What’s Up with Downward Collapse: Using the Easy-Hard Technique to Link Boolean and Polynomial Hierarchy Collapses

Edith Hemaspaandra†
Department of Mathematics
Le Moyne College
Syracuse, NY 13214
USA

Lane A. Hemaspaandra‡
Department of Computer Science
University of Rochester
Rochester, NY 14627
USA

Harald Hempel§
Institut für Informatik
Friedrich-Schiller-Universität Jena
07740 Jena
Germany

Technical Report UR-CS-TR-98-682
February, 1998

Abstract: During the past decade, nine papers have obtained increasingly strong consequences from the assumption that boolean or bounded-query hierarchies collapse. The final four papers of this nine-paper progression actually achieve downward collapse—that is, they show that high-level collapses induce collapses at (what beforehand were thought to be) lower complexity levels. For example, for each \( k \geq 2 \) it is now known that if \( \text{P}^{\Sigma_p^k[1]} = \text{P}^{\Sigma_p^k[2]} \) then \( \text{PH} = \Sigma_p^k \). This article surveys the history, the results, and the technique—the so-called easy-hard method—of these nine 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, Universität Augsburg, Institut für Mathematik, Augsburg, Germany, October 1987.

3. K. Wagner. Number-of-query hierarchies. Technical Report 4, Universität Würzburg, Institut für Informatik, 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.

*Supported in part by grants NSF-CCR-9322513 and NSF-INT-9513368/DAAD-315-PRO-fo-ab.
†edith@bamboo.lemoyne.edu. Work done in part while visiting Friedrich-Schiller-Universität Jena.
‡lane@cs.rochester.edu. Work done in part while visiting Friedrich-Schiller-Universität Jena.
§hempel@informatik.uni-jena.de. Work done in part while visiting Le Moyne College.
1 Introduction

Does the polynomial hierarchy collapse if the boolean hierarchy or the bounded-query hierarchy collapse? Kadin [Kad88] was able to answer this question affirmatively with the help of the easy-hard technique. Until 1995 this technique (slightly modified) has been used five times to obtain stronger and stronger collapses of the polynomial hierarchy from the assumption that the boolean hierarchy (also called difference hierarchy) or the bounded-query hierarchy collapse. All those results have in common that a collapse of the boolean or bounded-query hierarchy induces a collapse of the polynomial hierarchy at a higher level. In 1996 Hemaspaandra, Hemaspaandra, and Hempel [HHH] and also Buhrman and Fortnow [BF96] obtained so called downward collapse results within the polynomial hierarchy, the easy-hard technique playing a crucial role in the proofs. The term downward collapse refers to the fact that the collapse of larger classes implies the collapse of smaller classes, the collapse translates downwards. It seems that the time is right to take a close look at the easy-hard method, especially its applications for collapsing the polynomial hierarchy.

This survey will be structured as follows. The timeline of the nine papers using the easy-hard method is given in Section 2. Section 3 lists the key results of these nine papers. In Section 4 we prove a new and stronger result regarding to what exact level the polynomial hierarchy collapses if the boolean hierarchy over \( \Sigma^p_k \) collapses. Section 5 gives an overview of the history of the easy-hard technique. In particular, we will first informally discuss the contributions of the various papers to the evolution of the easy-hard technique. Second, will rigorously prove a special case of the main theorem of each of the papers and so illustrate the development of the easy-hard technique to yield stronger and stronger results. Finally, Section 6 suggests interesting open issues related to the topic of this article.

2 The Timeline

Table 1 gives the relevant dates and citations for all nine papers, in particular the dates of the earliest versions and pointers to the earliest and most recent versions.
### Table 1: Timeline.

| Author(s) | Date of Earliest Version | Earliest Version | Most Recent Version |
|-----------|--------------------------|------------------|---------------------|
| Kadin     | 6/87                     | Kad87            | Kad88               |
| Wagner    | 10/87                    | Wag87            | Wag87               |
| Wagner    | 2/89                     | Wag89            | Wag89               |
| Chang/Kadin | 5/89                   | CK89             | CK96                |
| Beigel/Chang/Ogiwara | 1/91               | BCO91            | BCO93               |
| Hemaspaandra/Hemaspaandra/Hempel | 6/96           | HHH96b           | HHH96b              |
| Hemaspaandra/Hemaspaandra/Hempel | 7/96         | HHH96a           | HHH                 |
| Buhrman/Fortnow   | 9/96               | BF96             | BF98                |
| Hemaspaandra/Hemaspaandra/Hempel | 4/97          | HHH97            | HHH97               |

### 3 The Results

The pace and amount of change undergone by the easy-hard technique, in particular how it has been used to collapse the polynomial hierarchy, can best be seen by a close look at the key results of the relevant papers. In this section we will list their main theorems (or most charismatic results). This will also nicely illustrate the improvements each made with respect to the results that were known before it.

A definition of the basic concepts involved seems appropriate at this point. The polynomial hierarchy was introduced by Stockmeyer [Sto77].

**Definition 3.1** 1. For any set of languages \( C \), let \( \text{co}C = \{ \overline{T} | L \in C \} \).

2. **[Sto77]**
   (a) \( \Delta^p_0 = \Sigma^p_0 = \Pi^p_0 = P \).
   (b) For all \( k \geq 0 \), \( \Delta^p_{k+1} = \text{P}^{\Sigma^p_k}, \Sigma^p_{k+1} = \text{NP}^{\Sigma^p_k}, \) and \( \Pi^p_{k+1} = \text{co} \Sigma^p_{k+1} \).
   (c) The polynomial hierarchy, \( \text{PH} \), is defined by \( \text{PH} = \bigcup_{k \geq 0} \Sigma^p_k \).

So, for instance, \( \Sigma^p_1 = \text{NP}, \Sigma^p_2 = \text{NP}^{\text{NP}}, \) and \( \Sigma^p_3 = \text{NP}^{\text{NP}^{\text{NP}}} \). The boolean or difference hierarchy is a concept used to study the structure within the boolean closure of a class \( C \) (the closure of \( C \) with respect to the boolean operations \( \land, \lor, \) and negation). It has particularly often been studied in terms of boolean hierarchies built on the classes \( \Sigma^p_k, k \geq 1 \).

**Definition 3.2** 1. For sets of languages \( C_1 \) and \( C_2 \), let \( C_1 \ominus C_2 = \{ L_1 - L_2 | L_1 \in C_1 \land L_2 \in C_2 \} \).

2. **[CGH+88]** For all \( k \geq 1 \),
   (a) \( \text{DIFF}_1(\Sigma^p_k) = \Sigma^p_k \).
   (b) For all \( m \geq 1 \), \( \text{DIFF}_{m+1}(\Sigma^p_k) = \Sigma^p_k \oplus \text{DIFF}_m(\Sigma^p_k) \).
   (c) The boolean or difference hierarchy over \( \Sigma^p_k \), \( \text{BH}(\Sigma^p_k) \), is defined as \( \text{BH}(\Sigma^p_k) = \bigcup_{m \geq 1} \text{DIFF}_m(\Sigma^p_k) \).
For instance, \( \text{DIFF}_2(\text{NP}) \) is exactly the class \( \text{DP} \) \([PY84]\). Similarly (and for reasons to be explained more formally later), we will for every \( k \geq 1 \) denote \( \text{DIFF}_k(\Sigma^p_k) \) by \( \text{DSigma}_k^p \). It is a well-known fact that, for every \( k \geq 1 \), it holds that \( \text{BH}(\Sigma^p_k) \), the difference hierarchy over \( \Sigma^p_k \), is sandwiched between \( \Sigma^p_k \cup \Pi^p_k \) and \( \Delta^p_{k+1} \).

Restricting the type of access to an oracle one has leads to the notions of bounded-Turing and bounded-truth-table query classes (see, e.g., Ladner, Lynch, and Selman \([LLS75]\)).

**Definition 3.3**  Let \( k \geq 0 \) be an integer.

1. For \( m \geq 1 \), \( P^{\Sigma^p_k[m]} \) denotes the set of languages recognizable by some deterministic polynomial-time Turing machine (DPTM) making at most \( m \) queries to a \( \Sigma^p_k \) oracle.
2. For \( m \geq 1 \), \( P^{\Sigma^p_k(m)} \) denotes the set of languages recognizable by some DPTM making \textit{in parallel} (at once, without knowing the answer of any query) at most \( m \) queries to a \( \Sigma^p_k \) oracle.
3. The bounded-query hierarchy and the bounded-truth-table hierarchy over \( \Sigma^p_k \) are formed by the classes \( P^{\Sigma^p_k[m]} \) and \( P^{\Sigma^p_k(m)} \), \( m \geq 1 \), respectively.

Obviously for all \( k \geq 1 \), \( P^{\Sigma^p_k(m)} \subseteq P^{\Sigma^p_k[m]} \) for all \( m \geq 1 \). Also, \( P^{\Sigma^p_k(m)} = P^{\Sigma^p_k[1]} \).

It is well-known \([KSW87]\) that the bounded-truth-table and the boolean hierarchy intertwine, i.e., for all \( m \geq 1 \) and all \( k \geq 1 \),

\[
\text{DIFF}_m(\Sigma^p_k) \cup \text{coDIFF}_m(\Sigma^p_k) \subseteq P^{\Sigma^p_k(m)} \subseteq \text{DIFF}_m(\Sigma^p_k) \cap \text{coDIFF}_m(\Sigma^p_k).
\]

Hence a result \( \text{DIFF}_m(\Sigma^p_k) = \text{coDIFF}_m(\Sigma^p_k) \iff \text{PH} = \text{C} \) yields as an easy corollary \( F^{\Sigma^p_k(m)}_{m-tt} = P^{\Sigma^p_k(m+1)-tt} \Rightarrow \text{PH} = \text{C} \). We are now prepared to turn to the results obtained in the papers under consideration.

1) \textbf{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 over \( \Sigma^p_k \) at level \( m \) implies a collapse of the polynomial hierarchy to its \((k + 2)\)nd level, \( \Sigma^p_{k+2} \).

**Theorem 3.4**  \([Kad87,Kad88]\)  For all \( m \geq 1 \) and all \( k \geq 1 \), if \( \text{DIFF}_m(\Sigma^p_k) = \text{coDIFF}_m(\Sigma^p_k) \) then \( \text{PH} = \Sigma^p_{k+2} \).

**Corollary 3.5**  \([Kad87,Kad88]\)  For all \( m \geq 1 \) and all \( k \geq 1 \), if \( P^{\Sigma^p_k(m)} = P^{\Sigma^p_k(m+1)-tt} \) then \( \text{PH} = \Sigma^p_{k+2} \).

2) \textbf{Wagner 1987} \([Wag87]\)  Kadin’s technique together with oracle replacement enabled Wagner to improve Kadin’s results significantly by showing that a collapse of the boolean hierarchy over \( \Sigma^p_k \) at level \( m \) implies a collapse of the polynomial hierarchy to \( \Delta^p_{k+2} \).

**Theorem 3.6**  \([Wag87]\)  For all \( m \geq 1 \) and all \( k \geq 1 \), if \( \text{DIFF}_m(\Sigma^p_k) = \text{coDIFF}_m(\Sigma^p_k) \) then \( \text{PH} = \Delta^p_{k+2} \).

**Corollary 3.7**  \([Wag87]\)  For all \( m \geq 1 \) and all \( k \geq 1 \), if \( P^{\Sigma^p_k} = P^{\Sigma^p_k(m+1)-tt} \) then \( \text{PH} = \Delta^p_{k+2} \).
For all $m \geq 1$ and all $k \geq 1$, if $\text{DIFF}_m(\Sigma_k^p) = \text{coDIFF}_m(\Sigma_k^p)$ then $\text{PH} = \text{BH}(\Sigma_{k+1}^p)$.

Corollary 3.9  
For all $m \geq 1$ and all $k \geq 1$, if $P_{m \text{-} \text{tt}}^{\Sigma_k^p} = P_{(m+1) \text{-} \text{tt}}^{\Sigma_k^p}$ then $\text{PH} = \text{BH}(\Sigma_{k+1}^p)$.

4) Chang/Kadin 1989  
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 over $\Sigma_k^p$ at level $m$ implies a collapse of the polynomial hierarchy to a level within the boolean closure of $\Sigma_{k+1}^p$, namely, the $m$th level of the boolean hierarchy over $\Sigma_{k+1}^p$.

Theorem 3.10  
For all $m \geq 1$ and all $k \geq 1$, if $\text{DIFF}_m(\Sigma_k^p) = \text{coDIFF}_m(\Sigma_k^p)$ then $\text{PH} = \text{DIFF}_{m+1}(\Sigma_{k+1}^p)$.

Corollary 3.11  
For all $m \geq 1$ and all $k \geq 1$, if $P_{m \text{-} \text{tt}}^{\Sigma_k^p} = P_{(m+1) \text{-} \text{tt}}^{\Sigma_k^p}$ then $\text{PH} = \text{DIFF}_{m+1}(\Sigma_{k+1}^p)$.

5) Beigel/Chang/Ogiwara 1991  
Beigel, Chang, and Ogiwara, while picking up ideas developed by Wagner, were able to draw a stronger conclusion. In particular, they showed that a collapse of the boolean hierarchy over $\Sigma_k^p$ at level $m$ implies a collapse of the polynomial hierarchy to a level within the $m$th level of the boolean hierarchy over $\Sigma_{k+1}^p$, namely, to $\left( P^{\text{NP}}_{(m-1) \text{-} \text{tt}} \right)^{\Sigma_k^p}$, the class of languages that can be accepted by some deterministic polynomial-time machine making at most $m-1$ parallel queries to an $\text{NP}^{\Sigma_k^p} = \Sigma_{k+1}^p$ oracle and an unlimited number of queries to a $\Sigma_k^p$ oracle.

Theorem 3.12  
For all $m \geq 1$ and all $k \geq 1$, if $\text{DIFF}_m(\Sigma_k^p) = \text{coDIFF}_m(\Sigma_k^p)$ then $\text{PH} = \left( P^{\text{NP}}_{(m-1) \text{-} \text{tt}} \right)^{\Sigma_k^p}$.

Corollary 3.13  
For all $m \geq 1$ and all $k \geq 1$, if $P_{m \text{-} \text{tt}}^{\Sigma_k^p} = P_{(m+1) \text{-} \text{tt}}^{\Sigma_k^p}$ then $\text{PH} = \left( P^{\text{NP}}_{m \text{-} \text{tt}} \right)^{\Sigma_k^p}$.

6) Hemaspaandra/Hemaspaandra/Hempel 1996  
Motivated by the question of whether the collapse of query order classes has some effect on the polynomial hierarchy, Hemaspaandra, Hemaspaandra, and Hempel came up with a very surprising downward collapse result. A collapse of the bounded-query hierarchy over $\Sigma_k^p$, $k > 2$, at its first level implies a collapse of the polynomial hierarchy to $\Sigma_k^p$ 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 translation of equality” (equivalently, “downward collapse) result ever obtained within the bounded query hierarchies.

Theorem 3.14  
For $k > 2$, if $P^{\Sigma_k^p}[1] = P^{\Sigma_k^p}[2]$ then $\text{PH} = \Sigma_k^p = \Pi_k^p$.  


7) Hemaspaandra/Hemaspaandra/Hempel 1996 [HHH96a,HHH] Generalizing the ideas developed in [HHH96a,HHH], the authors extended their results to also hold for $j$-vs-$(j+1)$ queries. More precisely, a collapse of the bounded-truth-table hierarchy over $\Sigma^p_k$ at level $m$ implies the collapse of the boolean hierarchy over $\Sigma^p_k$ at level $m$. This again is a downward collapse result as clearly $\text{DIFF}_m(\Sigma^p_k) \cup \text{coDIFF}_m(\Sigma^p_k) \subseteq P^{\Sigma^p_m}_{(m+1)-tt}$, and moreover the inclusion is believed to be strict.

**Theorem 3.15** [HHH96a,HHH] For all $m \geq 1$ and all $k > 2$, if $P^{\Sigma^p_k}_{m-tt} = P^{\Sigma^p_k}_{(m+1)-tt}$ then $\text{DIFF}_m(\Sigma^p_k) = \text{coDIFF}_m(\Sigma^p_k)$.

This, together with the upcoming Theorem 3.16, yields also a strong collapse of the polynomial hierarchy.

**Corollary 3.16** For all $m \geq 1$ and all $k > 2$, if $P^{\Sigma^p_k}_{m-tt} = P^{\Sigma^p_k}_{(m+1)-tt}$ then $\text{PH} = \text{DIFF}_m(\Sigma^p_k) \Delta \text{DIFF}_{m-1}(\Sigma^p_{k+1})$.

Note that the collapse of the polynomial hierarchy occurs, roughly speaking, one level lower in the boolean hierarchy over $\Sigma^p_{k+1}$ than could be concluded from the same hypothesis without Theorem 3.15.

8) Buhrman/Fortnow 1996 [BF96,BF98] Buhrman and Fortnow extended Theorem 3.14 to the $k=2$ case; they proved that $P^{\Sigma^p_2}_1 = P^{\Sigma^p_2}_2$ implies a collapse of the polynomial hierarchy to $\Sigma^p_2$, establishing a downward collapse in the second level of the polynomial hierarchy.

**Theorem 3.17** [BF96] If $P^{\Sigma^p_2}_1 = P^{\Sigma^p_2}_2$ then $\text{PH} = \Sigma^p_2 = \Pi^p_2$.

9) Hemaspaandra/Hemaspaandra/Hempel 1997 [HHH97] In [HHH97] the approaches of [HHH96a,HHH] and [BF96] were combined with new ideas to obtain a result that implies Theorems 3.14, 3.15, 3.17, and more.

**Theorem 3.18** [HHH97] For all $m \geq 1$ and all $k > 1$, if $P^{\Sigma^p_k}_{m-tt} = P^{\Sigma^p_k}_{(m+1)-tt}$ then $\text{DIFF}_m(\Sigma^p_k) = \text{coDIFF}_m(\Sigma^p_k)$.

This is a very general downward collapse result, as the $m$th level of the boolean hierarchy over $\Sigma^p_k$ is contained in $P^{\Sigma^p_m}_{m-tt}$. In light of Theorem 4.1 the above Theorem 3.18 also gives a collapse of the polynomial hierarchy that was previously unknown to hold.

**Corollary 3.19** For all $m \geq 1$ and all $k > 1$, if $P^{\Sigma^p_k}_{m-tt} = P^{\Sigma^p_k}_{(m+1)-tt}$ then $\text{PH} = \text{DIFF}_m(\Sigma^p_k) \Delta \text{DIFF}_{m-1}(\Sigma^p_{k+1})$.

4 Improving the Collapse of the Polynomial Hierarchy Under the Hypothesis that the Boolean Hierarchy Over $\Sigma^p_k$ Collapses

The first five papers of this survey obtained deeper and deeper collapses of the polynomial hierarchy if the boolean hierarchy over $\Sigma^p_k$ collapses. The strongest result previously known is due to Beigel, Chang, and Ogihara [BCO91,BCO93], see Theorem 3.12. Theorem 3.12 says that, given a collapse
of the boolean hierarchy over $\Sigma^p_k$ at level $m$, the polynomial hierarchy collapses to $\left( \text{P}^{\text{NP}^m}_{(m-1)\text{-tt}} \right)^{\Sigma^p_k}$, a class contained in DIFF$_m(\Sigma^p_k+1)$.

Define for complexity classes $C$ and $D$, $C \Delta D = \{ C \Delta D \mid C \in C \wedge D \in D \}$, where $C \Delta D$ denotes the symmetric difference of the sets $C$ and $D$. A careful analysis of the proof of Theorem 3.12 as given in BCO93 in combination with a new trick, namely applying an idea developed in BCO91, BCO93 twice, yields the following theorem. Theorem 4.1 has been independently obtained by Reith and Wagner [RW98].

**Theorem 4.1** For all $m \geq 1$ and all $k \geq 1$, if DIFF$_m(\Sigma^p_k) = \text{coDIFF}_m(\Sigma^p_k)$ then PH = DIFF$_m(\Sigma^p_k) \Delta \text{DIFF}_{m-1}(\Sigma^p_k+1)$.

Let us compare the results of Theorem 3.12 and Theorem 4.1. Though both theorems collapse the polynomial hierarchy to a class containing DIFF$_{m-1}(\Sigma^p_k+1)$ and contained in DIFF$_m(\Sigma^p_k+1)$, their results differ substantially. It is immediate from a recent paper of Wagner [Wag] that $\left( \text{P}^{\text{NP}^m}_{(m-1)\text{-tt}} \right)^{\Sigma^p_k}$ is a strict superset of DIFF$_m(\Sigma^p_k) \Delta \text{DIFF}_{m-1}(\Sigma^p_k+1)$ unless the polynomial hierarchy collapses.

Furthermore, observe that $\left( \text{P}^{\text{NP}^m}_{(m-1)\text{-tt}} \right)^{\Sigma^p_k}$ involves $m - 1$ parallel queries to a $\Sigma^p_k$ oracle and an unlimited number of queries to a $\Sigma^p_k$ oracle. So the P base machine of $\left( \text{P}^{\text{NP}^m}_{(m-1)\text{-tt}} \right)^{\Sigma^p_k}$ evaluates $m - 1$ bits of information originating from the parallel $\Sigma^p_{k+1}$ queries and polynomially many bits of information from the $\Sigma^p_k$ queries. In contrast, DIFF$_m(\Sigma^p_k) \Delta \text{DIFF}_{m-1}(\Sigma^p_k+1)$ 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 DIFF$_{m-1}(\Sigma^p_{k+1})$ part consisting of $m - 1$ underlying parallel queries to a $\Sigma^p_{k+1}$ oracle evaluated with one fixed truth-table. The second bit of information, the one from the DIFF$_m(\Sigma^p_k)$ part, implicitly contains $m$ parallel queries to a $\Sigma^p_k$ oracle which again are evaluated via a fixed truth-table. In a nutshell, we have improved from unlimited many queries to $\Sigma^p_k$ and $(m - 1)$-truth-table queries to $\Sigma^p_{k+1}$, to $m$-fixed-truth-table queries to $\Sigma^p_k$ and $(m - 1)$-fixed-truth-table queries to $\Sigma^p_{k+1}$.

Before we are prove Theorem 4.1, let us agree on the following convention: Whenever we talk about polynomials let us assume that those polynomials are of the form $n^a + b$ for some integers $a, b > 0$. Since all complexity classes under consideration are closed under many-one reductions and the polynomials involved in the upcoming proofs (in this section as well as in Section 3) always play the role of a function bounding the running time of some Turing machine or the length of some variable, 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 our proofs. Readers interested in the general flavor of the proof of Theorem 4.1 are encouraged to read the proof assuming $k = 3$ and $m = 2$.

**Proof of Theorem 4.1** The proof is structured in a way that the reader will easily find analogies to the proofs of the special cases in Subsection 5.2.

Observe that the claim is immediate for $m = 1$. So suppose $m \geq 2$.

**A** Let $\Sigma = \{0, 1\}$ and let $\# \not\in \Sigma$. Let $\langle \cdot \rangle$ be a pairing function that maps sequences of length at most $m + 1$ of strings over $\Sigma^* \cup \{\#\}$ to $\Sigma^*$ having the standard properties such as polynomial-time computability and invertibility etc. Let $s$ be a polynomial bounding the size of $\langle \cdot \rangle$, in particular let $|\langle x_1, x_2, \ldots, x_j \rangle| \leq s(\max \{|x_1|, |x_2|, \ldots, |x_j|\})$ for all $1 \leq j \leq m + 1$ and all $x_1, x_2, \ldots, x_j \in \Sigma^*$. Define $s^{(0)}(n) = n$ and $s^{(j)}(n) = s(s(\cdots s(n) \cdots))$ for all $n$ and all $j \geq 1$. 7
For every $k \geq 1$, let $L_{\Sigma_k^p}$ be a many-one complete language for $\Sigma_k^p$ and hence $L_{\Pi_k^p} = \overline{L_{\Sigma_k^p}}$ is a complete language for $\Pi_k^p$. Define $L_{\text{DIFF},(\Pi_k^p)} = L_{\Pi_k^p}$, and for every $m \geq 2$, $L_{\text{DIFF},m(\Pi_k^p)} = \{\langle x, y \rangle | x \in L_{\Pi_k^p} \land y \notin L_{\text{DIFF},m-1(\Pi_k^p)}\}$. It is not hard to verify that for all $m \geq 1$, $L_{\text{DIFF},m(\Pi_k^p)}$ is many-one complete for $\text{DIFF}_m(\Pi_k^p)$. So, $L_{\text{DIFF},m(\Pi_k^p)}$ is complete for $\text{coDIFF}_m(\Pi_k^p)$ for all $m$.

Note also that $\text{DIFF}_m(\Pi_k^p) = \text{DIFF}_m(\Sigma_k^p)$ if $m$ is even and $\text{DIFF}_m(\Pi_k^p) = \text{coDIFF}_m(\Sigma_k^p)$ if $m$ is odd. So in general,

$$\text{DIFF}_m(\Sigma_k^p) = \text{coDIFF}_m(\Sigma_k^p) \iff \text{DIFF}_m(\Pi_k^p) = \text{coDIFF}_m(\Pi_k^p).$$

B Suppose $\text{DIFF}_m(\Sigma_k^p) = \text{coDIFF}_m(\Sigma_k^p)$. Hence $\text{DIFF}_m(\Pi_k^p) = \text{coDIFF}_m(\Pi_k^p)$. Thus there exists a many-one reduction $h$ from $L_{\text{DIFF},m(\Pi_k^p)}$ to $L_{\text{DIFF},m(\Pi_k^p)}$. In other words, for all $x_1, x_2 \in \Sigma^*$,

$$\langle x_1, x_2 \rangle \in L_{\text{DIFF},m(\Pi_k^p)} \iff h(\langle x_1, x_2 \rangle) \notin L_{\text{DIFF},m(\Pi_k^p)}.$$ Let $h'$ and $h''$ be the polynomial-time computable functions such that for all $x_1, x_2 \in \Sigma^*$,

$$h(\langle x_1, x_2 \rangle) = h'(\langle x_1, x_2 \rangle), h''(\langle x_1, x_2 \rangle).$$ Hence, we have for all $x_1, x_2 \in \Sigma^*$,

$$\langle x_1, x_2 \rangle \in L_{\Pi_k^p} \land x_2 \notin L_{\text{DIFF},m-1(\Pi_k^p)} \iff h'(\langle x_1, x_2 \rangle) \notin L_{\Pi_k^p} \lor h''(\langle x_1, x_2 \rangle) \in L_{\text{DIFF},m-1(\Pi_k^p)}.$$ C Recall that we want to show a collapse of the polynomial hierarchy. Though we do not claim that we can prove $\Sigma_k^p = \Pi_k^p$ we will nevertheless show that a $\Sigma_k^p$ algorithm for $L_{\Pi_k^p}$ exists which requires certain additional input. We will extend this to also give $\Sigma_k^p$ algorithms for $L_{\Sigma_k^{p+1}}$ and $L_{\Sigma_k^{k+2}}$, 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 $(\exists x_2 | x_2 | s^{(m-2)}(n))[h'(\langle x_1, x_2 \rangle) \notin L_{\Pi_k^p}]$. Clearly, if $x_1$ is $m$-easy for length $n$ then $x_1 \in L_{\Pi_k^p}$.

A string $x_1$ is said to be $m$-hard for length $n$ if and only if $|x_1| \leq n, x_1 \in L_{\Pi_k^p}$, and $(\forall x_2 | x_2 | s^{(m-2)}(n))[h'(\langle x_1, x_2 \rangle) \in L_{\Pi_k^p}]$. It is not hard to verify that the strings in $(L_{\Pi_k^p})^{\leq n}$ 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 $(L_{\Pi_k^p})^{\leq n}$ are $m$-easy for length $n$. Thus deciding whether $x, |x| = n,$ is in $L_{\Pi_k^p}$ is equivalent to deciding whether $x$ is $m$-easy for length $n$. Note that the latter can be done by the following $\Sigma_k^p$ algorithm:

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

Case 2 There exist $m$-hard strings for length $n$.

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

$$u \in L_{\text{DIFF},m-1(\Pi_k^p)} \iff h_{(\omega_m)}(u) \notin L_{\text{DIFF},m-1(\Pi_k^p)}.$$
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. Let \( h'_{(\omega_m)} \) and \( h''_{(\omega_m)} \) be the functions such that for all \( x_1, x_2 \in \Sigma^* \), \( h'_{(\omega_m)}((x_1, x_2)) = (h'_{(\omega_m)}((x_1), x_2)), h''_{(\omega_m)}((x_1, x_2)). \)

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

\[
u_1 \in L^p_{\Pi_k} \land u_2 \notin L_{\text{DIFF}_{m-2}(\Pi_k)} \iff h'_{(\omega_m)}((u_1, u_2)) \notin L^p_{\Pi_k} \lor h''_{(\omega_m)}((u_1, u_2)) \in L_{\text{DIFF}_{m-1}(\Pi_k)}.
\]

We call the string \( u_1 \) \((m-1)\)-easy for length \( n \) if and only if \(|u_1| \leq n \) and \((\exists u_2 | u_2| \leq s^{(m-3)}(n))h'_{(\omega_m)}((u_1, u_2)) \notin L^p_{\Pi_k}\). If \( u_1 \) is \((m-1)\)-easy for length \( n \) then \( u_1 \in L^p_{\Pi_k} \).

A string \( u_1 \) is said to be \((m-1)\)-hard for length \( n \) if and only if \(|u_1| \leq n \), \( u_1 \in L^p_{\Pi_k} \), and \((\forall u_2 | u_2| \leq s^{(m-3)}(n))h''_{(\omega_m)}((u_1, u_2)) \in L^p_{\Pi_k}\).

It is not hard to verify that, given an \( m \)-hard string \( \omega_m \) for length \( n \), the strings in \( (L^p_{\Pi_k})_{\leq n} \) 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 \( \omega_m \).

**Case 2.1** There exist no \((m-1)\)-hard strings for length \( n \).

Hence similar to Case 1, all strings in \( (L^p_{\Pi_k})_{\leq n} \) are \((m-1)\)-easy for length \( n \), deciding whether \( x, |x| = n \), is in \( L^p_{\Pi_k} \) is equivalent to deciding whether \( x \) is \((m-1)\)-easy for length \( n \) which, with the help of \( \omega_m \), can be done with a \( \Sigma^p_k \) algorithm.

**Case 2.2** There exist \((m-1)\)-hard strings for length \( n \).

Let \( \omega_{m-1} \) be an \((m-1)\)-hard string for length \( n \), \(|\omega_{m-1}| \leq n \). Let \( h_{(\omega_m,\omega_{m-1})} \) be the function such that for all \( v \in \Sigma^* \), \( h_{(\omega_m,\omega_{m-1})}(v) = h''_{(\omega_m)}((\omega_{m-1}, v)) \).

Note that given \( \omega_m \) and \( \omega_{m-1} \), \( h_{(\omega_m,\omega_{m-1})}(v) \) is computable in time polynomial in \( \max\{n, |v|\} \).

Hence, for all \( v, |v| \leq s^{(m-3)}(n), \)

\[
v \in L_{\text{DIFF}_{m-2}(\Pi_k)} \iff h_{(\omega_m,\omega_{m-1})}(v) \notin L_{\text{DIFF}_{m-2}(\Pi_k)}.
\]

Continuing in that manner we define for \( \ell \geq 2 \), \( \ell \)-hard and \( \ell \)-easy strings for length \( n \). Note that these terms are defined with respect to some fixed \( m \)-hard, \((m-1)\)-hard, \( \ldots \), \((\ell+1)\)-hard strings. In other words, a string is only \( \ell \)-hard or \( \ell \)-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 \( h_{(\omega_m,\omega_{m-1},\ldots,\omega_2)}(z) \notin L^p_{\Pi_k} \).

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 \( \omega_j \) is \( j \)-hard (with respect to \( \omega_m, \omega_{m-1}, \ldots, \omega_{j+1} \)) for length \( n \) for all \( j, \ell \leq j \leq m \). 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 \( \omega_m, \omega_{m-1}, \ldots, \omega_\ell \)) for length \( n \). As a special case, \( \# \) 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 \( \Pi^p_k \) 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 part of $L_{\Pi_k^p}$ to a $\Sigma_k^p$ language.

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

$$x \in L_{\Pi_k^p} \iff \langle x, 1^l, \omega_m, \omega_{m-1}, \ldots, \omega_l \rangle \in A.$$ 

Let $x \in \Sigma^*$ and let $\omega_m, \omega_{m-1}, \ldots, \omega_l$ be a maximal hard sequence for length $l$, $l \geq |x|$. Note that $\ell \geq 2$. Since $\omega_m, \omega_{m-1}, \ldots, \omega_l$ is maximal hard, no string of length less or equal to $l$ is $(\ell - 1)$-hard with respect to $\omega_m, \omega_{m-1}, \ldots, \omega_l$. Hence, for every string $y$, $|y| \leq l$, $y \in (L_{\Pi_k^p})^{\leq l}$ 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 a $\Sigma_k^p$ algorithm when receiving $x$, $1^l$, and $\omega_m, \omega_{m-1}, \ldots, \omega_l$ as inputs. In particular, define $A = \{ \langle x, 1^l, \omega_m, \omega_{m-1}, \ldots, \omega_l \rangle \mid (\ell = 2 \land h(\omega_m, \omega_{m-1}, \ldots, \omega_l)(x) \notin L_{\Pi_k^p}) \lor (\ell > 2 \land (\exists y \ |y| \leq s^{\ell-2}(\ell)) [h(\omega_m, \omega_{m-1}, \ldots, \omega_l)(x, y) \notin L_{\Pi_k^p}] \}.$

E One maximal hard sequence for sufficiently large length also suffices to give a reduction from some of $L_{\Sigma_{k+1}^p}$ to a $\Sigma_k^p$ language.

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

$$x \in L_{\Sigma_{k+1}^p} \iff \langle x, 1^l, \omega_m, \omega_{m-1}, \ldots, \omega_l \rangle \in B.$$ 

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

$$x \in L_{\Sigma_{k+1}^p} \iff (\exists y \ |y| \leq p(|x|))(x, y) \in L_{\Pi_k^p}.$$

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

$$x \in L_{\Sigma_{k+1}^p} \iff (\exists y \ |y| \leq p(|x|))(\langle x, y \rangle, 1^l, \omega_m, \omega_{m-1}, \ldots, \omega_l) \in A.$$

Note that the right-hand-side of the above equivalence clearly defines a $\Sigma_k^p$ language $B$. Define $q$ to be a polynomial such that $q(n) \geq s(p(n))$ 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 some of $L_{\Sigma_{k+2}^p}$ to a $\Sigma_k^p$ language.

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

$$x \in L_{\Sigma_{k+2}^p} \iff \langle x, (\omega_m, \omega_{m-1}, \ldots, \omega_l), (\omega'_m, \omega'_{m-1}, \ldots, \omega'_l) \rangle \in C.$$ 

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

$$x \in L_{\Sigma_{k+2}^p} \iff (\exists y \ |y| \leq p'(|x|))(x, y) \notin L_{\Sigma_{k+1}^p}.$$
Applying Claim E we obtain that there is a set $B \in \Sigma^p_k$ and a polynomial $q$ such that for all $x \in \Sigma^*$ and all $l \geq q(s(p′(|x|)))$, if $\omega_m, \omega_{m-1}, \ldots, \omega_l$ is a maximal hard sequence for length $l$ then
\[
x \in L^p_{\Sigma^k_{l+2}} \iff (\exists y \mid |y| \leq p′(|x|))(\langle x, y \rangle, 1^l, \omega_m, \omega_{m-1}, \ldots, \omega_l) \not\in B.
\]
Define $q_1$ to be a polynomial such that $q_1(n) \geq q(s(p′(n)))$ for all $n$. Define $L′ = \{(x, 1^l, \omega_m, \omega_{m-1}, \ldots, \omega_l) \mid (\exists y \mid |y| \leq p′(|x|))(\langle x, y \rangle, 1^l, \omega_m, \omega_{m-1}, \ldots, \omega_l) \not\in B\}$. Note that $L′ \in \Sigma^p_{k+1}$ and let $g$ be a many-one reduction from $L′$ to $L^p_{\Sigma^k_{l+1}}$. Hence we have for all $x \in \Sigma^*$, if $\omega_m, \omega_{m-1}, \ldots, \omega_l$ is a maximal hard sequence for length $q_1(|x|)$ then
\[
x \in L^p_{\Sigma^k_{l+2}} \iff g(\langle x, 1^{q_1(|x|)}, \omega_m, \omega_{m-1}, \ldots, \omega_l \rangle) \in L^p_{\Sigma^k_{l+1}}.
\]
Applying Claim E for the second time we obtain that for all $x \in \Sigma^*$, if $\omega_m, \omega_{m-1}, \ldots, \omega_l$ is a maximal hard sequence for length $q_1(|x|)$ and $\omega_m′, \omega_{m-1}′, \ldots, \omega_l′$ is a maximal hard sequence for length $l$, $l \geq q(g(\langle x, 1^{q_1(|x|)}, \omega_m, \omega_{m-1}, \ldots, \omega_l \rangle))$, then
\[
x \in L^p_{\Sigma^k_{l+2}} \iff g(\langle x, 1^{q_1(|x|)}, \omega_m, \omega_{m-1}, \ldots, \omega_l \rangle), 1^l, \omega_m′, \omega_{m-1}′, \ldots, \omega_l′ \in B.
\]
Let $\hat{q}$ be a polynomial such that $|\hat{q}(z)|$ is bounded by $\hat{q}(|z|)$ for all $z$. Define $q_2$ to be a polynomial such that $q_2(n) \geq \hat{q}(\hat{q}(s(q_1(n))))$ for all $n$. Set
\[
C = \{(x, \omega_m, \omega_{m-1}, \ldots, \omega_l), (\omega_m′, \omega_{m-1}′, \ldots, \omega_l′) \mid g(\langle x, 1^{q_1(|x|)}, \omega_m, \omega_{m-1}, \ldots, \omega_l \rangle), 1^l, \omega_m′, \omega_{m-1}′, \ldots, \omega_l′ \in B\}
\]
and note that clearly $C \in \Sigma^p_k$. This proves the claim.

**Claim G**: Applying the so-called mind change technique in light of Claim C yields that $L^p_{\Sigma^k_{l+2}} \in \Sigma^p_k \Delta \text{DIFF}_{2m-2}(\Sigma^p_{k+1})$.

To prove the claim it suffices to give a $\Sigma^p_k \Delta \text{DIFF}_{2m-2}(\Sigma^p_{k+1})$ algorithm for $L^p_{\Sigma^k_{l+2}}$.

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 $x, x \in L^p_{\Sigma^k_{l+2}}$ 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 $\langle x, s_1, s_2 \rangle \in C$.

Define $Q_0 = \{x \mid \langle x, \#, \# \rangle \in C\}$. Define for $1 \leq j$,
\[
Q_j = \{x \mid \text{there exist } r_1, s_1, r_2, s_2, \ldots, r_j, s_j \text{ such that}
\]
\[1. \text{for all } 1 \leq i \leq j, \text{ } r_i \text{ and } s_i \text{ are hard sequences for length } q_1(|x|) \text{ and } q_2(|x|), \text{ respectively};
\]
\[2. \text{for all } 1 \leq i \leq j-1, \text{ } (r_{i+1}, s_{i+1}) \text{ is a proper extension of } (r_i, s_i), \text{ and}
\]
\[3. \chi_C(\langle x, \#, \# \rangle) \neq \chi_C(\langle x, (r_1), (s_1) \rangle) \text{ and for all } 1 \leq i \leq j-1, \chi_C(\langle x, (r_i), (s_i) \rangle) \neq \chi_C(\langle x, (r_{i+1}), (s_{i+1}) \rangle).\]
Observe that $Q_0 \in \Sigma^p_k$ and $Q_j \in \Sigma^p_{k+1}$, $1 \leq j$. 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 point 3 in the definition of $Q_j$ requires $(r_1, s_1) \neq (\#, \#)$ we obtain that for all $j > 2m-2$, $Q_j = \emptyset$. Furthermore, it is not hard to verify that for all $x \in \Sigma^*$,

$$x \in L_{\Sigma^p_{k+2}} \iff x \in Q_0 \Delta (Q_1 - (Q_2 - (\cdots - (Q_{2m-3} - Q_{2m-2}) \cdots))).$$

This shows $L_{\Sigma^p_{k+2}} \subseteq \Sigma^p_k \Delta \text{DIFF}_{2m-2}(\Sigma^p_{k+1})$.

**Claim H:** $\text{PH} \subseteq \text{DIFF}_{2m-1}(\Sigma^p_k) \Delta \text{DIFF}_{m-1}(\Sigma^p_{k+1})$.

In light of Claim G and the fact that $\Sigma^p_k \Delta \text{DIFF}_{2m-2}(\Sigma^p_{k+1}) \subseteq \text{DIFF}_{2m-1}(\Sigma^p_{k+1})$ it suffices to show $\text{DIFF}_{2m-1}(\Sigma^p_{k+1}) \subseteq \text{DIFF}_{2m-1}(\Sigma^p_k) \Delta \text{DIFF}_{m-1}(\Sigma^p_{k+1})$.

Let $L \in \text{DIFF}_{2m-1}(\Sigma^p_{k+1})$, hence there exist sets $L_1, L_2, \ldots, L_{2m-1} \in \Sigma^p_k$ such that $L = L_1 \cup L_2 \cup (\cdots \cup L_{2m-2} \cup L_{2m-1})$. According to Claim E (note that Claim E can be easily extended to hold for all $\Sigma^p_k$ languages and not just for $L_{\Sigma^p_k}$) there exist sets $B_1, B_2, \ldots, B_{2m-1} \in \Sigma^p_k$ and polynomials $p_1, p_2, \ldots, p_{2m-1}$ such that for all $x \in \Sigma^*$ and all $1 \leq i \leq 2m-1$, if $\omega^i_1, \omega^i_{m-1}, \ldots, \omega^i_1$ is a maximal hard sequence for length $l_i$, $l_i \geq p_i(|x|)$, then

$$x \in L_i \iff \langle x, 1^{l_i}, \omega^i_m, \omega^i_{m-1}, \ldots, \omega^i_1 \rangle \in B_i.$$

Let $\tilde{p}$ be a polynomial such that $\tilde{p}(n) \geq p_i(n)$ for all $n$ and all $1 \leq i \leq 2m-1$. Define $D = \{(x, 1^{\tilde{p}(|x|)}, \omega^i_m, \omega^i_{m-1}, \ldots, \omega^i_1) \in L_1 \cup L_2 \cup (\cdots \cup L_{2m-2} \cup L_{2m-1}) \}$. Note that $D \in \text{DIFF}_{2m-1}(\Sigma^p_k)$. We have for all $x \in \Sigma^*$, if $\omega^i_m, \omega^i_{m-1}, \ldots, \omega^i_1$ is a maximal hard sequence for length $\tilde{p}(|x|)$ then

$$x \in L \iff \langle x, \omega^i_m, \omega^i_{m-1}, \ldots, \omega^i_1 \rangle \in D.$$

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 $P_0 = \{x \mid (x, \#) \in D\}$. Define for $1 \leq j$,

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

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

Note that $P_0 \in \text{DIFF}_{2m-1}(\Sigma^p_k)$ and $P_j \in \Sigma^p_{k+1}, 1 \leq j$. Since all hard sequences have order at most $m-1$ and point 3 in the definition of $P_j$ requires $s_1 \neq \#$ we obtain that for all $j > m-1$, $P_j = \emptyset$. Thus 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 $L \in \text{DIFF}_{2m-1}(\Sigma^p_k) \Delta \text{DIFF}_{m-1}(\Sigma^p_{k+1})$. 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: $\text{PH} \subseteq \text{DIFF}_m(\Sigma^p_k) \Delta \text{DIFF}_{m-1}(\Sigma^p_{k+1})$.

Observe that $\text{DIFF}_{2m-1}(\Sigma^p_k) \Delta \text{DIFF}_{m-1}(\Sigma^p_{k+1}) \subseteq \text{DIFF}_m(\Sigma^p_{k+1})$. In light of Claim H it suffices to show $\text{DIFF}_m(\Sigma^p_{k+1}) \subseteq \text{DIFF}_m(\Sigma^p_{k+1}) \Delta \text{DIFF}_{m-1}(\Sigma^p_{k+1})$ which can be done quite analogous to the proof of Claim H.

5 The Evolution of the Easy-Hard Technique

This section is structured as follows. Section 5.1 gives a short, non-technical, exceedingly informal summary of the technical contribution of each of the nine papers. Section 5.2 gives a detailed explanation of the easy-hard technique, and of the specific technical advances made by each of the nine papers.

5.1 In a Nutshell

The term easy-hard originates from Kadin’s observation that in case that the boolean hierarchy collapses at level $m$ the strings of any particular length $n$ in a coNP complete language divide into easy and hard strings. Hard strings are strings that allow one to translate a collapse of the boolean hierarchy from level $m$ to level $m - 1$ in a restricted sense. Several hard strings eventually allow one to reduce a coNP predicate to a NP predicate. In contrast, if no hard string at length $n$ exists then all strings of length $n$ in the coNP complete language are easy and this allows one to directly reduce a coNP predicate to a NP predicate. So, if we know whether there exist hard strings or not, and if, in case they exist, we are able to effectively compute them, we can with their help reduce a coNP predicate to a NP predicate and eventually collapse the polynomial hierarchy. This approach is central in each of the first five papers of our nine-paper survey. The major difference among the five papers, and the main reasons for the difference in their results, is the way in which one obtains the needed information about the hard strings (their existence and the strings itself), and in which way one uses this information to collapse the polynomial hierarchy. The last four papers of our survey use the (modified) easy-hard method to prove downward collapse results within the polynomial hierarchy.

Kadin [Kad88] constructed a sparse set $S$ containing enough information such that one can effectively extract a lexicographically extreme hard string for a given length if there exists one. He shows that $\text{coNP} \subseteq \text{NP}^S$, which by a result of Yap [Yap83] implies $\text{PH} \subseteq \Sigma^p_3$.\footnote{In his original work Kadin claimed that his sparse set $S$ is even contained in $\Sigma^p_2$ (a claim he retracted later), which would have allowed him to conclude $\text{PH} = \Theta^p_{k+2}$.}

Wagner used a different approach in his two papers [Wag87, Wag89]. He collapsed the polynomial hierarchy directly (without constructing a sparse oracle) using oracle replacement and hard strings in the 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 (and thus one needs a hard string for each length when collapsing the polynomial hierarchy), Wagner’s new definition yields that hard strings can give a reduction for all strings of length below a particular threshold.

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 one to effectively gain information about the existence of hard strings at lower length together with an elegant use of the nested difference structure of the boolean hierarchy over $\Sigma^p_2$, leads to their final result.
Beigel, Chang, and Ogihara \cite{BCO93} further improved the results of Chang and Kadin. They followed the approach of Wagner, but with two major innovations: First, they used complete languages for the levels of the boolean hierarchy that do not force them to distinguish between odd and even levels. Second, they made use of the mind change technique to effectively check the existence of hard strings and their effect on the outcome of the reduction using those hard strings. Their second innovation includes a modified argumentation for collapsing the polynomial hierarchy.

The key contribution of \cite{HHH96b} was the insight that, since each string is easy or hard, one can completely discard the search for such strings! Rather, one can simply always use the input itself as an easy or hard string (whichever it happened to be). \cite{HHH96a} and \cite{BF96} extended this approach in two different directions. The approach of \cite{HHH96b} works only for 1-vs-2 query access to $\Sigma_p^k$, $k > 2$. \cite{BF96} extends the result to the 1-vs-2 query case for $k \geq 2$, via 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. They add a new bit of “code” to ensure that the nondeterministic branch making the wrong assumption will do no harm to the overall algorithm. \cite{HHH}, on the other hand, removes the 1-vs-2 restriction but adds a scheme implementing “0-bit communication” between machines. They do this by having the machines independently latch onto a certain lexicographically extreme string signaled by the input.

However, recall from Section 3 the two improvements just mentioned—from $k > 2$ to $k \geq 2$ via \cite{BF96} and from 1-vs-2 to $j$-vs-$j+1$ via \cite{HHH}—are incomparable. Neither paper allows both improvements to work simultaneously. However, this was achieved in \cite{HHH97}, via a new twist. \cite{HHH97} provides an improved way of allowing the underlying $\Sigma_p^k$ machines of DIFF($\Sigma_p^k$) languages to work together. In particular, \cite{HHH97} does so by exploiting the so-called telescoping normal form of boolean (or difference) hierarchies \cite{CGH+88—a normal form that in concept dates as far back as the work of Hausdorff \cite{Hau14}.

5.2 Detailed Technical Discussion of the Easy-Hard Technique and its Extensions

In this section we study the easy-hard technique in its original version, and in the increasingly strong extensions developed in the series of papers that this article studies. We will in the following prove for each of the papers just a special case of the key result obtained. In particular we will start from one of the assumptions $P^{\Sigma_p^1} = P^{\Sigma_p^2}$, $P^{\Sigma_p^2}_{2{\text{tt}}} = P^{\Sigma_p^3}_{3{\text{tt}}}$, $P^{\Sigma_p^1}[1] = P^{\Sigma_p^2}[2]$, or $P^{\Sigma_p^2}_{2{\text{tt}}} = P^{\Sigma_p^3}_{3{\text{tt}}}$ and prove the collapse of the polynomial hierarchy obtained by the corresponding paper.

Let $\Sigma = \{0, 1\}$ and let $\# \not\in \Sigma$ be a new symbol. Let $\langle \cdot \cdot \cdot \rangle$ be a pairing function mapping finite sequences of strings from $\Sigma^* \cup \{\#\}$ to $\Sigma^*$, and let this function have the standard properties, such as for instance being polynomially computable and invertible. 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 $|\langle x, y, z \rangle| \leq s(\max\{|x|, |y|, |z|\})$. Recall our convention regarding polynomials from Section 4, in particular, 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, we without loss of generality assume that such polynomials always have the form $n^a + b$ for some integers $a, b > 0$. This, for example, guarantees that for a polynomial $p$ it now holds that $p(n + 1) > p(n) > n$ for all $n$.

1) Kadin 1987 \cite{Kad87,Kad88}

Theorem 5.1 If $P^{\Sigma_p^1} = P^{\Sigma_p^2}[2]$ then $PH = \Sigma_p^3$.

Proof:
Main Claim: If $D \cdot \Sigma^p_3 = \text{co}D \cdot \Sigma^p_3$ then $PH = \Sigma^p_5$.

where $D \Sigma^p_3$ is the $\Sigma^p_3$ analogue of the class DP [PY84], in particular $D \Sigma^p_3 = \{L_1 - L_2 \mid L_1, L_2 \in \Sigma^p_3\}$.

Since $P^{\Sigma^p_3[1]} \subseteq D \cdot \Sigma^p_3 \subseteq P^{\Sigma^p_3[2]}$ and $P^{\Sigma^p_3[1]} \subseteq \text{co}D \cdot \Sigma^p_3 \subseteq P^{\Sigma^p_3[2]}$ the theorem follows immediately from the above claim. In what follows we will prove the correctness of the main claim.

Proof of Main Claim:

A Suppose $D \cdot \Sigma^p_3 = \text{co}D \cdot \Sigma^p_3$. Let $L_{\Sigma^p_3}$ be a many-one complete language for $\Sigma^p_3$. It is not hard to verify that $L_{D, \Sigma^p_3} = \{(x, y) \mid x \in L_{\Sigma^p_3} \land y \notin L_{\Sigma^p_3}\}$ is a many-one complete language for $D \cdot \Sigma^p_3$.

According to our assumption, $D \Sigma^p_3 = \text{co}D \cdot \Sigma^p_3$, there is a polynomial-time computable function $h$ reducing $L_{D, \Sigma^p_3}$ to $L_{D, \Sigma^p_3}$, i.e. for all $x_1, x_2 \in \Sigma^*$,

$$\langle x_1, x_2 \rangle \in L_{D, \Sigma^p_3} \iff h((x_1, x_2)) \in L_{D, \Sigma^p_3}.$$ 

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

$$h((x_1, x_2)) = (h'((x_1, x_2)), h''((x_1, x_2))).$$

Hence

$$x_1 \in L_{\Sigma^p_3} \land x_2 \notin L_{\Sigma^p_3} \iff h'((x_1, x_2)) \notin L_{\Sigma^p_3} \lor h''((x_1, x_2)) \in L_{\Sigma^p_3}.$$ 

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 $(\exists x_1 \mid |x_1| = |x_2|)[h''((x_1, x_2)) \in L_{\Sigma^p_3}]$. Clearly, if $x_2$ is easy then $x_2 \notin L_{\Sigma^p_3}$. But note that checking whether a particular string is easy can be done with a $\Sigma^p_3$ algorithm.

$x_2$ is said to be hard if and only if $x_2 \notin L_{\Sigma^p_3}$ and $(\forall x_1 \mid |x_1| = |x_2|)[h''((x_1, x_2)) \notin L_{\Sigma^p_3}]$. Hence, if $x_2$ is a hard string we have for all $x_1$, $|x_1| = |x_2|$,

$$x_1 \in L_{\Sigma^p_3} \iff h'((x_1, x_2)) \notin L_{\Sigma^p_3}.$$ 

Note that the strings in $L_{\Sigma^p_3}$ divide into easy and hard strings.

C Define the set $S' = \{\omega \mid \omega$ is the lexicographically smallest hard string of length $|\omega|\}$ and the set $S$ of marked prefixes of $S'$, $S = \{y\#^i \mid i \geq 0 \land (\exists v \mid |v| = i)[yv \in S']\}$. Note that $S$ is sparse.

D Claim D: $\Pi^p_3 \subseteq (\Sigma^p_3)^S$.

We will prove the above claim by giving a $(\Sigma^p_3)^S$ algorithm for $L_{\Sigma^p_3}$:

1. On input $x$, $|x| = n$, check whether $S^{=n}$ is empty or not. This can be done by querying $0\#^{n-1} \in S$ and $1\#^{n-1} \in S$. 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, $x \in L_{\Sigma^p_3}$ if and only if $x$ is easy. Thus, guess $x_1$, $|x_1| = n$, compute $h((x_1, x))$, and accept if and only if $h''((x_1, x)) \in L_{\Sigma^p_3}$.

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 $\omega$, with adaptive queries to $S^{=n}$. Compute $h((x, \omega))$ and accept if and only if $h'((x, \omega)) \in L_{\Sigma^p_3}$. 

15
According to B, this algorithm is correct.

E By a result of Yap [Yap83], $\Pi^p_j \subseteq (\Sigma^p_j)^S$ for a sparse set $S$ implies $\Sigma^p_0 = \Pi^p_3$ and hence $PH = \Sigma^p_1$.

**End of Proof of Main Claim**

2) Wagner 1987 [Wag87]

**Theorem 5.2** If $P^{\Sigma^p_1} = P^{\Sigma^p_1} = P^{\Sigma^p_1}$ then $PH = P^{\Sigma^p_1}$.

**Proof:**

**Main Claim** If $D \cdot \Sigma^p_3 = coD \cdot \Sigma^p_3$ then $PH = P^{\Sigma^p_1}$.

Clearly, the theorem follows as an immediate corollary from the above claim. We now prove the main claim.

**Proof of Main Claim:**

A and B As in the proof of Theorem 5.1 (Kadin 1987).

C Let $L_{\Sigma^p_3}$, $L_{\Sigma^p_4}$, and $L_{\Sigma^p_5}$ be many-one complete languages for $\Sigma^p_3$, $\Sigma^p_4$, and $\Sigma^p_5$, respectively, $p_4$ and $p_5$ be polynomials such that

$$L_{\Sigma^p_4} = \{ x \mid (\exists y \mid |y| \leq p_4(|x|))[(x, y) \not\in L_{\Sigma^p_3}] \}$$

and

$$L_{\Sigma^p_5} = \{ x \mid (\exists y \mid |y| \leq p_5(|x|))[(x, y) \not\in L_{\Sigma^p_4}] \}.$$ 

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 $f \in FP^{\Sigma^p_1}[j]$, where $FP^{\Sigma^p_1}[j]$ is defined similar to $P^{\Sigma^p_1}[j]$ with the modification that the base $P$ machine computes a function instead of accepting a language. Note that $f(x)$ is equal for all equal-length strings $x$.

We call $(1^l, \#)$ a hard pair if and only if $f(1^l) = 0$. $(1^l, y)$, $y \in \Sigma^*$, 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 $L_{\Sigma^p_3}$ to a $\Sigma^p_3$ language.

**Claim E:** There exists a set $A \in \Sigma^p_3$ such that for all $x \in \Sigma^*$, if $(1^{|x|}, \omega)$ is a hard pair then

$$x \not\in L_{\Sigma^p_3} \iff (x, \omega) \in A.$$ 

Let $x \in \Sigma^*$. Let $(1^{|x|}, \omega)$ 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 $|y| = |x|$, $y \not\in L_{\Sigma^p_3}$ if and only if $y$ is easy. This holds in particular for $x$ itself. According to B we thus have

$$x \not\in L_{\Sigma^p_3} \iff (\exists x_1 \mid |x_1| = |x|)[h''(x_1, x) \in L_{\Sigma^p_3}].$$
Applying Claim E, a series of hard pairs of growing length gives a reduction from $G$. According to Claim E there exists a set $A$ such that for all $0 \leq i \leq q(|x|)$, $(1^i, \omega_i)$ is a hard pair then $x \in L_{\Sigma_4^p} \iff \langle x, \omega_0, \omega_1, \ldots, \omega_q(|x|) \rangle \in B$.

Let $x \in \Sigma^*$. By definition of $L_{\Sigma_4^p}$ we have

$$x \in L_{\Sigma_4^p} \iff (\exists y \ |y| \leq p_4(|x|))[\langle x, y \rangle \not\in L_{\Sigma_4^p}].$$

According to Claim E there exists a set $A \in \Sigma_4^p$ such that for all $y \in \Sigma^*$ if $(1^i, \omega_i) \not\in A$ is a hard pair then $x \not\in L_{\Sigma_4^p} \iff \langle x, y \rangle \not\in A$.

Hence if for all $0 \leq i \leq s(p_4(|x|))$, $(1^i, \omega_i)$ is a hard pair then $x \in L_{\Sigma_4^p} \iff (\exists y \ |y| \leq p_4(|x|))[\langle x, y \rangle \not\in A]$.

Let $q$ be a polynomial such that $q(n) \geq s(p_4(n))$ for all $n$. Define

$$B = \{\langle x, \omega_0, \omega_1, \ldots, \omega_q(|x|) \rangle | (\exists y \ |y| \leq p_4(|x|))[\langle x, y \rangle \not\in A]\}$$

and note that $B \in \Sigma_4^p$. This proves Claim F.

Applying Claim E, a series of hard pairs of growing length gives a reduction from $L_{\Sigma_4^p}$ to a $\Sigma_3^p$ language.

Claim F: There exist a set $B \in \Sigma_3^p$ and a polynomial $q$ such that for all $x \in \Sigma^*$, if for all $0 \leq i \leq q(|x|)$, $(1^i, \omega_i)$ is a hard pair then $x \in L_{\Sigma_4^p} \iff \langle x, \omega_0, \omega_1, \ldots, \omega_q(|x|) \rangle \in B$.

Let $x \in \Sigma^*$. By definition of $L_{\Sigma_4^p}$ we have

$$x \in L_{\Sigma_4^p} \iff (\exists y \ |y| \leq p_4(|x|))[\langle x, y \rangle \not\in L_{\Sigma_4^p}].$$

According to Claim E there exists a set $A \in \Sigma_4^p$ such that for all $y \in \Sigma^*$ if $(1^i, \omega_i) \not\in A$ is a hard pair then $x \not\in L_{\Sigma_4^p} \iff \langle x, y \rangle \not\in A$.

Hence if for all $0 \leq i \leq s(p_4(|x|))$, $(1^i, \omega_i)$ is a hard pair then $x \in L_{\Sigma_4^p} \iff (\exists y \ |y| \leq p_4(|x|))[\langle x, y \rangle \not\in A]$.

Let $q$ be a polynomial such that $q(n) \geq s(p_4(n))$ for all $n$. Define

$$B = \{\langle x, \omega_0, \omega_1, \ldots, \omega_q(|x|) \rangle | (\exists y \ |y| \leq p_4(|x|))[\langle x, y \rangle \not\in A]\}$$

and note that $B \in \Sigma_3^p$. This proves Claim F.

Taking the result of Claim F one step further, hard pairs of growing length provide a reduction from $L_{\Sigma_4^p}$ to a $\Sigma_4^p$ language.

Claim G: There exist a set $D \in \Sigma_4^p$ and a polynomial $p$ such that for all $x \in \Sigma^*$, if for all $0 \leq i \leq p(|x|)$, $(1^i, \omega_i)$ is a hard pair then $x \in L_{\Sigma_4^p} \iff \langle x, \omega_0, \omega_1, \ldots, \omega_{p(|x|)} \rangle \in D$.

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

$$x \in L_{\Sigma_4^p} \iff (\exists y \ |y| \leq p_5(|x|))[\langle x, y \rangle \not\in L_{\Sigma_4^p}].$$

According to Claim F there exist a set $B \in \Sigma_3^p$ and a polynomial $q$ such that for all $y$, $|y| \leq p_5(|x|)$, if for all $0 \leq i \leq q(s(p_5(|x|)))$, $(1^i, \omega_i)$ is a hard pair, then $\langle x, y \rangle \in L_{\Sigma_4^p} \iff \langle \langle x, y \rangle, \omega_0, \omega_1, \ldots, \omega_q(|x,y|) \rangle \not\in B$.
and hence
\[ x \in L_{\Sigma^p_{\mathsf{5}}} \iff (\exists y \mid |y| \leq p_5(|x|))[\langle (x, y), \omega_0, \omega_1, \ldots, \omega_{q(|(x, y)|)} \rangle \notin B]. \]

Let \( p \) be a polynomial such that \( p(n) \geq q(s(p_5(n))) \) for all \( n \). Define
\[
D = \{(x, \omega_0, \omega_1, \ldots, \omega_{p(|x|)}) \mid (\exists y \mid |y| \leq p_5(|x|))[(\langle x, y \rangle, \omega_0, \omega_1, \ldots, \omega_{q(|(x, y)|)}) \notin B]\}.
\]

Clearly, \( D \in \Sigma^p_4 \).

**Claim H:** \( L_{\Sigma^p_{\mathsf{5}}} \in \mathsf{P}^{\Sigma^p_4} \).

Let \( D \in \Sigma^p_4 \) and \( p \) be a polynomial as defined in \( G \). We have the following \( \mathsf{P}^{\Sigma^p_4} \) algorithm for \( L_{\Sigma^p_{\mathsf{5}}} \):

1. On input \( x \), compute \( f(1^0), f(1^1), f(1^2), \ldots, f(1^{p(|x|)}) \). This can be done with \( p(|x|) \) parallel queries to a \( \Sigma^p_4 \) oracle since \( f \in \mathsf{FP}^{\Sigma^p_4}[1] \).
2. (a) For all \( 0 \leq i \leq p(|x|) \) such that \( f(1^i) = 0 \) set \( \omega_i = \# \).
   (b) For all \( 0 \leq i \leq p(|x|) \) such that \( f(1^i) = 1 \) guess \( \omega_i, |\omega_i| = i \), and verify that \( \omega_i \) is a hard string. Continue if this verification succeeds, otherwise reject.
   (c) Verify that \( \langle x, \omega_0, \omega_1, \ldots, \omega_{p(|x|)} \rangle \in D \).
   Note that all this can be done with a single \( \Sigma^p_4 \) oracle query when handing \( x, f(1^0), f(1^1), f(1^2), \ldots \), and \( f(1^{p(|x|)}) \) over to the oracle.
3. Accept if and only if the query under 2. returns “yes.”

It is not hard to verify that the above algorithm in light of Claim G proves the claim. Note that the oracles queried in steps 1 and 2 might be different. Since \( \Sigma^p_5 \) 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 \( L_{\Sigma^p_{\mathsf{5}}} \) is complete for \( \Sigma^p_5 \) we conclude \( \Sigma^p_5 = \mathsf{P}^{\Sigma^p_4} \) and thus \( \mathsf{PH} = \mathsf{P}^{\Sigma^p_4} \).

**End of Proof of Main Claim**

---

3) Wagner 1989 [Wag89]

**Theorem 5.3** If \( \mathsf{P}^{\Sigma^p_3[1]} = \mathsf{P}^{\Sigma^p_3[2]} \) then \( \mathsf{PH} = \mathsf{P}^{\Sigma^p_3[3]} \).

**Proof:**

**Main Claim** As in the proof of Theorem 5.2 (Wagner 1987), but \( \mathsf{P}^{\Sigma^p_4} \) replaced by \( \mathsf{P}^{\Sigma^p_3[3]} \).

**Proof of Main Claim:** The major difference to the proof of Theorem 5.2 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 5.2 to this new definition would suffice to prove Theorem 5.3. 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, they appear in the papers of Chang and Kadin [CK89,CK90] and Beigel, Chang, and Oghara [BCO91,BCO93] again and play an important role there.
A As in the proof of Theorem 5.1 (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 $(\exists x_1 |x_1| \leq l)[h''((x_1, x_2)) \in \Sigma_3^p]$. Clearly, if $x_2$ is easy for length $l$ then $x_2 \notin \Sigma_3^p$.

$x_2$ is said to be hard for length $l$ if and only if $|x_2| \leq l$, $x_2 \notin \Sigma_3^p$, and $(\forall x_1 |x_1| \leq l)[h''((x_1, x_2)) \notin \Sigma_3^p]$. Hence, if $x_2$ is a hard string for length $l$ we have for all $x_1$, $|x_1| \leq l$,

$$x_1 \in \Sigma_3^p \iff h''((x_1, x_2)) \notin \Sigma_3^p.$$  

Note that the strings in $(\Sigma_3^p)^{\leq l}$ divide into easy and hard strings for length $l$.

C As in the proof of Theorem 5.2 (Wagner 1987).

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

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

Note, $f \in \text{FP}^{\Sigma^*_3[1]}$ and $f(x)$ is equal for all equal-length strings $x$.

We call $(1^l, \#)$ a hard pair if and only if $f(1^l) = 0$. $(1^l, y)$, $y \in \Sigma^*$, 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 5.2 (Wagner 1987) (with the obvious adoptions due to the changed definition of easy and hard strings) one hard pair gives a reduction from $\Sigma_3^p$ to a $\Sigma_3^p$ language.

**Claim E:** There exists a set $A \in \Sigma_3^p$ such that for all $x \in \Sigma^*$ and all $l \geq |x|$, if $(1^l, \omega)$ is a hard pair then

$$x \notin \Sigma_3^p \iff \langle x, 1^l, \omega \rangle \in A.$$  

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

If $f(1^l) = 0$ then $\omega = \#$ and for every string $y$, $|y| \leq l$, $y \in T_{\Sigma_3^p}$ if and only if $y$ is easy for length $l$. This holds in particular for $x$ itself. According to B we thus have

$$x \notin \Sigma_3^p \iff (\exists x_1 |x_1| \leq l)[h''((x_1, x)) \in \Sigma_3^p].$$

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

$$x \in \Sigma_3^p \iff h''((x, \omega)) \notin \Sigma_3^p.$$  

We define $A = \{\langle x, 1^l, \omega \rangle | (\omega = \# \land (\exists x_1 |x_1| \leq l)[h''((x_1, x)) \in \Sigma_3^p]) \lor (\omega \in \Sigma^* \land h''((x, \omega)) \in L_{\Sigma_3^p})\}$. It is not hard to verify that $A \in \Sigma_3^p$. This completes the proof of Claim E.

F In contrast to F in the proof of Theorem 5.2 (Wagner 1987), the new definition of easy and hard strings yields that one hard pair for sufficiently large length suffices to reduce $\Sigma_3^p$ to a $\Sigma_3^p$ language.
Claim F: There exist a set $B \in \Sigma_3^p$ 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
\[ x \in L_{\Sigma_3^p} \iff \langle x, 1^l, \omega \rangle \in B. \]

Let $x \in \Sigma^*$. By definition of $L_{\Sigma_3^p}$ we have
\[ x \in L_{\Sigma_3^p} \iff (\exists y \mid |y| \leq p_4(|x|))[(x, y) \notin L_{\Sigma_3^p}]. \]

According to Claim E there exists a set $A \in \Sigma_3^p$ such that if $\langle 1^l, \omega \rangle$, $l \geq s(p_4(|x|))$, is a hard pair then for all $y$, $|y| \leq p_4(|x|)$,
\[ \langle x, y \rangle \notin L_{\Sigma_3^p} \iff \langle \langle x, y \rangle, 1^l, \omega \rangle \in A, \]

and hence
\[ x \in L_{\Sigma_3^p} \iff (\exists y \mid |y| \leq p_4(|x|))[(\langle x, y \rangle, 1^l, \omega \rangle \in A). \]

Let $q$ be a polynomial such that $q(n) \geq s(p_4(n))$ for all $n$. Define $B = \{\langle x, 1^l, \omega \rangle \mid (\exists y \mid |y| \leq p_4(|x|))[(\langle x, y \rangle, 1^l, \omega \rangle \in A]\}$ and note that $B \in \Sigma_3^p$. This proves Claim F.

Claim G: There exist a set $C \in \Sigma_3^p$ 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
\[ x \in L_{\Sigma_3^p} \iff \langle x, \omega_1, \omega_2 \rangle \in C. \]

Let $x \in \Sigma^*$. We have
\[ x \in L_{\Sigma_3^p} \iff (\exists y \mid |y| \leq p_5(|x|))[(x, y) \notin L_{\Sigma_3^p}]. \]

According to Claim F there exist a set $B \in \Sigma_3^p$ and a polynomial $q$ such that if $\langle 1^l, \omega_1 \rangle$, $l \geq q(s(p_5(|x|)))$, is a hard pair then
\[ x \in L_{\Sigma_3^p} \iff (\exists y \mid |y| \leq p_5(|x|))[(\langle x, y \rangle, 1^l, \omega_1 \rangle \notin B]. \]

Let $q_1$ be a polynomial such that $q_1(n) \geq q(s(p_5(n)))$ for all $n$. Define
\[ D = \{\langle x, 1^l, \omega_1 \rangle \mid (\exists y \mid |y| \leq p_5(|x|))[(\langle x, y \rangle, 1^l, \omega_1 \rangle \notin B]\}. \]

Note that $D \in \Sigma_3^p$ and let $g$ be a many-one reduction from $D$ to $L_{\Sigma_4^p}$. Let $\tilde{q}$ be a polynomial such that for all $z \in \Sigma^*$, $|g(z)| \leq \tilde{q}(|z|)$. Hence we have that if $\langle 1^{q_1(|x|)}, \omega_1 \rangle$ is a hard pair then
\[ x \in L_{\Sigma_3^p} \iff g(\langle x, 1^{q_1(|x|)}, \omega_1 \rangle) \in L_{\Sigma_4^p}. \]

Applying Claim F again we obtain that if $\langle 1^l, \omega_2 \rangle$, $l \geq q(\tilde{q}(s(q_1(|x|))))$, is a hard pair and $|\omega_1| \leq q_1(|x|)$,
\[ g(\langle x, 1^{q_1(|x|)}, \omega_1 \rangle) \in L_{\Sigma_4^p} \iff g(\langle x, 1^{q_1(|x|)}, \omega_1 \rangle), 1^l, \omega_2 \rangle \in B. \]
All together, if \( \langle p_1([x]), \omega_1 \rangle \), and \( \langle 1, \omega_2 \rangle \), \( l \geq q(\hat{q}(s(q_1([x])))) \), are hard pairs then
\[
x \in L_{\Sigma_5^p} \iff \langle g((x, \hat{q}(p_1([x])), \omega_1)), 1^l, \omega_2 \rangle \in B.
\]

Let \( q_2 \) be a polynomial such that \( q_2(n) \geq q(\hat{q}(s(q_1(n)))) \) for all \( n \). Define \( C = \{ \langle x, \omega_1, \omega_2 \rangle \mid \langle g((x, \hat{q}(p_1([x])), \omega_1), 1^l, \omega_2 \rangle \in B \} \). This completes the proof of Claim G.\(^2\)

**Claim H:** \( L_{\Sigma_5^p} \in \text{P}^{\Sigma_5^p}[3] \).

Let \( C \in \Sigma_5^p \) and \( q_1 \) and \( q_2 \) be polynomials as defined in G. We give a \( \text{P}^{\Sigma_5^p}[3] \) algorithm for \( L_{\Sigma_5^p} \).

1. On input \( x \) compute \( f(\hat{q}(p_1([x]))) \) and \( f(\hat{q}(p_1([x]))) \). This amounts to two \( \Sigma_1^p \) queries.
2. (a) If \( f(\hat{q}(p_1([x]))) = 0 \) set \( \omega_1 = \# \). If \( f(\hat{q}(p_1([x]))) = 1 \) guess \( \omega_1, |\omega_1| = q_1([x]) \), and verify that \( \omega_1 \) is a hard string for length \( q_1([x]) \). Continue if this verification succeeds, otherwise reject.
   (b) If \( f(\hat{q}(p_1([x]))) = 0 \) set \( \omega_2 = \# \). If \( f(\hat{q}(p_1([x]))) = 1 \) guess \( \omega_2, |\omega_2| = q_2([x]) \), and verify that \( \omega_2 \) is a hard string for length \( q_2([x]) \). Continue if this verification succeeds, otherwise reject.
   (c) Verify that \( \langle x, \omega_1, \omega_2 \rangle \in C \).
   Note that all this can be done with a single \( \Sigma_1^p \) oracle query when handing \( x, f(\hat{q}(p_1([x]))) \),
   and \( f(\hat{q}(p_1([x]))) \) over to the oracle.
3. Accept if and only if the query under 2. returns “yes.”

The correctness of this algorithm is obvious, in particular recall from H of the proof of Theorem 5.2 (Wagner 1987) that the use of different \( \Sigma_5^p \) oracles does no harm to the algorithm.

**End of Proof of Main Claim**

---

4) Chang/Kadin 1989 [CK89,CK96]

**Theorem 5.4** If \( \text{P}^{\Sigma_5^p}[1] = \text{P}^{\Sigma_5^p}[2] \) then \( \text{PH} = \text{D} \cdot \Sigma_4^p \).

**Proof:**

**Main Claim** If \( \text{D} \cdot \Sigma_3^p = \text{coD} \cdot \Sigma_3^p \) then \( \text{PH} = \text{D} \cdot \Sigma_4^p \).

Obviously, it suffices to prove the main claim.

**Proof of Main Claim:**

\(^2\) The reader will observe that in light of the algorithm given in H below a reduction of \( L_{\Sigma_5^p} \) to the \( \Sigma_5^p \) language (as it was done in the proof of Theorem 6.2 (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 \( L_{\Sigma_5^p} \in \text{P}^{\Sigma_5^p}[2] \). However, the possibility of reducing \( L_{\Sigma_5^p} \) to a \( \Sigma_3^p \) language first appeared in [Wag87] and was crucially used in [CK89,CK96] and [BCO99,BCO93].
A As in the proof of Theorem 5.1 (Kadin 1987).

B, C, D, E, F, and G As in the proof of Theorem 5.2 (Wagner 1989).

H In contrast to G one hard pair suffices to reduce a $\Pi^P_3$ complete language to a $\Sigma^P_4$ language.

Claim H: There exist a set $D_1 \in \Sigma^P_4$ and a polynomial $p_1$ such that for all $x \in \Sigma^*$ and all

$$l \geq p_1(|x|)$$

if $\langle 1^l, \omega \rangle$ is a hard pair then

$$x \in L_{\Sigma^P_5} \iff \langle x, 1^l, \omega \rangle \in D_1.$$

Claim H is the analogue of G in the proof of Theorem 5.2 (Wagner 1987) with the modifications induced by the different hard strings definition and is implicitly contained in G of the proof of Theorem 5.3 (Wagner 1989). In particular, setting $p_1 = q_1$ and $D_1 = D$, where $q_1$ and $D$ are as defined in G of the proof of Theorem 5.3 (Wagner 1989) proves the claim.

I Define $S = \{1^l \mid f(1^l) = 1\}$. Though $f \in \text{FP}^{\Sigma^P_4}[1]$, testing whether $f(1^l) = 1$ can be done with a $\Sigma^P_4$ algorithm, as it is just testing whether there exists a hard string for length $l$. So $S \in \Sigma^P_4$.

Claim I: There exist a set $T \in \Sigma^P_3$ and a polynomial $p_4$ such that for all $l \in \mathbb{N}$ and all $l' \geq p_4(l)$, 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 $\Pi^P_3$ complete language to a $\Sigma^P_4$ language with the help of just one hard pair.

Claim J: There exist a set $D_2 \in \Sigma^P_4$ and a polynomial $p_2$ such that for all $x \in \Sigma^*$ and all $l \geq p_2(|x|)$, if $\langle 1^l, \omega \rangle$ is a hard pair then

$$x \notin L_{\Sigma^P_5} \iff \langle x, 1^l, \omega \rangle \in D_2.$$

The reader will soon observe that Claim J is the key trick in the current proof.

Let $p_4$, $q_1$, and $q_2$ be polynomials and $C \in \Sigma^P_3$ as defined in G and I. Let $p_2$ be a polynomial such that $p_2(n) \geq p_4(q_2(n))$ for all $n$. $D_2$ is defined by the following $\Sigma^P_4$ 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 $l \geq p_4(q_1(|x|))$ we determine $f(1^{q_1(|x|)})$ by applying Claim I. This is done as follows: Test using a $\Sigma^P_3$ 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 $l \geq p_4(q_2(|x|))$ we determine $f(1^{q_2(|x|)})$ by applying Claim I. This is done similar to step 2: Test using a $\Sigma^P_3$ 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 $\omega_1$, $|\omega_1| \leq q_1(|x|)$, verify that $\omega_1$ 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 \( \omega_2, |\omega_2| \leq q_2(|x|) \), verify that \( \omega_2 \) 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 \( (1^{q_1(|x|)}, \omega_1) \) and \( (1^{q_2(|x|)}, \omega_2) \) are hard pairs we determine whether \( x \not\in L_{\Sigma^p_{\mathbb{G}}} \) using Claim G. In other words, accept if and only if \( \langle x, \omega_1, \omega_2 \rangle \not\in C \).

Observe that if \( \langle 1^t, \omega \rangle \) is a hard pair and \( l \geq p_l(q_1(|x|)) \), step 2 indeed yields \( j_1 = f(1^{q_1(|x|)}) \) according to I. Similarly, if \( \langle 1^t, \omega \rangle \) is a hard pair and \( l \geq p_l(q_2(|x|)) \), step 3 correctly determines \( j_2 = f(1^{q_2(|x|)}) \) according to I. Furthermore, if \( (1^{q_1(|x|)}, \omega_1) \) and \( (1^{q_2(|x|)}, \omega_2) \) are hard pairs then we accept in step 6 if and only if \( x \not\in L_{\Sigma^p_{\mathbb{G}}} \). 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 correctness of steps 2 and 3. Hence setting \( p_2(n) \geq p_l(q_2(n)) \) 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 \( D \cdot \Sigma_{4}^p \) yields

\textbf{Claim K:} \( L_{\Sigma^p_{\mathbb{G}}} \subseteq \Sigma^p_{\mathbb{G}} \cdot \Sigma^p_{\mathbb{G}} \).

Let the sets \( D_1, D_2 \subseteq \Sigma^p_{\mathbb{G}} \) and the polynomials \( p_1 \) and \( p_2 \) be as defined in H and J. Let \( p \) be a polynomial such that \( p(n) \geq \max\{p_1(n), p_2(n)\} \) for all \( n \). Define

\[ E_1 = \{ x | \langle x, 1^{p(|x|)}, \# \rangle \in D_1 \}, \]

\[ E_2 = \{ x | (\exists \omega \in \Sigma^*)[\omega \text{ is a hard string for length } p(|x|)] \land \langle x, 1^{p(|x|)}, \omega \rangle \in D_1 \}, \]

and

\[ E_3 = \{ x | (\exists \omega \in \Sigma^*)[\omega \text{ is a hard string for length } p(|x|)] \land \langle x, 1^{p(|x|)}, \omega \rangle \in D_2 \}. \]

Clearly, \( E_1, E_2, E_3 \subseteq \Sigma^p_{\mathbb{G}} \). Since \( \Sigma^p_{\mathbb{G}} \) is closed under union we also have \( E_1 \cup E_2 \subseteq \Sigma^p_{\mathbb{G}} \). Hence \( (E_1 \cup E_2) \setminus E_3 \subseteq D \cdot \Sigma^p_{\mathbb{G}} \). We show \( L_{\Sigma^p_{\mathbb{G}}} = (E_1 \cup E_2) \setminus E_3 \). To see this consider the following case distinction. Let \( x \in \Sigma^* \).

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

Hence \( x \not\in E_2 \) and \( x \not\in E_3 \). Furthermore, \( \langle 1^{p(|x|)}, \# \rangle \) is a hard pair and hence according to Claim H,

\[ x \in L_{\Sigma^p_{\mathbb{G}}} \iff \langle x, 1^{p(|x|)}, \# \rangle \in D_1 \]

\[ \iff x \in E_1 \]

\[ \iff x \in (E_1 \cup E_2) \setminus E_3. \]

\textbf{Case 2} \( f(1^{p(|x|)}) = 1 \).

Hence there exist hard strings for length \( p(|x|) \). If \( x \in L_{\Sigma^p_{\mathbb{G}}} \) then clearly \( \langle x, 1^{p(|x|)}, \omega \rangle \in D_1 \) and \( \langle x, 1^{p(|x|)}, \omega \rangle \not\in D_2 \) for all hard strings \( \omega \) for length \( p(|x|) \), according to Claims H and J. Hence \( x \in (E_1 \cup E_2) \setminus E_3 \). If \( x \not\in L_{\Sigma^p_{\mathbb{G}}} \) then \( \langle x, 1^{p(|x|)}, \omega \rangle \not\in D_1 \) and \( \langle x, 1^{p(|x|)}, \omega \rangle \in D_2 \) for all hard strings \( \omega \) for length \( p(|x|) \), according to Claims H and J. Independent of whether \( x \in E_1 \) or \( x \not\in E_1 \) we have \( x \not\in (E_1 \cup E_2) \setminus E_3 \).

\textbf{L} We have shown \( L_{\Sigma^p_{\mathbb{G}}} \subseteq D \cdot \Sigma^p_{\mathbb{G}} \) and thus \( \Sigma^p_{\mathbb{G}} = D \cdot \Sigma^p_{\mathbb{G}} \) which immediately implies \( \text{PH} = D \cdot \Sigma^p_{\mathbb{G}} \).

\textit{End of Proof of Main Claim}
Theorem 5.5 If $P^{\Sigma^p_3[1]} = P^{\Sigma^p_3[2]}$ then $PH = (P^{NP})^{\Sigma^p_3}$.

Proof:

Main Claim If $D \cdot \Sigma^p_3 = \text{co}D \cdot \Sigma^p_3$ then $PH = (P^{NP})^{\Sigma^p_3}$,

where $(P^{NP})^{\Sigma^p_3}$ is the class of languages accepted by some DPTM making at most one query to a $NP^{\Sigma^p_3} = \Sigma^p_3$ oracle and polynomially many queries to a $\Sigma^p_3$ oracle.

The theorem follows immediately from the main claim, which we will prove now.

Proof of Main Claim:

A As in the proof of Theorem 5.1 (Kadin 1987).

B,C,D,E,F and G As in the proof of Theorem 5.3 (Wagner 1989).

H Quite similar to H of the proof of Theorem 5.4 (Chang/Kadin 1989) one hard pair suffices to reduce a $P^{\Sigma^p_3}$ language to a $P^{\Sigma^p_3}$ language.

Claim H: Let $L \in P^{\Sigma^p_3}$. There exist a set $D \in P^{\Sigma^p_3}$ and a polynomial $p$ such that for all $x \in \Sigma^*$, if $\langle 1^{p(|x|)}, \omega \rangle$ is a hard pair then

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

The proof is a straightforward application of F. Let $L \in P^{\Sigma^p_3}$, hence $L = L(N_1^{L_{\Sigma^p_3}})$ for some DPTM $N_1$ running in time $\widetilde{p}$ for some polynomial $p$. According to F there exist a language $B \in \Sigma^p_3$ 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

$$x \in L^{\Sigma^p_3} \iff \langle x, 1^l, \omega \rangle \in B.$$

We use this to reduce $L$ to a $P^{\Sigma^p_3}$ language with the help of one hard pair. Let $p$ be a polynomial such that $p(n) \geq q(\widetilde{p}(n))$ for all $n$. Define the DPTM $N_2$ as follows: $N_2^{L_{\Sigma^p_3}}(x, \omega)$ simulates the work of $N_1^{L_{\Sigma^p_3}}(x)$ but replaces every query $v$ to $L_{\Sigma^p_3}$ by a query $\langle v, 1^{p(|x|)}, \omega \rangle$ to $B$. Let $D = L(N_2^{L_{\Sigma^p_3}})$. Clearly, $D \in P^{\Sigma^p_3}$. It is not hard to verify that this proves the claim.

I Claim I: $L^{\Sigma^p_3} \in (P^{NP})^{\Sigma^p_3}$.

According to Claim G there exist a language $C \in \Sigma^p_3$ 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

$$x \in L^{\Sigma^p_3} \iff \langle x, \omega_1, \omega_2 \rangle \in C.$$
1. On input $x$, determine whether $\langle x, \# \rangle \in C$. This can be done with one query to $C$.

2. (a) Determine whether
   
   i. there exists a hard string $\omega_1$ 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 $\omega_2$ 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 $\omega_1$ and $\omega_2$ 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) Determine whether there exist two hard strings $\omega_1$ and $\omega_2$ for length $q_1(|x|)$ and $q_2(|x|)$, respectively, such that

   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 $\chi_C(\langle x, \#, \omega_2 \rangle) \neq \chi_C(\langle x, \#, \# \rangle)$.

Note that all this can be done with two parallel queries to a $\Sigma_4^p$ oracle, one query for (a) and one for (b).

3. Accept if and only if the three queries from 1., 2.(a), and 2.(b) return in this order the answers “yes,no,no,” “no,yes,no,” or “yes,yes,yes.”

The correctness of this algorithm follows immediately from the construction. As already pointed out in $H$ in the proof of Theorem 5.22 (Wagner 1987) the use of different $\Sigma^p_4$ oracles in step 2 does not affect the correctness of our algorithm. Note that step 2.(a) corresponds to checking whether the existence of hard strings causes at least one mind change, whereas step 2.(b) corresponds to determining whether the existence of hard strings causes two mind changes.

### Claim J

In order to prove the **Main Claim** one somehow has to show that the $(P^{NP}_{1-tt})^{\Sigma_3^p}$ algorithm for $L_{\Sigma_3^p}$ as given in $I$ can be improved to a $(P^{NP}_{1-tt})^{\Sigma_3}$ algorithm. This is achieved by exploiting Claim $H$.

**Claim J**: $P^{\Sigma_3^p} \subseteq (P^{NP}_{1-tt})^{\Sigma_3}$.

Let $L \in P^{\Sigma_2^p}$. Let $D \in P^{\Sigma_5^p}$ and $p$ be a polynomial as defined in $H$. We describe a $(P^{NP}_{1-tt})^{\Sigma_3}$ algorithm for $L$ which uses the same idea as in $I$.

1. On input $x$ determine whether $\langle x, \# \rangle \in D$. This can be done with the help of queries to a $\Sigma_5^p$ oracle, since $D \in P^{\Sigma_5^p}$.

2. Check whether there exists a hard string $\omega$ for length $p(|x|)$ such that $\chi_D(\langle x, \omega \rangle) \neq \chi_D(\langle x, \# \rangle)$. This can be done with one query to a $\Sigma_4$ oracle.

3. Accept if and only if the two queries return different answers.

The correctness of this algorithm follows immediately from the construction. Note that step 2 corresponds to determining whether the existence of a hard string causes a mind change.

### Claim K

Since $\Sigma_5^p \subseteq (P^{NP}_{1-tt})^{\Sigma_3}$ (Claim I), $(P^{NP}_{2-tt})^{\Sigma_3} \subseteq P^{\Sigma_3^p}$, and $P^{\Sigma_3^p} \subseteq (P^{NP}_{1-tt})^{\Sigma_3}$ (Claim J) we have proven $\Sigma_5^p \subseteq (P^{NP}_{1-tt})^{\Sigma_3}$ and thus $PH = (P^{NP}_{1-tt})^{\Sigma_3}$.

**End of Proof of Main Claim**
Theorem 5.6 If \( P^{\Sigma_3^P[1]} = P^{\Sigma_3^P[2]} \) then \( \text{PH} = \Sigma_3^P \).

Proof:

Main Claim: If \( P^{\Sigma_3^P} = P^{\Sigma_3^P} \) then \( \text{PH} = \Sigma_3^P \),

where \( P^{\Sigma_3^P} \) and \( P^{\Sigma_3^P} \) 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 \( \Sigma_3^P \) oracle.

Since \( P^{\Sigma_3^P[1]} \subseteq P^{\Sigma_3^P[1]} \subseteq P^{\Sigma_3^P[1]} \subseteq P^{\Sigma_3^P[2]} \) the theorem follows immediately from the above claim. Thus, it remains to prove the main claim.

Proof of Main Claim:

A Suppose \( P^{\Sigma_3^P} = P^{\Sigma_3^P} \). Let \( L_P \) and \( L_{P^{\Sigma_3^P[1]}} \) be many-one complete languages for \( P^{[1]} = \text{P} \) and \( P^{\Sigma_3^P[1]} \), respectively. Let \( L_{\Sigma_3^P} \) be a \( \Sigma_3^P \) complete language. In order to prove the Main Claim it suffices to give a \( \Sigma_3^P \) algorithm for \( L_{\Sigma_3^P} \).

B Define for any two sets \( A \) and \( B \), \( A \Delta B = \{ (x,y) \mid x \in A \iff y \notin B \} \).

Claim B: \( L_P \Delta L_{\Sigma_3^P} \) and \( L_{P^{\Sigma_3^P[1]}} \Delta L_{\Sigma_3^P} \) are many-one complete languages for \( P^{\Sigma_3^P} \) and \( P^{\Sigma_3^P} \), respectively.

We will only show the claim for \( L_{P^{\Sigma_3^P[1]}} \Delta L_{\Sigma_3^P} \). Obviously, \( L_{P^{\Sigma_3^P[1]}} \Delta L_{\Sigma_3^P} \in P^{\Sigma_3^P} \). Let \( L \) be an arbitrary language from \( P^{\Sigma_3^P} \). 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 \( A \in \text{NP} \) and \( B \in \Sigma_3^P \). Hence \( L = L(N(A,B)) \). Define

\[
C = \{ x \mid N(A,B)(x) \text{ accepts if } x_A \text{ is answered correctly and } x_B \text{ is answered } \text{“no”} \} \quad \text{and} \quad \n \]

\[
D = \{ x \mid N(A,B)(x) \text{ after answering the query } x_A \text{ correctly neither accepts nor rejects regardless of the answer to } x_B, \text{ and } x_B \in B \}.
\]

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, \( C \in P^{\Sigma_3^P} \) and \( D \in \Sigma_3^P \). Furthermore, it is not hard to verify that for all \( x \in \Sigma^* \),

\[
x \in L \iff \langle x, x \rangle \in C \Delta D.
\]

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

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

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

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

\[
\langle x_1, x_2 \rangle \in L_{P^{\Sigma_3^P[1]}} \Delta L_{\Sigma_3^P} \iff h(\langle x_1, x_2 \rangle) \in L_P \Delta L_{\Sigma_3^P}.
\]

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

\[
h(\langle x_1, x_2 \rangle) = h'(\langle x_1, x_2 \rangle), h''(\langle x_1, x_2 \rangle)
\]

and thus

\[
\langle x_1 \in L_{P^{\Sigma_3^P[1]} \iff x_2 \notin \Sigma_3^P \iff h'(\langle x_1, x_2 \rangle) \in L_P \iff h''(\langle x_1, x_2 \rangle) \notin \Sigma_3^P \rangle.
\]
D Let \( l \) be an integer. The string \( x_2 \) is said to be *easy for length* \( l \) if and only if \((\exists x_1 \mid |x_1| \leq l) [x_1 \in L_{P^{NP[1]}} \iff h'(⟨x_1, x_2⟩) \notin L_P] \) holds. x_2 is said to be *hard for length* \( l \) if and only if \((\forall x_1 \mid |x_1| \leq l) [x_1 \in L_{P^{NP[1]}} \iff h'(⟨x_1, x_2⟩) \in L_P] \) holds.

Thus, every string is either easy or hard for length \( l \). This observation will be used to divide the problem of giving a \( \Sigma^p_3 \) algorithm for \( L_{\Sigma^p_3} \) 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 \( \Sigma^p_2 \) 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 \( A \in \Sigma^p_3 \) and a polynomial \( q \) such that for all \( x \in \Sigma^* \), if \( x \) is hard for length \( q(|x|) \) then
\[
x \notin L_{\Sigma^p_3} \iff x \in A.
\]

Let \( p \) be a polynomial such that for all \( x \in \Sigma^* \),
\[
x \notin L_{\Sigma^p_3} \iff (\forall y \mid |y| \leq p(|x|)) (\exists z \mid |z| \leq p(|x|))[⟨x, y, z⟩ \in L_{P^{NP[1]}}].
\]

Recall that if \( x \) is a hard string for length \( l \), where \( l \) is some integer, then
\[
(\forall x_1 \mid |x_1| \leq l) [x_1 \in L_{P^{NP[1]}} \iff h'(⟨x_1, x⟩) \in L_P].
\]

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 \( y, z \in \Sigma^* \), \( |y|, |z| \leq p(|x|) \),
\[
⟨x, y, z⟩ \in L_{P^{NP[1]}} \iff h'(⟨x, y, z⟩, x)⟩ \in L_P
\]
and thus
\[
x \notin L_{\Sigma^p_3} \iff (\forall y \mid |y| \leq p(|x|)) (\exists z \mid |z| \leq p(|x|))[h'(⟨⟨x, y⟩, z⟩, x)⟩ \in L_P].
\]

Note that \( h'(⟨v, x⟩) \) is computable in time polynomial in \( \max\{ |v|, |x| \} \). Set \( A = \{ x \mid (\forall y \mid |y| \leq p(|x|)) (\exists z \mid |z| \leq p(|x|))[h'(⟨⟨x, y⟩, z⟩, x)⟩ \in L_P] \} \) and note that \( A \in \Sigma^p_3 \).

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 \( B \in \Sigma^p_3 \) such that for all \( x \in \Sigma^* \), if \( x \) is easy for length \( q(|x|) \) then
\[
x \notin L_{\Sigma^p_3} \iff x \in B.
\]

Define \( B = \{ x \mid (\exists x_1 \mid |x_1| \leq q(|x|)) [x_1 \in L_{P^{NP[1]}} \iff h'(⟨x_1, x⟩) \notin L_P] \wedge h''(⟨x_1, x⟩) \in L_{\Sigma^p_3} \} \).

Note that \( B \in \Sigma^p_3 \). 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 \( \Sigma^p_3 \) algorithm for \( \overline{L_{\Sigma^p_3}} \).
Claim G: $\overline{L_{\Sigma_3^P}} \in \Sigma_3^P$.

Let $A, B \in \Sigma_3^P$ 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 $\Sigma_3^P$ algorithm for $L_{\Sigma_3^P}$:

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 $\Sigma_2^P$ 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 $x \in B$.

As already pointed out in previous proofs, the use of different oracles in the above algorithm does not affect its correctness.

Since $\overline{L_{\Sigma_3^P}}$ is complete for $\Pi_3^P$ we have shown $\Pi_3^P \subseteq \Sigma_3^P$ and hence $PH = \Sigma_3^P$.

End of Proof of Main Claim

7) Hemaspaandra/Hemaspaandra/Hempel 1996 [HHH96a, HHH]

Theorem 5.7 If $P_{\Sigma_3^P}^{NP \in 2-tt} = P_{\Sigma_3^P}^{NP \in 3-tt}$ then $PH = \Sigma_3^P \Delta \Sigma_3^P$.

Proof:

Main Claim If $P_{\Pi_2 \Sigma_3^P}^{(P, \Sigma_3^P)} = P_{\Pi_2 \Sigma_3^P}^{(NP, \Sigma_3^P)}$ then $\Sigma_3^P = \text{coD} \cdot \Sigma_3^P$,

where $P_{\Pi_2 \Sigma_3^P}^{(P, \Sigma_3^P)}$ and $P_{\Pi_2 \Sigma_3^P}^{(NP, \Sigma_3^P)}$ 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 two queries to a $\Sigma_3^P$ oracle. Since $P_{\Sigma_3^P}^{NP \in 2-tt} \subseteq P_{\Pi_2 \Sigma_3^P}^{(P, \Sigma_3^P)} \subseteq P_{\Pi_2 \Sigma_3^P}^{(NP, \Sigma_3^P)} \subseteq P_{\Sigma_3^P}^{NP \in 3-tt}$, the theorem follows immediately from the above claim in light of Theorem 4.1. Thus, it suffices to show the correctness of the main claim.

Proof of Main Claim:

A Suppose $P_{\Pi_2 \Sigma_3^P}^{(P, \Sigma_3^P)} = P_{\Pi_2 \Sigma_3^P}^{(NP, \Sigma_3^P)}$. Let $L_P$ and $L_{P^{NP[1]}}$ be many-one complete languages for $P^{NP[1]} = P$ and $P^{NP[1]}$, respectively. Let $L_{D, \Sigma_3^P}$ be a $D \cdot \Sigma_3^P$ complete language. We will give a $D \cdot \Sigma_3^P$ algorithm for $L_{D, \Sigma_3^P}$. Let $L_1, L_2 \in \Sigma_3^P$ such that $L_{D, \Sigma_3^P} = L_1 - L_2$.

B Claim B: $L_P \Delta L_{D, \Sigma_3^P}$ and $L_{P^{NP[1]}} \Delta L_{D, \Sigma_3^P}$ are many-one complete for $P_{\Pi_2 \Sigma_3^P}^{(P, \Sigma_3^P)}$ and $P_{\Pi_2 \Sigma_3^P}^{(NP, \Sigma_3^P)}$, respectively.

The proof is similar to B in the proof of Theorem 5.4 (Hemaspaandra/Hemaspaandra/Hempel 1996). But here the $L_{D, \Sigma_3^P}$ part of $L_{P^{NP[1]}} \Delta L_{D, \Sigma_3^P}$ 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 $P_{\Pi_2 \Sigma_3^P}^{(P, \Sigma_3^P)} = P_{\Pi_2 \Sigma_3^P}^{(NP, \Sigma_3^P)}$ it follows that there is a many-one reduction $h$ between $L_{P^{NP[1]}} \Delta L_{D, \Sigma_3^P}$ and $L_P \Delta L_{D, \Sigma_3^P}$. While continuing as in C in the proof of Theorem 5.4 (Hemaspaandra/Hemaspaandra/Hempel 1996) and replacing $L_{\Sigma_3^P}$ by $L_{D, \Sigma_3^P}$ we obtain for all $x_1, x_2 \in \Sigma^*$,\[
(x_1 \in L_{P^{NP[1]}} \iff x_2 \notin L_{D, \Sigma_3^P} ) \iff (h'(\langle x_1, x_2 \rangle) \in L_P \iff h''(\langle x_1, x_2 \rangle) \notin L_{D, \Sigma_3^P}).\]
D As in the proof of Theorem 5.6 (Hemaspaandra/Hemaspaandra/Hempel 1996).

E As in the proof of Theorem 5.6 (Hemaspaandra/Hemaspaandra/Hempel 1996) we are first going to solve the sub-problem for the strings being hard for a certain length.

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

\[
x \notin L_{D, \Sigma^3_3} \iff x \in A_1 - A_2.
\]

A straightforward application of the key idea of E from the proof of Theorem 5.6 (Hemaspaandra/Hemaspaandra/Hempel 1996) leads to a \( D \cdot \Sigma^P_3 \) algorithm to test whether \( x \in L_{D, \Sigma^3_3} \) if \( x \) is a hard string. Without loss of generality let \( p_1 \) and \( p_2 \) be two polynomials such that for all \( x \in \Sigma^* \) and all \( i = 1, 2 \),

\[
x \in L_i \iff (\exists y \ | \ y \leq p_i(|x|))(\forall z \ | \ z \leq p_i(|x|))[h'((x, y, z), x) \in L_P].
\]

Let \( q \) be a polynomial such that \( q(n) \geq \max\{s(p_1(n)), s(p_2(n))\} \) for all \( n \). Let \( x \) be hard for length \( q(|x|) \). Hence we have for all \( x_1, \ |x_1| \leq q(|x|), \ x_1 \in L_{NP[1]} \iff h'((x_1, x)) \in L_P \). Hence for all \( i = 1, 2 \),

\[
x \in L_i \iff (\exists y \ | \ y \leq p_i(|x|))(\forall z \ | \ z \leq p_i(|x|))[h'((x, y, z), x) \in L_P].
\]

Define for \( i = 1, 2, \ A'_i = \{x \ | \ (\exists y \ | \ y \leq p_i(|x|))(\forall z \ | \ z \leq p_i(|x|))[h'((x, y, z), x) \in L_P]\} \). Note that \( A'_1, A'_2 \in \Sigma^P_3 \) and that for all \( x \in \Sigma^* \), if \( x \) is hard for length \( q(|x|) \) then

\[
x \in L_{D, \Sigma^3_3} \iff x \in A'_1 - A'_2.
\]

Since \( coD \cdot \Sigma^P_3 \subseteq \Sigma^{2P}[2] \subseteq D \cdot \Sigma^P_3 \) there exist sets \( A_1, A_2 \in \Sigma^P_3 \) such that \( A_1 - A_2 = A_1 - A_2 \). Hence, if \( x \) is a hard string for length \( q(|x|) \) then \( x \notin L_{D, \Sigma^3_3} \iff x \in A_1 - A_2 \).

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 \( B_1, B_2 \in \Sigma^P_3 \) such that for all \( x \in \Sigma^* \), if \( x \) is an easy string for length \( q(|x|) \) then

\[
x \notin L_{D, \Sigma^3_3} \iff x \in B_1 - B_2.
\]

Recall \( L_{D, \Sigma^3_3} = L_1 - L_2 \), where \( L_1, L_2 \in \Sigma^P_3 \). Define for \( i = 1, 2 \),

\[
B_i = \{x \ | \ (\exists x_1 \ |x_1| \leq q(|x|))[x_1 \in L_{NP[1]} \iff h'((x_1, x)) \notin L_P] \land \forall v <_{lex} x_1[v \in L_{NP[1]} \iff h'((v, x)) \in L_P] \land h''((x_1, x)) \in L_i\}.
\]

Obviously, \( B_1, B_2 \in \Sigma^P_3 \). 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

\[
x \notin L_{D, \Sigma^3_3} \iff x \in B_1 - B_2.
\]
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 \(D \cdot \Sigma_3^p\) algorithm for \(\overline{L_{D \cdot \Sigma_3^p}}\).

**Claim G:** \(\overline{L_{D \cdot \Sigma_3^p}} \in D \cdot \Sigma_3^p\).

Let \(A_1, A_2, B_1, B_2 \in \Sigma_3^p\) and \(q\) be a polynomial, all as defined in E and F.

For \(i = 1, 2\) let \(\hat{L}_i\) 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 \(\Sigma_2^p\) 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, \(\hat{L}_1, \hat{L}_2 \in \Sigma_3^p\). Furthermore, for all \(x \in \Sigma^*\), \(x \in \overline{L_{D \cdot \Sigma_3^p}} \iff x \in \hat{L}_1 - \hat{L}_2\) due to Claims E and F. Hence \(\overline{L_{D \cdot \Sigma_3^p}} \in D \cdot \Sigma_3^p\).

**End of Proof of Main Claim**

---

8) Buhrman/Fortnow 1996 [BF96]

**Theorem 5.8** If \(P^{\Sigma_2^p[1]} = P^{\Sigma_2^p[2]}\) then \(PH = \Sigma_2^p\).

**Proof:**

**Main Claim** If \(P^{\Sigma_2^p[1]} = NP^{\Delta \Sigma_2^p}\) then \(PH = \Sigma_2^p\),

where \(NP^{\Delta \Sigma_2^p} = \{A \Delta B \mid A \in NP \land B \in \Sigma_2^p\}\). Since \(P^{\Sigma_2^p[1]} \subseteq NP^{\Delta \Sigma_2^p} \subseteq P^{\Sigma_2^p[2]}\) 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 \(P^{\Sigma_2^p[1]} = NP^{\Delta \Sigma_2^p}\). Let \(L_P\) and \(L_{\Sigma_2^p}\) be complete languages for P and \(\Sigma_2^p\), respectively. 

\(L_P \overline{\Delta} L_{\Sigma_2^p}\) is complete for \(P^{(P, \Sigma_2^p)} = P^{\Sigma_2^p[1]}\). This can be shown quite analogous to B from the proof of Theorem 5.6 (Hemaspaandra/Hemaspaandra/Hempel 1996). Furthermore, observe that \(SAT \overline{\Delta} L_{\Sigma_2^p} \in NP^{\Delta \Sigma_2^p}\).

**B** Since \(P^{\Sigma_2^p[1]} = NP^{\Delta \Sigma_2^p}\) we have \(SAT \overline{\Delta} L_{\Sigma_2^p} \in P^{\Sigma_2^p[1]}\). Consequently there is a many-one reduction \(h\) from \(SAT \overline{\Delta} L_{\Sigma_2^p}\) to \(L_P \overline{\Delta} L_{\Sigma_2^p}\). Continuing as in C in the proof of Theorem 5.6 (Hemaspaandra/Hemaspaandra/Hempel 1996) while replacing \(L_{\Sigma_2^p}\) by \(L_{\Sigma_2^p}\) and \(L_{NP[1]}\) by \(SAT\) yields for all \(x_1, x_2 \in \Sigma^*\),

\[(x_1 \in SAT \iff x_2 \notin L_{\Sigma_2^p}) \iff (h'((x_1, x_2)) \in L_P \iff h''((x_1, x_2)) \notin L_{\Sigma_2^p}).\]

**C** As in D in the proof of Theorem 5.6 (Hemaspaandra/Hemaspaandra/Hempel 1996) but replace \(L_{NP[1]}\) by \(SAT\).
D Observe that the proof of Theorem 5.6 (Hemaspaandra/Hemasp aandra/Hempel 1996) is not valid for $\Sigma_p^p$ instead of $\Sigma_p^q$. The crucial point in the proof of Theorem 5.6 is the test in the final algorithm whether a string is easy or hard for a certain length (done by a $\Sigma_p^p$ oracle query). Since the final algorithm of this proof has to be a $\Sigma_p^p$ 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 $A \in \text{coNP}$ and a polynomial $q$ such that for all $x \in \Sigma^*$,

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

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

\[ x \notin L_{\Sigma_p^2} \iff (\forall z \ |z| \leq p(|x|))[\langle x, z \rangle \in \text{SAT}], \]

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,

\[ (* \quad (\forall x_1 \ |x_1| \leq q(|x|))[x_1 \in \text{SAT} \iff h'(\langle x_1, x \rangle) \in L_P]. \]

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 $\omega_F$ an assignment for $F$. Let the variables in $F$ be ordered. 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 $h'(\langle v_0, x \rangle) \in L_P$. If $h'(\langle v_0, x \rangle) \in L_P$ then under assumption (*) $F_0 \in \text{SAT}$. Thus, set the value for the first variable in $\omega_F$ to 0 and repeat this procedure with $F_0$ until $\omega_F$ assigns a value to each variable in $F$. If $h'(\langle v_0, x \rangle) \notin L_P$ 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 $\omega_F$ to 1 and repeat this procedure with $F_1$ until $\omega_F$ 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 $\omega_F$ 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

\[ x \notin L_{\Sigma_p^2} \iff x \in A. \]

But note that even if $x$ is not a hard string for length $q(|x|)$, $x \in A \implies x \notin L_{\Sigma_p^2}$. 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 5.6 (Hemaspaandra/Hemasp aandra/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.
Proof of Main Claim:

Main Claim

If \( G \) is many-one complete for \( \Pi^p_2 \), \( G \) is hard for length \( q(|x|) \). If \( G \) is easy for length \( q(|x|) \), \( G \) is complete for \( \Pi^p_2 \).

End of Proof of Main Claim

9) Hemaspaandra/Hemaspaandra/Hempel 1997 [HHH97]

Theorem 5.9 If \( P^{\Sigma^p_2}_{2-4t} = P^{\Sigma^p_2}_{3-4t} \) then \( PH = D \cdot \Sigma^p_2 \Delta \Sigma^p_3 \).

Main Claim If \( P^D \cdot \Sigma^p_2 = NP \Delta \Sigma^p_2 \) then \( D \cdot \Sigma^p_2 = coD \cdot \Sigma^p_2 \),

where \( P^D \Delta \Sigma^p_2 \) is defined similar to \( NP \Delta D \cdot \Sigma^p_2 \) as in Main Claim of the proof of Theorem 5.8 (Buhman/2096). Since \( P^{\Sigma^p_2}_{2-4t} \subseteq P^D \cdot \Sigma^p_2 \subseteq NP \Delta D \cdot \Sigma^p_2 \subseteq P^{\Sigma^p_2}_{3-4t} \) the theorem follows immediately from the above claim in light of Theorem 4.1. It remains to prove the main claim.

Proof of Main Claim:

A Suppose \( P^D \Delta \Sigma^p_2 = NP \Delta D \cdot \Sigma^p_2 \). Let \( L_P, L_{NP}, \) and \( L_{D \cdot \Sigma^p_2} \) be complete languages for \( P, NP, \) and \( D \cdot \Sigma^p_2 \), respectively. Let \( L_1, L_2 \in \Sigma^p_2 \) such that \( L_{D \cdot \Sigma^p_2} = L_1 - L_2 \).
The result of

As

D

Since 

C

Claim B: Let and are many-one complete for and respectively.

Thus the proof is straightforward and thus omitted.

C

As in the proof of Theorem 5.6 (Hemaspaandra/Hemaspaandra/Hempel 1996) but replace by . Thus for all , ,

\[ (x_1 \in L_{NP} \iff x_2 \notin L_{D, \Sigma^p_2}) \iff (h'(x_1, x_2)) \in L_P \iff h''(x_1, x_2) \notin L_{D, \Sigma^p_2}. \]

D

As in the proof of Theorem 5.6 (Hemaspaandra/Hemaspaandra/Hempel 1996) but replace by .

E

As of Theorem 5.8 (Buhrman/Fortnow 1996) but replace by .

F

The result of can be extended to yield a algorithm for the hard strings for length that is protected against accepting if the input string in fact is an easy string for length and .

Claim F: Let be the polynomial defined in E. There exists a set such that

1. 

2. for all , if is a hard string for length then

\[ x \notin L_{D, \Sigma^p_2} \iff x \in A'. \]

Let and be as defined in A and E. Define . Clearly, . Note that

\[ x \in A' \iff x \in A \lor x \in L_2 \]
\[ \iff x \in \overline{L_1} \lor x \in L_2 \]
\[ \iff x \in L_1 \cup L_2 \]
\[ \iff x \notin L_1 - L_2 \]
\[ \iff x \in \overline{L_{D, \Sigma^p_2}}. \]

Hence . Furthermore, in case is hard for length 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 being easy for length . Observe that a straightforward adaption of F from the proof of Theorem 5.6 (Hemaspaandra/Hemaspaandra/Hempel 1996) does not work here, since the sets constructed there would (even with the adaption to the current situation) remain sets, but one needs sets.

Claim G: Let be the polynomial defined in E. There exist sets such that

1. 

2. for all , if is an easy string for length then

\[ x \notin L_{D, \Sigma^p_2} \iff x \in B_1 - B_2. \]
Recall that $L_{D \cdot \Sigma_2^p} = L_1 - L_2$ where $L_1, L_2 \in \Sigma_2^p$. Without loss of generality let $L_1 \supseteq L_2$ [CGH+88]. Define for $i = 1, 2$,

$$B_i = \{ x | (\exists x_1 \mid x_1 \leq q(|x|))(x_1 \in L_{NP} \iff h'(\langle x_1, x \rangle) \not\in L_{P}) \land h''(\langle x_1, x \rangle) \in L_i \}.$$ 

Note that $B_1, B_2 \in \Sigma_2^p$ 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 $x \not\in L_1 - L_2$.

Let $x \in \Sigma^*$. Observe that for all $i = 1, 2$, $x \in B_1 - B_2$ implies $x$ is easy for length $q(|x|)$. So it suffices to show that if $x$ is easy for length $q(|x|)$ then $x \not\in L_{D \cdot \Sigma_2^p} \iff x \in B_1 - B_2$.

So 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

$$(\exists x_1 \mid x_1 \leq q(|x|)) [z_2 = h''((x_1, x)) \land (x_1 \in L_{NP} \iff h'(\langle x_1, x \rangle) \not\in L_{P}) \land (t > 0 \iff z_2 \in L_i)].$$

Such a string $z_2$ exists since $x$ is easy for length $q(|x|)$. Note that $x \not\in L_1 - L_2 \iff z_2 \in L_1 - L_2$. This follows from the definition of $z_2$ and the fact the equivalence

$$(x_1 \in L_{NP} \iff x \not\in L_{D \cdot \Sigma_2^p}) \iff (h'(\langle x_1, x \rangle) \in L_{P} \iff h''(\langle x_1, x \rangle) \not\in L_{D \cdot \Sigma_2^p})$$

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,

$$x \in \overline{L_{D \cdot \Sigma_2^p}} \iff x \not\in L_1 - L_2 \iff z_2 \in L_1 - L_2 \iff x \in B_1 - B_2.$$ 

H Combining Claims F and G while exploiting the structure of $D \cdot \Sigma_2^p$ shows

Claim H: $\overline{L_{D \cdot \Sigma_2^p}} \in D \cdot \Sigma_2^p$.

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

Suppose $x \in \overline{L_{D \cdot \Sigma_2^p}}$. Note that $x$ is 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 $x \not\in B_2$ according to Claim 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 $x \in \overline{L_{D \cdot \Sigma_2^p}}$ according to Claim F. If $x \not\in A'$ then $x \in B_1 - B_2$. But this implies $x \in \overline{L_{D \cdot \Sigma_2^p}}$ according to Claim G.

H Since $\overline{L_{D \cdot \Sigma_2^p}}$ is complete for $\text{coD} \cdot \Sigma_2^p$ we obtain $D \cdot \Sigma_2^p = \text{coD} \cdot \Sigma_2^p$.

End of Proof of Main Claim
6 Open Questions and Literature Starting Points

In our final section we would like to address some very interesting open questions related to the topic of this survey. Can one show

\[ P^{\Sigma_p^k[j]} = P^{\Sigma_p^k[j+1]} \implies PH = P^{\Sigma_p^k[j]} \]

for \( k \geq 1 \) and \( j \geq 2 \)? Note that the above claim for \( k > 2 \) and \( j = 1 \) has been proven in [HHH96b] whereas the claim for \( k = 2 \) and \( j = 1 \) first appeared in [BF96].

Can one prove a downward collapse result within the bounded-truth-table hierarchy and the boolean hierarchy over \( NP \)? In particular, for which, if any, \( j \) does it hold that

\[ p^{NP}_{j-\text{tt}} = p^{NP}_{(j+1)-\text{tt}} \implies \text{DIFF}_j(NP) = \text{coDIFF}_j(NP)? \]

Buhrman and Fortnow [BF96] give an oracle relative to which this fails for \( j = 1 \). But for \( j = 2, 3, \ldots \) not even that is known. Note that the \( NP \) case seems special; for \( \Sigma_p^k, k \geq 2 \), the relation holds for all \( j \) (by [HHH96b, BF96, HHH97]).

Hemaspaandra and Rothe [HR97] study boolean hierarchies over UP, and Bertoni et al. [BBJ+89] study boolean hierarchies over R. Do the techniques developed in the papers [HHH96b, BF96, HHH97] have any application to the unambiguous polynomial hierarchy, or to the “R hierarchy”?

Finally, if one wants to go into the actual journal literature, where should one start? Perhaps the best place in the original literature to get a first feel for the easy-hard technique in its initial form is Kadin’s paper [Kad88], which initiated the technique. Perhaps the best place in the original literature to get a feel for the easy-hard technique in the more powerful “downward collapse” form is the paper of Hemaspaandra, Hemaspaandra, and Hempel [HHH97], which initiated this form of the technique. Finally, for completeness, we mention that strongest currently known downward translations regarding the research line described in this survey are found in the final paper surveyed, [HHH97], except that very recently the key result of one subpart of that paper has itself been further extended in [HHH98].

References

[BBJ+89] A. Bertoni, D. Bruschi, D. Joseph, M. Sitharam, and P. Young. Generalized boolean hierarchies and boolean hierarchies over RP. In Proceedings of the 7th Conference on Fundamentals of Computation Theory, pages 35–46. Springer-Verlag Lecture Notes in Computer Science #380, August 1989.

[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(3):293–310, 1993.

[BF96] H. Buhrman and L. Fortnow. Two queries. Technical Report 96-20, Department of Computer Science, University of Chicago, Chicago, IL, September 1996.

[BF98] 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.
[CGH+88] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. S. Seuelsen, 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. S. Seuelsen, K. Wagner, and G. Wechsung. The boolean hierarchy II: Applications. *SIAM Journal on Computing*, 18(1):95–111, 1989.

[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.

[Hau14] F. Hausdorff. *Grundzüge der Mengenlehre*. Leipzig, 1914.

[HHH] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. A downward collapse within the polynomial hierarchy. *SIAM Journal on Computing*. To appear.

[HHH96a] 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.

[HHH96b] 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, Germany, June 1996.

[HHH97] 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.

[HHH98] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. Downward collapse from a weaker hypothesis. Technical Report TR-681, Department of Computer Science, University of Rochester, Rochester, NY, February 1998.

[HR97] L. Hemaspaandra and J. Rothe. Unambiguous computation: Boolean hierarchies and sparse Turing-complete sets. *SIAM Journal on Computing*, 26(3):634–653, 1997.

[Kad87] J. Kadin. The polynomial time hierarchy collapses if the boolean hierarchy collapses. Technical Report TR 87-843, Department of Computer Science, Cornell University, 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.

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

[LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–124, 1975.
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.

S. Reith and K. Wagner. On boolean lowness and boolean highness. Manuscript, January 1998.

L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

K. Wagner. A note on parallel queries and the symmetric-difference hierarchy. *Information Processing Letters*. To appear.

K. Wagner. Number-of-query hierarchies. Technical Report 158, Institut für Mathematik, Universität Augsburg, Augsburg, Germany, October 1987.

K. Wagner. Number-of-query hierarchies. Technical Report 4, Institut für Informatik, Universität Würzburg, Würzburg, Germany, February 1989.

C. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26:287–300, 1983.