially lack complete sets. Gurevich [Gur83] and Ambos-Spies [Amb86] have shown that, for all classes $\mathcal{C}$ closed downwards under Turing reductions, it holds robustly that: $\mathcal{C}$ has $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete sets if and only if $\mathcal{C}$ has $\leq_{\mathrm{T}}^{\mathrm{p}}$-complete sets. Nonetheless, by studying a strong nondeterministic closure of NP that, it turns out, exactly equals the query-order class $\mathrm{P}^{\mathrm{NP}\cap\mathrm{coNP}:\mathrm{NP}}$, Hemaspaandra, Hemaspaandra, and Hempel have recently shown that on some reducibility closures of NP, $\leq_{\mathrm{m}}^{\mathrm{p}}$-completeness and $\leq_{\mathrm{T}}^{\mathrm{p}}$-completeness do not robustly stand or fall together [HHH98c].

Bibliography

[ABT96] M. Agrawal, R. Beigel, and T. Thierauf. Pinpointing computations with modular queries in the boolean hierarchy. In *Proceedings of the 13th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 322–334. Springer-Verlag *Lecture Notes in Computer Science #1180*, December 1996.

[All86] E. Allender. The complexity of sparse sets in P. In *Proceedings of the 1st Structure in Complexity Theory Conference*, pages 1–11. Springer-Verlag *Lecture Notes in Computer Science #223*, June 1986.

[Amb86] K. Ambos-Spies. A note on complete problems for complexity classes. *Information Processing Letters*, 23:227–230, 1986.

[BC] R. Beigel and R. Chang. Commutative queries. *Information and Computation*. To appear.

[BCO91] R. Beigel, R. Chang, and M. Ogiwara. A relationship between difference hierarchies and relativized polynomial hierarchies. Technical Report TR 91-1184, Department of Computer Science, Cornell University, Ithaca, NY, January 1991.

[BCO93] R. Beigel, R. Chang, and M. Ogiwara. A relationship between difference hierarchies and relativized polynomial hierarchies. *Mathematical Systems Theory*, 26:293–310, 1993.

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1988. 2nd edition 1995.

[BDG90] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II.* EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1990.

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

[BF96] H. Buhrman and L. Fortnow. Two queries. Technical Report 96-20, University of Chicago, Department of Computer Science, Chicago, IL, September 1996.

[BF98] H. Buhrman and L. Fortnow. Two queries. In *Proceedings of the 13th Annual IEEE Conference on Computational Complexity*, pages 13–19. IEEE Computer Society Press, June 1998.

[BFNA93] L. Babai, L. Fortnow, N.Nisan, and A.Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BJY90] D. Bruschi, D. Joseph, and P. Young. Strong separations for the boolean hierarchy over RP. *International Journal of Foundations of Computer Science*, 1(3):201–218, 1990.

[BM76] J. Bondy and U. Murty. *Graph Theory with Applications*. Macmilliam, 1976.

[Boo74] R. Book. Tally languages and complexity classes. *Information and Control*, 26:186–193, 1974.

[CF91] J. Cai and M. Furst. PSPACE survives constant-width bottlenecks. *International Journal of Foundations of Computer Science*, 2(1):67–76, 1991.

[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, 1988.

[CGH$^+$89] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy II: Applications. *SIAM Journal on Computing*, 18(1):95–111, 1989.

[CH86] J. Cai and L. Hemachandra. The boolean hierarchy: Hardware over NP. In *Proceedings of the 1st Structure in Complexity Theory Conference*, pages 105–124. Springer-Verlag *Lecture Notes in Computer Science #223*, June 1986.

[Cha] R. Chang. Bounded queries, approximations, and the boolean hierarchy. *Information and Computation*. To appear.

[Cha91] R. Chang. *On the Structure of NP Computations under Boolean Operators*. PhD thesis, Cornell University, Ithaca, NY, 1991.

[CK89] R. Chang and J. Kadin. The boolean hierarchy and the polynomial hierarchy: A closer connection. Technical Report TR 89-1008, Department of Computer Science, Cornell University, Ithaca, NY, May 1989.

[CK96] R. Chang and J. Kadin. The boolean hierarchy and the polynomial hierarchy: A closer connection. *SIAM Journal on Computing*, 25(2):340–354, 1996.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, pages 151–158, 1971.

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

[Eul36] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Comment. Academiae Sci. I. Petropolitanae*, 8:128–140, 1736.

[FR91] L. Fortnow and N. Reingold. PP is closed under truth-table reductions. In *Proceedings of the 6th Structure in Complexity Theory Conference*, pages 13–15. IEEE Computer Society Press, June/July 1991.

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

[GNW90] T. Gundermann, N. Nasser, and G. Wechsung. A survey on counting classes. In *Proceedings of the 5th Structure in Complexity Theory Conference*, pages 140–153. IEEE Computer Society Press, July 1990.

[Gur83] Y. Gurevich. Algebras of feasible functions. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 210–214. IEEE Computer Society Press, November 1983.

[Hau14] F. Hausdorff. *Grundzüge der Mengenlehre.* Leipzig, 1914.

[Hem89] L. Hemachandra. The strong exponential hierarchy collapses. *Journal of Computer and System Sciences*, 39(3):299–322, 1989.

[Her97] U. Hertrampf. Acceptance by transformation monoids (with an application to local-self-reductions). In *Proceedings of the 12th Annual IEEE Conference on Computational Complexity*, pages 213–224. IEEE Computer Society Press, June 1997.

[HHH96a] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. An upward separation in the polynomial hierarchy. Technical Report Math/Inf/96/15, Institut für Informatik, Friedrich-Schiller-Universität Jena, Jena, Germany, June 1996.

[HHH96b] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. A downward translation in the polynomial hierarchy. Technical Report TR-630, Department of Computer Science, University of Rochester, Rochester, NY, July 1996.

[HHH97a] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. An introduction to query order. *Bulletin of the EATCS*, 63:93–107, 1997.

[HHH97b] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. Translating equality downwards. Technical Report TR-657, Department of Computer Science, University of Rochester, Rochester, NY, April 1997.

[HHH98a] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. Downward collapse from a weaker hypothesis. In *Proceedings of the 6th Italian Conference on Theoretical Computer Science*. World Scientific, November 1998.

[HHH98b] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. Query order in the polynomial hierarchy. *Journal of Universal Computer Science*, 4(6):574–588, June 1998.

[HHH98c] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. $\mathrm{R}^{\mathcal{SN}}_{1\text{-}tt}(\mathrm{NP})$ distinguishes robust many-one and Turing completeness. *Theory of Computing Systems*, 31:307–325, 1998.

[HHH98d] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. What's up with downward collapse: Using the easy-hard technique to link boolean and polynomial hierarchy collapses. Technical Report TR-682, Department of Computer Science, University of Rochester, Rochester, NY, February 1998.

[HHH98e] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. What's up with downward collapse: Using the easy-hard technique to link boolean and polynomial hierarchy collapses. *SIGACT News*, 29(3):10–22, 1998.

[HHH99] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. A downward collapse within the polynomial hierarchy. *SIAM Journal on Computing*, 28(2):383–393, 1999.

[HHW95] L. Hemaspaandra, H. Hempel, and G. Wechsung. Query order and self-specifying machines. Technical Report TR-596, Department of Computer Science, University of Rochester, Rochester, NY, October 1995.

[HHW97] L. Hemaspaandra, H. Hempel, and G. Wechsung. Self-specifying machines. Technical Report TR-654, Department of Computer Science, University of Rochester, Rochester, NY, April 1997.

[HHW99] L. Hemaspaandra, H. Hempel, and G. Wechsung. Query order. *SIAM Journal on Computing*, 28(2):637–651, 1999.

[HIS85] J. Hartmanis, N. Immerman, and V. Sewelson. Sparse sets in NP−P: EXPTIME versus NEXPTIME. *Information and Control*, 65(2/3):159–181, 1985.

[HJ95] L. Hemaspaandra and S. Jha. Defying upward and downward separation. *Information and Computation*, 121(1):1–13, 1995.

[HKR93] S. Homer, S. Kurtz, and J. Royer. On 1-truth-table-hard languages. *Theoretical Computer Science*, 115(2):383–389, 1993.

[HR97] L. Hemaspaandra and J. Rothe. Unambiguous computation: Boolean hierarchies and sparse turing-complete sets. *SIAM Journal on Computing*, 26(3):634–653, 1997.

[HRW97] L. Hemaspaandra, J. Rothe, and G. Wechsung. Easy sets and hard certificate schemes. *Acta Informatica*, 34:859–879, 1997.

[HS66] F. Hennie and R. Stearns. Two-way simulation of multi tape turing machines. *Journal of the ACM*, 13:533–546, 1966.

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

[HW97a] E. Hemaspaandra and G. Wechsung. The minimization problem for boolean formulas. In *Proceedings of the 38rd IEEE Symposium on Foundations of Computer Science*, pages 575–584. IEEE Computer Society Press, October 1997.

[HW97b] H. Hempel and G. Wechsung. On the power of #P reductions. Technical Report Math/Inf/97/2, Institut für Informatik, Friedrich-Schiller-Universität Jena, Jena, Germany, January 1997.

[HW97c] H. Hempel and G. Wechsung. The operators min and max on the polynomial time hierarchy. In *Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science*, pages 93–104. Springer-Verlag *Lecture Notes in Computer Science #1200*, February 1997.

[HY84] J. Hartmanis and Y. Yesha. Computation times of NP sets of different densities. *Theoretical Computer Science*, 34:17–32, 1984.

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

[Kad87] J. Kadin. The polynomial time hierarchy collapses if the boolean hierarchy collapses. Technical Report TR 87-843, Cornell University, 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, 1988. Erratum appears in the same journal, 20(2):404.

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

[KL80] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th ACM Symposium on Theory of Computing*, pages 302–309, April 1980. An extended version has also appeared as: Turing machines that take advice, *L'Enseignement Mathématique*, 2nd series 28, 1982, pages 191–209.

[Ko82] K. Ko. Some observations on the probabilistic algorithms and NP-hard problems. *Information Processing Letters*, 14(1):39–43, 1982.

[Ko88] K. Ko. Relativized polynomial time hierarchies having exactly k levels. In *Proceedings of the 20th ACM Symposium on Theory of Computing.* ACM Press, May 1988.

[KSW87] J. Köbler, U. Schöning, and K. Wagner. The difference and truth-table hierarchies for NP. *R.A.I.R.O. Informatique théorique et Applications*, 21:419–435, 1987.

[KT94] J. Köbler and T. Thierauf. Complexity-restricted advice functions. *SIAM Journal on Computing*, 23(2):261–275, 1994.

[LJK89] K.-J. Lange, B. Jenner, and B. Kirsig. The logarithmic alternation hierarchy collapses: $A\Sigma_2^L = A\Pi_2^L$. *Information and Computation*, 1989.

[LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–124, 1975.

[MS72] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th IEEE Symposium on Switching and Automata Theory*, pages 125–129, 1972.

[MS73] A. Meyer and L. Stockmeyer. Word problems requiring exponential time. In *Proceedings of the 5th ACM Symposium on Theory of Computing*, pages 1–9, 1973.

[Pap94] C. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[Pos36] E. Post. Finite combinatory process. *Journal of Symbolic Logic*, 1:103–105, 1936.

[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, 1984.

[RRW94] R. Rao, J. Rothe, and O. Watanabe. Upward separation for FewP and related classes. *Information Processing Letters*, 52(4):175–180, 1994.

[RW98] S. Reith and K. Wagner. On boolean lowness and boolean highness. In *Proceedings of the 4th Annual International Computing & Combinatorics Conference*, 1998. To appear.

[Sel94a] V.L. Selivanov. Two refinements of the polynomial hierarchy. In *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science*, pages 439–448. Springer-Verlag *Lecture Notes in Computer Science #775*, February 1994.

[Sel94b] A. Selman. A taxonomy of complexity classes of functions. *Journal of Computer and System Sciences*, 48(2):357–381, 1994.

[Sel95] V.L. Selivanov. Fine hierarchies and boolean terms. *Journal of Symbolic Logic*, 60(1):289–317, 1995.

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

[SW88] U. Schöning and K. Wagner. Collapsing oracle hierarchies, census functions, and logarithmically many queries. In *Proceedings of the 5th Annual Symposium on Theoretical Aspects of Computer Science*. Springer-Verlag *Lecture Notes in Computer Science*, February 1988.

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

[Tod87] S. Toda. Σ_2SPACE[n] is closed under complement. *Journal of Computer and System Sciences*, 35:145–152, 1987.

[Tur36] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, ser. 2*, 42:230–265, 1936. Correction, *ibid*, vol. 43, pp. 544–546, 1937.

[Val76] L. Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, 5:20–23, 1976.

[VW93] H. Vollmer and K. Wagner. The complexity of finding middle elements. *International Journal of Foundations of Computer Science*, 4:293–307, 1993.

[Wag79] K. Wagner. On ω regular sets. *Information and Control*, 43:123–177, 1979.

[Wag87] K. Wagner. Number-of-query hierarchies. Technical Report 158, Institut für Mathematik, Universität Augsburg, Augsburg, Germany, October 1987.

[Wag89] K. Wagner. Number-of-query hierarchies. Technical Report 4, Institut für Informatik, Universität Würzburg, Würzburg, Germany, February 1989.

[Wag90] K. Wagner. Bounded query classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.

[Wag98] K. Wagner. A note on parallel queries and the symmetric-difference hierarchy. *Information Processing Letters*, 66(1):13–20, 1998.

[Wec85] G. Wechsung. On the boolean closure of NP. In *Proceedings of the 5th Conference on Fundamentals of Computation Theory*, pages 485–493. Springer-Verlag *Lecture Notes in Computer Science #199* , 1985. (An unpublished precursor of this paper was coauthored by K. Wagner).

[Wra77] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3:23–33, 1977.

[WW86] K. Wagner and G. Wechsung. *Computational Complexity.* D. Reidel Publishing Company, 1986. Distributors for the U.S.A. and Canada: Kluwer Academic Publishers.

[Yao85] A. Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 1–10, 1985.

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

List of Symbols

Subject Index