MASTER'S THESIS

Vrije Universiteit in Amsterdam
Faculty of Sciences
Division of Mathematics and Computer Science
Department of Theoretical Computer Science

Stanislav Živný

Properties of oracle classes that collapse or separate complexity classes

Supervisor : dr. Václav Koubek, MFF UK
Second reader : dr. Femke van Raamsdonk, VU

Amsterdam 2005

Acknowledgements

First of all, I would like to thank my supervisor RNDr. Václav Koubek, DrSc. from Charles University in Prague for showing me the interesting world of structural complexity and all his patience he has had with me during all those years, especially in the last year during my stay in Amsterdam.

Further, I would like to thank my second reader dr. Femke van Raamsdonk from the Vrije Universiteit in Amsterdam for her support concerning my studies at the Vrije Universiteit and the whole special "One Year Master's Program".

During my second half-year term in Amsterdam, I had opportunity to work with Leen Torenvliet from Institute for Logic, Languages & Computation at University of Amsterdam. I thank him for all his support.

I am also grateful to Harry Buhrman from CWI (Centrum voor Wiskunde en Informaticae) in Amsterdam, who allowed me to visit CWI and attend their lectures.

Last no least, I would like to thank Petra Pivničková for her significant help with language corrections.

This thesis was also submitted at Faculty of Mathematics and Physics, Charles University in Prague under name "Relation between accepting languages and complexity of questions on oracle".

Amsterdam, July 28, 2005 Stanislav Živný

Contents

Chapter 1

Introduction

Computational complexity is a part of theoretical computer science which investigates the amount of resources (mostly time and space, but also for example nondeterminism, randomness, interaction and others) which is necessary to compute a solution of a problem. We are interested in two estimates, upper and lower bounds. An upper bound is given by any algorithm (Turing machine) which computes the solution of the problem. A lower bound is intrinsically much more difficult. It says something about all possible algorithms which solve the given problem.

A complexity class is the class of all problems which can be solved within given restrictions on some resource. The main goal of computational complexity is to investigate characteristics, properties and relationships between different complexity classes. The most difficult questions are about a collapse, i.e. an equality, and a separation, i.e. an inequality, of two classes.

An algorithm is considered to be effective if its time complexity is polynomial in the length of its input. The class of all problems which can be computed effectively, i.e. in polynomial time, is commonly denoted P. Another important class, NP, is the class of all problems which can be computed nondeterministically (Turing machine can "guess") in polynomial time. Hundreds of practical problems from real life were shown to belong to this class. The difference between the classes P and NP is the difference between verifying a solution (in case of P) and finding a solution (in case of NP). Intuitively, the latter one is more difficult. Nowadays, nobody can prove that. The $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem seems to be one of the most difficult problems in theoretical computer science.

While trying to tackle the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem, many new areas of computational complexity have emerged. One of the thoughts, relativization, was borrowed from recursion theory. If we cannot decide whether $\mathsf{P} = \mathsf{NP}$ or

$\mathsf{P} \neq \mathsf{NP}$, can we find some relativized world (oracle) such that in this world foregoing question can be answered?

Since the middle seventies, after the seminal paper on relativization [BGS75], many oracles have been constructed for many relationships between complexity classes. Consider two complexity classes, for example $\mathcal{C}$ and $\mathcal{D}$. A typical situation is that $\mathcal{C} \subseteq \mathcal{D}$, but it is a question whether $\mathcal{C} \subsetneq \mathcal{D}$ or $\mathcal{C} = \mathcal{D}$. We denote by $\mathcal{C}^A$ the relativized complexity class $\mathcal{C}$ relative to the oracle A. Construction of an oracle A such that $\mathcal{C}^A \neq \mathcal{D}^A$ is perceived as a difficulty of proving $\mathcal{C} = \mathcal{D}$. A possible proof of the unrelativized equality, i.e. $\mathcal{C} = \mathcal{D}$, would have to use some technique which does not relativize. Analogously, a construction of an oracle A such that $\mathcal{C}^A = \mathcal{D}^A$ is perceived as a difficulty of proving $\mathcal{C} \neq \mathcal{D}$.

This thesis

This thesis follows the idea of Balcázar [Bal84] to investigate properties of oracles relative to the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem.

We denote by $\mathcal{X}$ the class of sets relative to which $\mathsf{P} = \mathsf{NP}$ relativized, and by $\mathcal{Z}$ the class of sets relative to which $\mathsf{P} \neq \mathsf{NP}$. We investigate which problems belong to $\mathcal{X}$ and $\mathcal{Z}$ and what are the characteristics and structural properties of these classes. Of course, most of these questions are hard to answer. Any nontrivial information about $\mathcal{X}$ or $\mathcal{Z}$ (like f.e. $\emptyset \in \mathcal{X}$) would resolve the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem.

Chapter 2 introduces the necessary definitions and notations.

Chapter 3 surveys some related facts from the theory of relativization.

Chapter 4 investigates properties of the class $\mathcal{X}$. We present known properties about $\mathcal{X}$. We strengthen the known facts about complete problems for exponential classes to stronger classes. We show that some complete problems, if they ever exist, for deterministic classes between polynomial and exponential time do not belong to $\mathcal{X}$. We give an example of reduction under which there complete problems out of $\mathcal{X}$. We show that hard problems for exponential classes do not generally belong to $\mathcal{X}$. From that, we conclude that $\mathcal{X}$ is not closed under polynomial reductions neither upward nor downward. We characterize sets in $\mathcal{X}$ as the sets in the intersection of the first level of the extended low and the zeroth level of the extended high hierarchy. We show a closer connection between $\mathcal{X}$ and the extended hierarchies. Further, we prove that $\mathcal{X}$ is not closed under unions, intersections and symmetric differences. We mention results about sets reducible to $\mathcal{X}$ and other related topics.

Chapter 5 investigates properties of the class $\mathcal{Z}$. We show a close connection to the class $\mathcal{X}$. We prove that $\mathcal{Z}$ is not closed under unions, intersections and symmetric differences. We also prove that $\mathcal{Z}$ is not closed under disjoint unions. We conclude that disjoint union can lower complexity measured in terms of extended lowness.

Chapter 6 concludes this thesis and discusses further possible research.

The electronic version of this document can be found on the Internet at URL `http://standa.matfyz.cz/download/msthesis/`, you can also contact the author by e-mail `standa@matfyz.cz`.

Chapter 2

Preliminaries

In this chapter, we summarize basic definitions and explain the notation used in this thesis. Most of this chapter follows standard textbooks on structural complexity, [BDG88] and [BDG90]. We do not define everything in detail here, and we refer our kind reader to these books.

2.1 Numbers and functions

We work with functions on natural numbers. These functions are often used as time/space bounds. We often use the following "big O" notation. Let f be a function on natural numbers.

Definition 2.1

1. *$O(f)$ is the set of functions g such that for some $r > 0$ and for all but finitely many n, $g(n) \leq rf(n)$.*
2. *$o(f)$ is the set of functions g such that for every $r > 0$ and for all but finitely many n, $g(n) \leq rf(n)$.*

The inverse of a function f is denoted by f^{-1}. All logarithmic functions in this thesis have base 2. To simplify the text, $\log n$ always means $\lceil \log n \rceil$.

2.2 Alphabet, words, languages

Definition 2.2

1. *An* alphabet *is any non-empty, finite set. We use Σ to denote an alphabet. If not said otherwise, $\Sigma = \{0, 1\}$.*

2. *A* symbol*, or also a* letter *(or a* bit*, in case* $\Sigma = \{0, 1\}$*), is an element of an alphabet* Σ*.*
3. *Given an alphabet* Σ*, a* word*, or* string *over* Σ *is a finite sequence of symbols from* Σ*.*
4. *If* w *is a string over* Σ*, the* length *of* w*, written* $|w|$*, is the number of symbols that* w *contains.*
5. *The* empty word λ *is the unique word consisting of zero symbols.*
6. *Given a word* x *and a word* y*, the* concatenation *of* x *and* y*, written* xy*, is the word obtained by appending* y *to the end of* x*.*
7. *Given a word* w *over* Σ *and integer* n*, define* w^n *inductively by:* $w^0 = \lambda$*, and* $w^{n+1} = ww^n$ *for all* $n \geq 0$ *(we often consider the case when* $w = 0$*).*
8. *The set of all finite words over an alphabet* Σ *is denoted* Σ^**.*
9. *Given an alphabet* Σ*, the set of all words over* Σ *with length less than or equal to* n *is denoted* $\Sigma^{\leq n}$*, the set of all words over* Σ *with length less than* n *is denoted* $\Sigma^{<n}$*, and finally* $\Sigma^{=n} = \Sigma^{\leq n} \setminus \Sigma^{<n}$*.*
10. *Given an alphabet* Σ*, a* language*,* set*, or* oracle (set)*, over* Σ *is subset of* Σ^**.*
11. *Given a set* L*,* $|L|$ *denotes* cardinality *of* L*, i.e. the number of elements in* L*.*
12. *Given a set* L*,* $\chi_L(.)$ *is the characteristic function of* L*.*
13. *Given a set* L*,* exponential padding of L *is a set* $\{\langle x, 0^{f(|x|)}\rangle \mid x \in L \wedge f(n) \in O(2^{n^c})\}$ *for some constant* c*.*

We use standard, polynomial time computable, encoding of n-tuples $\langle x_1, x_2, \ldots, x_n\rangle$.

Given an alphabet Σ, we can order the symbols of Σ. In our case, $\Sigma = \{0, 1\}$, $0 < 1$. We suppose a standard lexicographical ordering on Σ^*: $a_1a_2 \cdots a_n \leq_{lex} b_1b_2 \cdots b_m$ if and only if either 1. $n < m$ or 2. $n = m$ and there is k, $1 \leq k \leq n$, such that $a_i = b_i$ for every $i \leq k - 1$ and $a_k < b_k$.

With our implicit alphabet, $\Sigma = \{0, 1\}$, we can identify words over Σ with natural numbers with zero: $0 = \lambda$, $1 = 0$, $2 = 1$, $3 = 00$, $4 = 01$, and so on (also called as enumeration of Σ^*. Generally, the n-th word over Σ in lexicographical ordering corresponds to the natural number n. Another possible correspondence is this one: A word $\alpha \in \Sigma^*$ corresponds to the natural number 1α in binary notation. In this thesis, we can use any of these two, for example the second one.

Definition 2.3 *Given languages A and B on Σ, define:*

1. *The* union $A \cup B$ *is the following language:*

$$A \cup B = \{w \mid w \in A \vee w \in B\}$$

2. *The* intersection $A \cap B$ *is the following language:*

$$A \cap B = \{w \mid w \in A \wedge w \in B\}$$

3. *The* difference *of A and B is:*

$$A \setminus B = \{x \mid x \in A \wedge x \notin B\}$$

4. *The* symmetric difference *of A and B is:*

$$A \triangle B = (A \setminus B) \cup (B \setminus A)$$

   Notice that $x \in A \triangle B$ means that x is in A or in B, but not in their intersection.

5. *The* complement *of a language L over Σ is the language $coL = \Sigma^* \setminus L$.*
6. *The* disjoint union*, or* join*, or even* marked union *of A and B is:*

$$A \oplus B = \{0w \mid w \in A\} \cup \{1w \mid w \in B\}$$

Definition 2.4 *A language is* tally *if and only if it is included in $\{a\}^*$ for some symbol a.* TALLY *denotes the class of all tally sets.*

Definition 2.5 *We say that a set A is* sparse *if and only if the number of words of length n is bounded a polynomial in n, i.e. if there exists a polynomial $p(n)$ such that $|\Sigma^{=n} \cap A| < p(n)$ for every n.* SPARSE *denotes the class of all sparse sets.*

Note that an equivalent definition of *sparse* set A bounds $\Sigma^{\leq n} \cap A$ by some polynomial $p(n)$.

We define something like complement on the classes of sets.

Definition 2.6 *Given a class $\mathcal{C}$ of sets, define $co\mathcal{C} = \{L \mid coL \in \mathcal{C}\}$.*

2.3 Computational model

As a computational model, we use a standard deterministic Turing machine with k working tapes. We also use its nondeterministic and oracle variants. We suppose that our reader is familiar with basic knowledge about Turing machines (computation, accepting/rejecting computation, configuration, computation tree of nondeterministic Turing machine, computation tree of Turing machine with an oracle, universal Turing machine, running time, working space, time/space constructible functions). Without loss of generality, we can suppose that the considered nondeterministic Turing machines have "nondeterministic fan-out 2". We consider "oracle" and "oracle set" as the same. For more details, see [BDG88].

Definition 2.7 *We denote by $L(M)$ the language accepted by the Turing machine M.*

Definition 2.8 *We denote by $L(M, A)$ the language accepted by the Turing machine M with the oracle A.*

Definition 2.9 *For any function $t(n) \geq n + 1$ and $s(n) \geq 1$, define:*

1. $\mathsf{DTIME}[t(n)]$ *is the class of all sets accepted by deterministic Turing machines whose running time is bounded above by $t(n)$.*
2. $\mathsf{NTIME}[t(n)]$ *is the class of all sets accepted by nondeterministic Turing machines whose running time is bounded above by $t(n)$.*
3. $\mathsf{DSPACE}[s(n)]$ *is the class of all sets accepted by deterministic Turing machines whose work space is bounded above by $s(n)$.*
4. $\mathsf{NSPACE}[s(n)]$ *is the class of all sets accepted by nondeterministic Turing machines whose work space is bounded above by $s(n)$.*

Note that we suppose $t(n) \geq n+1$. In this thesis, we always suppose that time bound is at least linear (Turing machine should at least read its input).

From the *Tape Compression Theorem* [BDG88], the class $\mathsf{DSPACE}[s(n)]$ ($\mathsf{NSPACE}[s(n)]$, respectively) is equivalent to $\mathsf{DSPACE}[s'(n)]$ ($\mathsf{NSPACE}[s'(n)]$, respectively) for every $s'(n) \in O(s(n))$.

From the *Linear Speed-up Theorem* [BDG88], it follows that $\mathsf{DTIME}[t'(n)] \subseteq \mathsf{DTIME}[t(n)]$ ($\mathsf{NTIME}[t'(n)] \subseteq \mathsf{NTIME}[t(n)]$, respectively) for every $t'(n) \in O(t(n))$ and $n \in o(t(n))$.

Theorem 2.10 (Deterministic Time Hierarchy Theorem) *Let t and t' be time bounds such that t' is time constructible and $t \log t \in o(t')$. Then $\mathsf{DTIME}[t']$ contains a language which is not in $\mathsf{DTIME}[t]$.*

Theorem 2.11 (Savitch Theorem) *If $s(n) \geq \log n$ is space constructible, then* $\mathsf{NSPACE}[s(n)]$ *is included in* $\mathsf{DSPACE}[s^2(n)]$.

2.4 Complexity classes and reductions

Definition 2.12 (Complexity classes)

$\mathsf{P} = \bigcup_{i\geq 0} \mathsf{DTIME}[n^i]$
$\mathsf{NP} = \bigcup_{i\geq 0} \mathsf{NTIME}[n^i]$
$\mathsf{PSPACE} = \bigcup_{i\geq 0} \mathsf{DSPACE}[n^i]$
$\mathsf{NPSPACE} = \bigcup_{i\geq 0} \mathsf{NSPACE}[n^i]$
$\mathsf{DEXT} = \bigcup_{c\geq 0} \mathsf{DTIME}[2^{cn}]$
$\mathsf{NEXT} = \bigcup_{c\geq 0} \mathsf{NTIME}[2^{cn}]$
$\mathsf{EXP} = \bigcup_{c\geq 0} \mathsf{DTIME}[2^{n^c}]$
$\mathsf{NEXP} = \bigcup_{c\geq 0} \mathsf{NTIME}[2^{n^c}]$
$\mathsf{DEXTSPACE} = \bigcup_{c\geq 0} \mathsf{DSPACE}[2^{cn}]$
$\mathsf{EXPSPACE} = \bigcup_{c\geq 0} \mathsf{DSPACE}[2^{n^c}]$

Definition 2.13 PF *is the class of (partial) functions that can be computed in polynomial time.*

Definition 2.14 (Many-one reduction) *Given two sets A_1 and A_2, we say that A_1 is* polynomial time many-one reducible *to A_2 if and only if there exists a total function $f : \Sigma^* \to \Sigma^*$, $f \in \mathsf{PF}$, such that $x \in A_1$ if and only if $f(x) \in A_2$ holds for all $x \in \Sigma^*$. We call such a function f* polynomial time many-one reduction.

Definition 2.15 (Turing reduction) *Given two sets A_1 and A_2, we say that A_1 is* polynomial time Turing reducible *to A_2 if and only if there exists a deterministic polynomial time Turing machine with an oracle such that $A_1 = L(M, A_2)$. We denote the fact that A_1 is Turing reducible to A_2 in polynomial time by $A_1 \leq_T^p A_2$ or $A_1 \in \mathsf{P}^{A_2}$.*

Definition 2.16 *Given a set A, the class P^A consists of all sets L such that there exists a deterministic polynomial time Turing machine M with an oracle such that $L = L(M, A)$.*

In a similar way, we get relativization of other complexity classes, PF^A, NP^A, PSPACE^A and so on (see [BDG88]).

Definition 2.17 (Polynomial time hierarchy) *The polynomial time hierarchy is the structure formed by the classes Σ_k^p, Π_k^p, Δ_k^p and Θ_k^p for each $k \geq 0$, where*

1. $\Sigma_0^p = \Pi_0^p = \Delta_0^p = \Theta_0^p = \mathsf{P}$.
2. $\Sigma_{k+1}^p = \mathsf{NP}^{\Sigma_k^p}$ *for* $k \geq 0$.
3. $\Pi_{k+1}^p = \mathsf{coNP}^{\Sigma_k^p}$ *for* $k \geq 0$.
4. $\Delta_{k+1}^p = \mathsf{P}^{\Sigma_k^p}$ *for* $k \geq 0$.
5. $\Theta_{k+1}^p = \mathsf{P}^{\Sigma_k^p[O(\log n)]}$ *for* $k \geq 0$.

Define also $\mathsf{PH} = \cup_{k\geq 0}\Sigma_k^p = \cup_{k\geq 0}\Pi_k^p = \cup_{k\geq 0}\Delta_k^p = \cup_{k\geq 0}\Theta_k^p$.

Θ_{k+1}^p levels of PH are from [HO02] and $\mathsf{P}^{A[O(\log n)]}$ means that the deterministic polynomial time Turing machine with the oracle A is allowed to query only $O(\log n)$ queries on input of length n.

We do not know whether either PH is infinite, i.e. $\Sigma_k^p \subsetneq \Sigma_{k+1}^p$ for every $k \geq 0$, or whether PH collapses to the k-th level, i.e. $\mathsf{PH} = \Sigma_k^p$. Note that $\mathsf{PH} = \mathsf{PSPACE}$ implies collapse of PH.

Definition 2.18 (Relativized polynomial time hierarchy) *The polynomial time hierarchy relative to an oracle A is the structure formed by the classes $\Sigma_k^{p,A}$, $\Pi_k^{p,A}$, $\Delta_k^{p,A}$ and $\Theta_k^{p,A}$ for each $k \geq 0$, where*

1. $\Sigma_0^{p,A} = \Pi_0^{p,A} = \Delta_0^{p,A} = \Theta_0^{p,A} = \mathsf{P}^A$.
2. $\Sigma_{k+1}^{p,A} = \mathsf{NP}^{\Sigma_k^{p,A}}$ *for* $k \geq 0$.
3. $\Pi_{k+1}^{p,A} = \mathsf{coNP}^{\Sigma_k^{p,A}}$ *for* $k \geq 0$.
4. $\Delta_{k+1}^{p,A} = \mathsf{P}^{\Sigma_k^{p,A}}$ *for* $k \geq 0$.
5. $\Theta_{k+1}^{p,A} = \mathsf{P}^{\Sigma_k^{p,A}[O(\log n)]}$ *for* $k \geq 0$.

Define also $\mathsf{PH}^A = \cup_{k\geq 0}\Sigma_k^{p,A} = \cup_{k\geq 0}\Pi_k^{p,A} = \cup_{k\geq 0}\Delta_k^{p,A} = \cup_{k\geq 0}\Theta_k^{p,A}$.

Definition 2.19 (Strong exponential hierarchy [Hem89]) *The strong exponential hierarchy is the the structure formed by the classes Σ_k^{SEH}, Π_k^{SEH} and Δ_k^{SEH} for $k \geq 0$, where*

1. $\Sigma_0^{\mathsf{SEH}} = \Pi_0^{\mathsf{SEH}} = \Delta_0^{\mathsf{SEH}} = \mathsf{DEXT}$.
2. $\Sigma_1^{\mathsf{SEH}} = \mathsf{NEXT}$, $\Pi_1^{\mathsf{SEH}} = \mathsf{coNEXT}$, $\Delta_1^{\mathsf{SEH}} = \mathsf{P}^{\mathsf{DEXT}} = \mathsf{EXP}$.
3. $\Sigma_{k+1}^{\mathsf{SEH}} = \mathsf{NP}^{\Sigma_k^{\mathsf{SEH}}}$ *for* $k \geq 1$.
4. $\Pi_{k+1}^{\mathsf{SEH}} = \mathsf{coNP}^{\Sigma_k^{\mathsf{SEH}}}$ *for* $k \geq 1$.
5. $\Delta_{k+1}^{\mathsf{SEH}} = \mathsf{P}^{\Sigma_k^{\mathsf{SEH}}}$ *for* $k \geq 1$.

Define also $\mathsf{SEH} = \cup_{k\geq 0}\Sigma_k^{\mathsf{SEH}} = \cup_k \Pi_k^{\mathsf{SEH}} = \cup_{k\geq 0}\Delta_k^{\mathsf{SEH}}$.

Note that we would get the same hierarchy if we replaced NEXT by NEXP. Hemachandra [Hem89] showed the collapse of the strong exponential hierarchy to the second level, $\mathsf{SEH} = \Delta_2^{\mathsf{SEH}}$. Note why the foregoing hierarchy is called "strong". The reason is the contrast with the (ordinary) exponential hierarchy, which it defined as $\mathsf{EH} = \mathsf{NEXT} \cup \mathsf{NEXT}^{\mathsf{NP}} \cup \mathsf{NEXT}^{\mathsf{NP}^{\mathsf{NP}}} \cdots$. EH is included in SEH, but there exist an oracle A such that SEH^A is not contained in EH^A. This is simply because from the second level up, EH can query strings in A of length 2^{cn} but SEH can query strings of length 2^{n^k}. The argument is the same as for showing $\mathsf{DEXT}^{\mathsf{P}} \subsetneq \mathsf{P}^{\mathsf{DEXT}}$, since $\mathsf{DEXT}^{\mathsf{P}} = \mathsf{DEXT}$ but $\mathsf{P}^{\mathsf{DEXT}} = \mathsf{EXP}$, and $\mathsf{DEXT} \subsetneq \mathsf{EXP}$ by the Time Hierarchy Theorem 2.10.

Definition 2.20 *Two languages $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$ are* p-isomorphic (polynomial time isomorphic) *if and only if there exists a bijection $f : \Sigma^* \to \Sigma^*$ such that $f \in \mathsf{PF}$, $f^{-1} \in \mathsf{PF}$ and f is a reduction from A to B.*

From now on till the end of this chapter, consider a and b as an arbitrary text strings for which $\leq_a^b$ is a reduction (we consider just two cases, $\leq_m^p$ and $\leq_T^p$). If b is omitted, then it implicitly means $b = p$, p for polynomial time. Analogously, when we talk about many-one or Turing reductions, and not said otherwise, we implicitly mean polynomial time reductions. If $A \leq_a^b B$ and $B \leq_a^b A$, then $A \equiv_a^b B$. If $A \not\leq_a^b B$ and $B \not\leq_a^b A$, then we call A and B *incomparable* under $\leq_a^b$-reductions.

Definition 2.21 ([HO02])

1. *Given a set C, define $R_a^b(C) = \{L \mid L \leq_a^b C\}$.*
2. *Given a class $\mathcal{C}$, define $R_a^b(\mathcal{C}) = \{L \mid (\exists C \in \mathcal{C})[L \leq_a^b C]\}$.*

Definition 2.22 *Given class $\mathcal{C}$, say*

1. *$\mathcal{C}$ is* closed upward *under $\leq_a^b$ if and only if $([A \leq_a^b B \wedge A \in \mathcal{C}] \Rightarrow B \in \mathcal{C})$.*
2. *$\mathcal{C}$ is* closed downward *with under $\leq_a^b$ if and only if $([A \leq_a^b B \wedge B \in \mathcal{C}] \Rightarrow A \in \mathcal{C})$.*

Note that class $\mathcal{C}$ is closed downward under $\leq_a^b$ if and only if $\mathcal{C} = R_a^b(\mathcal{C})$.

Note that most of the standard complexity classes (and all considered in this thesis) are closed under polynomial time reductions downward.

Remark 2.23 *Some reductions are powerful enough to bridge the difference between seemingly (or absolutely) different classes.*

1. $\mathsf{TALLY} \subsetneq \mathsf{SPARSE}$, *but* $R^p_T(\mathsf{TALLY}) = R^p_T(\mathsf{SPARSE})$ *(Hartmanis coding, see [BDG88]) and even* $R^p_{ctt}(\mathsf{TALLY}) = R^p_{ctt}(\mathsf{SPARSE})$ *(see [BHL95]), where* $\leq^p_{ctt}$ *denotes polynomial time conjunctive truth-table reduction.*
2. $\mathsf{DEXT} \subsetneq \mathsf{EXP}$, *but* $R^p_m(\mathsf{DEXT}) = R^p_m(\mathsf{EXP}) = \mathsf{EXP}$ *(via padding).*
3. *We suppose* $\mathsf{NP} \neq \mathsf{coNP}$, *but* $R^p_T(\mathsf{NP}) = R^p_T(\mathsf{coNP})$.

Definition 2.24 (The extended low hierarchy)
For every $n \geq 1$, *define:*

$$\mathsf{EL}_n = \{A \mid \Sigma^{p,A}_n \subseteq \Sigma^{p,A\oplus SAT}_{n-1}\}.$$

The extended low hierarchy *is* $\mathsf{ELH} = \cup_{n\geq 1}\mathsf{EL}_n$.

Definition 2.25 (The extended high hierarchy)
For every $n \geq 0$, *define:*

$$\mathsf{EH}_n = \{A \mid \Sigma^{p,A\oplus SAT}_n \subseteq \Sigma^{p,A}_n\}.$$

The extended high hierarchy *is* $\mathsf{EHH} = \cup_{n\geq 0}\mathsf{EH}_n$.

Immediately from the definitions of the extended hierarchies it follows that:

Fact 2.26

1. *For every* $n \geq 1$, $\mathsf{EL}_n \subseteq \mathsf{EL}_{n+1}$.
2. *For every* $n \geq 0$, $\mathsf{EH}_n \subseteq \mathsf{EH}_{n+1}$.

Fact 2.27 *For every recursive set* A, *there exists an effective enumeration* $\{M_i\}_{i\geq 1}$ *of deterministic Turing machines with oracles such that the running time of the machine* M_i *is bounded above by polynomial* p_i, *the machine* M_i *can be found in polynomial time in* i *and* $\mathsf{P}^A = \{L(M_i, A) \mid i \geq 1\}$. *Without loss of generality, we can assume that polynomial* p_i *is a form of* $n^i + i$ *for all* i *and for all* n, *and* $p_0(n) = 0$ *for all* n. *This guarantees that polynomial* p_i *is nondecreasing for all* i *and that* $p_i(n) \leq p_{i+1}(n)$ *for all* i *and for all* n. *In this thesis, we refer to this enumeration as* enumeration $\{M_i\}_{i\geq 1}$. *See [BDG88] for existence of this enumeration.*

Recall that it is possible to design a Turing machine, called a *universal machine*, which receives as an input the encoding of a machine M together with an input w to M, and which is able to simulate the computation of M on w. This fact is similar to the existence of "interpreters" for computer programming languages, i.e. programs that read and execute other programs.

Let U be a fixed universal machine.

Definition 2.28 (Resource-bounded Kolmogorov complexity)
The Kolmogorov time-bounded complexity *set* $\mathsf{K}[f,g]$ *is the set formed by the strings* u *such that there is a word* w*, of length* $|w| \leq f(|u|)$*, such that* $U(w) = u$ *and this result is obtained in at most* $g(|u|)$ *steps.*

Consider two relativizable complexity classes $\mathcal{C}$ and $\mathcal{D}$. We call the oracle A *positive* or *collapsing*, relative to the $\mathcal{C} \stackrel{?}{=} \mathcal{D}$ problem, if $\mathcal{C}^A = \mathcal{D}^A$. Similarly, we call the oracle A *negative* or *separating* if $\mathcal{C}^A \neq \mathcal{D}^A$. Most of the time, we use this for the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem.

2.5 Complete and hard problems

Definition 2.29 $K(A) = \{\langle M, x, 1^t\rangle \mid M$ *is a nondeterministic Turing machine that accepts* x *with the oracle* A *in at most* t *steps* $\}$.

Definition 2.30 $KS(A) = \{\langle M, x, 1^s\rangle \mid M$ *is a deterministic Turing machine that accepts* x *with the oracle* A *using an amount of space bounded above by* s $\}$.

Definition 2.31 (Hard problems)
Given a complexity class C*, define*

$$H_a^b = \{L \mid (\forall K \in \mathsf{C})[K \leq_a^b L]\}.$$

Definition 2.32 (Complete problems)
Given a complexity class C*, define*

$$C_a^b = \{L \mid L \in \mathsf{C} \wedge (\forall K \in \mathsf{C})[K \leq_a^b L]\}.$$

$K(A)$ is a standard complete problem for NP^A under $\leq_m^p$-reductions and $KS(A)$ is a standard complete problem for PSPACE^A under $\leq_m^p$-reductions. Recall that SAT, the set of satisfiable quantifier-free boolean formulas in the conjunctive normal form, is complete for NP under $\leq_m^p$-reductions. Recall that QBF, the set of quantified boolean formulas without free variables which evaluate to true, is complete for PSPACE under $\leq_m^p$-reductions.

Note that every level of the polynomial time hierarchy (and the relativized polynomial time hierarchy also) has complete problem under $\leq_m^p$-reductions.

Chapter 3

Relativization

This chapter surveys some facts from the theory of relativization, which are used later in this thesis. We present some basic theorems including the existence of an oracle relative to which $\mathsf{P} = \mathsf{NP}$, and the existence of an oracle relative to which $\mathsf{P} \neq \mathsf{NP}$. We also show the complexity of the latter one. Then we mention the Random oracle hypothesis. We present the concept of the positive relativization. We show a few examples of sets which provide a positive relativization of the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem. Sometimes, we just mention results and concepts without explicit definitions and explanations, but with references, where can be read more about the subject. The last section includes references to the related work.

3.1 Relativization

The $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem is one of the most important problems in computational complexity. In order to understand it better, one can try to solve a more general problem. The computational complexity has borrowed the relativization technique from recursion theory. In terms of Turing machines, by which complexity classes P and NP are defined, we add to machine access to an oracle. The oracle can be queried about membership of a given word in the oracle set. These queries are performed in constant time. We get a relativized version of the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem. For a given oracle A, $\mathsf{P}^A \stackrel{?}{=} \mathsf{NP}^A$.

The complexity theory concerns resource-bounded reductions. Many of the questions ask about the properties of these resource-bounded reductions. Relativization can be viewed as a question of deterministic and nondeterministic polynomial time Turing reductions. Do these two reductions differ? By definition, $A \leq_T B$ if and only if $A \in \mathsf{P}^B$, and $A \leq_T^{NP} B$ if and only if $A \in \mathsf{NP}^B$. Suppose for every set B, $\mathsf{P}^B = \mathsf{NP}^B$. Then for every A and B,

$A \leq_T B$ if and only if $A \leq_T^{NP} B$. Thus $A \leq_T B$ and $A \leq_T^{NP} B$ would be equivalent. Thus, for showing that these two reductions are different, it is sufficient to show the existence of a set B such that $\mathsf{P}^B \neq \mathsf{NP}^B$.

Baker, Gill and Solovay [BGS75] showed the existence of an oracle relative to which $\mathsf{P} \neq \mathsf{NP}$. Their seminal paper about relativization [BGS75] started several decades of an intensive research in relativization.

They also showed the existence of an oracle relative to which $\mathsf{P} = \mathsf{NP}$. This has led to many discussions about proof techniques and their chances to resolve the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem. Most of the known techniques can be relativized, i.e. hold with the presence of the oracle. These techniques can not solve the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem. A great many articles are devoted to discussions about usefulness and strength of relativization. See the references in the last section.

The relativization technique has been used in many areas of computational complexity. Not only for the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem, but also for many other complexity classes oracles have been constructed relative to which different relationships hold. For more, see [BDG90] and [HO02].

3.2 Separation

In this section, we show the existence of an oracle relative to which $\mathsf{P} = \mathsf{NP}$ and we show the existence of an oracle relative to which $\mathsf{P} \neq \mathsf{NP}$. We also show that the separating oracle belongs to DEXT.

First of all, we present some basic properties about $K(A)$, standard complete sets for NP^A under $\leq_m^p$-reductions.

Lemma 3.1 (folklore) *$K(A) \in C_m^p(\mathsf{NP}^A)$.*

Lemma 3.2 (folklore) *$\mathsf{P}^A = \mathsf{NP}^A$ if and only if $K(A) \in \mathsf{P}^A$.*

Now we present the existence of the *collapsing oracle*, i.e. the existence of the oracle relative to which $\mathsf{P} = \mathsf{NP}$.

Theorem 3.3 ([BGS75]) *There is an oracle A such that $\mathsf{P}^A = \mathsf{NP}^A$.*

Proof We construct inductively an oracle A such that $A = K(A)$ by the length of the strings. An empty string is not in A. Provided we have decided about all strings up to length k whether they belong to A or not, we now show how to decide about an arbitrary string of length $k+1$. Let z be an arbitrary string such that $|z| = k+1$. If z is not a form of $\langle M, x, 1^n \rangle$, then z does not belong to A. If z has the desired form $\langle M, x, 1^n \rangle$, then we simulate

machine M with the oracle A on input x for time $\leq n$. This simulation can be performed because the machine M in this computation does not query strings longer than n (it does not have time for it) and $n \leq k+1$. Therefore, M can query only for words shorter than its input and these words have been already decided. Then $z \in A$ if and only if M accepts x in at most n steps with the oracle A. □

We can prove more. Not only such an oracle exists, but complete sets for PSPACE under $\leq^p_m$-reductions accomplish a collapse of $\mathsf{P} = \mathsf{NP}$ relativized.

Theorem 3.4 ([BGS75]) *If $A \in C^p_m(\mathsf{PSPACE})$, then $\mathsf{P}^A = \mathsf{NP}^A$.*

Proof Given $A \in C^p_m(\mathsf{PSPACE})$ and using facts $\mathsf{PSPACE} = \mathsf{P}^A$ and $\mathsf{PSPACE}^{\mathsf{PSPACE}} = \mathsf{PSPACE}$ we have $\mathsf{PSPACE} = \mathsf{P}^A \subseteq \mathsf{NP}^A \subseteq \mathsf{PSPACE}^A \subseteq \mathsf{PSPACE}^{\mathsf{PSPACE}} = \mathsf{PSPACE}$. □

Now we prove the existence of the *separating oracle*, i.e. the existence of the oracle relative to which $\mathsf{P} \neq \mathsf{NP}$.

Definition 3.5 *For a language B, define $L(B) = \{0^n \mid (\exists x \in B)[|x| = n]\}$.*

Theorem 3.6 ([BGS75]) *There is an oracle B such that $\mathsf{P}^B \neq \mathsf{NP}^B$.*

Proof Clearly, $L(B) \in \mathsf{NP}^B$ for every B: For a given input x, $|x| = n$, guess y of length n and check whether $y \in B$ or not.

We construct B such that $L(B) \notin \mathsf{P}^B$ by diagonalization against machines from P^B.

Consider an enumeration $\{M_i\}_{i\geq 1}$ from Fact 2.27.

We construct B in stages. Denote by $k(n)$ an increasing sequence of natural numbers: $k(n)$ is the length of word which is used in the n-th stage to ensure that M_n does not accept $L(B)$. After n stages, we denote the so far constructed oracle set by B_{n-1}. In the n-th stage, we add at most one word of length $k(n)$ to ensure that $L(B) \neq L(M_n, B)$. We show that either $0^{k(n)} \in L(M_n, B)$ and $B \cap \Sigma^{=k(n)} = \emptyset$, or $0^{k(n)} \notin L(M_n, B)$ and $B \cap \Sigma^{=k(n)} \neq \emptyset$. Figure 3.1 describes the n-th stage in the construction of B. The final oracle $B = \cup_n B_n$.

From the condition on $k(n)$, an appropriate word $y(n)$ of length $k(n)$ always exists if needed. There are $2^{k(n)}$ words of length $k(n)$ and $p_n(k(n)) < 2^{k(n)}$.

The condition on $k(n)$ also ensures that $k(n)$ is long enough not to disturb, by possible adding of $y(n)$ to B, any computation from the previous stages. It means that $0^{k(n)} \in L(M_n, B_{n-1}) \Leftrightarrow 0^{k(n)} \in L(M_n, B)$. This holds for every n. Therefore, $0^{k(n)} \in L(B) \Leftrightarrow 0^{k(n)} \notin L(M_n, B)$ for every n. That means $L(B) \notin \mathsf{P}^B$. Since $L(B) \in \mathsf{NP}^B$, we obtain $\mathsf{P}^B \neq \mathsf{NP}^B$.

□

stage 0:
  $k(0) = 0$
  $B_0 = \emptyset$
stage n:
  Let $k(n)$ be the smallest natural number such that
    $k(n) > p_{n-1}(k(n-1))$ and $p_n(k(n)) < 2^{k(n)}$.
  If $0^{k(n)} \in L(M_n, B_{n-1})$ then $B_n = B_{n-1}$.
  If $0^{k(n)} \notin L(M_n, B_{n-1})$ then $B_n = B_{n-1} \cup \{y(n)\}$,
  where $y(n)$ is the first word, in lexicographical order, of length $k(n)$
  such that $y(n)$ is not queried in the computation of M_n on input $0^{k(n)}$
  with the oracle B_{n-1}.

Figure 3.1 Construction of B such that $L(B) \notin \mathsf{P}^B$.

Find(n):
  $i = 1$
  $k(0) = 0$
  $k(1) = 1$
  while $(k(i) < n)$ **do**
    $k(i) = p_{i-1}(k(i-1)) + 1$
    if $k(i) > n$ then **return** 0
    if $(k(i) = n)$ and $(k(i))^i + i < 2^{k(i)})$ then **return** i
    while $((k(i))^i + i \geq 2^{k(i)})$ **do**
      if $k(i) \geq n$ then **return** 0
      $k(i) = k(i) + 1$
    endwhile
    if $k(i) > n$ then **return** 0
    if $k(i) = n$ then **return** i
    if $k(i) < n$ then $i = i + 1$
  endwhile

Figure 3.2 Computation of stage i such that $k(i) = n$.

Inspecting the previous proof, we obtain an upper bound on time complexity of the separating oracle.

Theorem 3.7 *The oracle B from Theorem 3.6 belongs to* DEXT.

Proof We describe an algorithm for B which belongs to DEXT. Suppose input x of length n.

Recall that $p_i = n^i + i$ and $p_0(n) = 0$ for $i \geq 1$ and for all n.

First we need to find out whether there exists an l such that $k(l) = n$, where $k(.)$ is defined in the proof of Theorem 3.6. If such l does not exist, then $x \notin B$. Figure 3.2 presents the procedure Find, which for given argument n finds out whether there exists a stage i (command "**return** i", for $i > 0$) such that $k(i) = n$ (i.e. in the i-th stage, word 0^n is used for diagonalization), or finds out that such i does not exist (command "**return** 0").

The procedure Find works in linear time in n. We are looking for the first i such that $k(i) \geq n$ and because the sequence $k(.)$ is increasing, we have to evaluate $k(.)$ at most n times (note that in case $k(i) > n$ for some i, we can stop immediately when we know $k(i) > n$ and we do not have to evaluate $k(n)$).

If we find l such that $k(l) = n$ (and $p_l(n) < 2^n$), we need to simulate M_l on the input 0^n with the oracle B. If the simulation accepts 0^n, then any word of length n is not in B, and so $x \notin B$. If the simulation rejects 0^n, then $x \in B$ if and only if x is the first word, in lexicographical order, of length n which is not queried in the performed simulation.

Machine M_l works in time p_l, and the simulation can be performed in time $O(p_l(n))$. But we do not know the oracle B. For every $i = \mathsf{Find}(m) > 0$, $m = 1, \ldots, n$, we have to perform the simulation of M_i on input 0^m and store the information about the strings which belong to B.

Together, we have at most l simulations, $l \leq n$, of the polynomial time machines $M_1, M_2, \ldots, M_l$ working in polynomial time $p_1, p_2, \ldots, p_l$ on inputs of length $k(1), k(2), \ldots, k(l)$, where $k(i) \leq n$ for all $1 \leq i \leq l$. The foregoing estimate for the number of simulations, $l \leq n$, is not sufficient and we do a better one. We claim that $k(i+1) > 2k(i)$ for all $i \geq 1$. From the construction of B (Figure 3.1), it is not hard to verify that $k(1) = 2$ and $k(2) = 5$. From the condition on $k(i+1)$, $k(i+1) > p_i(k(i)) = k(i)^i + i > k(i)^2 > 2k(i)$ for $i > 1$ (note that $i > 1$ implies $k(i) \geq 5$). We obtain an estimate for the number of simulations, $l \leq \log n$. During the simulation of the machine M_i on input $0^{k(i)}$, we need to store all queries of length $k(i)$. This takes time $O(p_i(n))$. After the simulation, we need to find the word $y(i)$ (the first word of length $k(i)$, in lexicographical order, which is not queried during the simulation). This can be done in time $O(np_i(n)\log(np_i(n)))$, which is $O(np_i(n)\log n)$.

Therefore, time complexity of the algorithm for B can be bounded as

$$\sum_{i=1}^{\log n} np_i(n)\log n = n\log n \sum_{i=1}^{\log n}(n^i+i) \leq (n\log^2 n)(n^{\log n}+\log n) = O(n^{c+\log n}) = O(n^{O(\log n)})$$

for a constant c. Since $\mathsf{DTIME}[.]$ is closed under $O(.)$, $B \in \mathsf{DTIME}[t(n)]$ for every function $t(n)$ such that $t(n) = n^{O(logn)}$. Function 2^{cn} for a constant c fulfils this requirement.

□

Note that the proof yields the following corollary.

Corollary 3.8 (from proof of Theorem 3.7) *Let $t(n)$ be a function such that $t(n) = n^{O(\log n)}$. Then there exists an oracle $B \in \mathsf{DTIME}[t(n)]$ with $\mathsf{P}^B \neq \mathsf{NP}^B$.*

3.3 Random oracle hypothesis

In one line of research, the following question was investigated: How frequent are oracles relative to which $\mathsf{P} = \mathsf{NP}$ relativized?

Mehlhorn [Meh73] showed that the class of oracles which makes $\mathsf{P} = \mathsf{NP}$ relativized is *meager* (a topological concept which can be interpreted as "such sets are infrequent").

In the seminal paper on the Random oracle hypothesis, Bennett and Gill [BG81] presented that for almost all oracles $\mathsf{P} \neq \mathsf{NP}$ relativized. What does "almost all" mean ? It means that $\mathsf{P} \neq \mathsf{NP}$ relativized to the random oracle.

Definition 3.9 ([BG81]) *The oracle A is called the* random oracle *if and only if every string is in A with probability 1/2.*

Theorem 3.10 ([BG81]) *For the random oracle A holds $\mathsf{P}^A \neq \mathsf{NP}^A$ with probability* 1 .

The previous theorem can be formulated in terms of the measure theory. Let us denote by μ the Lebesque measure on the unit interval and Ω as the set of all languages. Since we can represent every element in Ω as an infinite sequence of 0's and 1's (a characteristic sequence), we can identify each language with a real number between 0 and 1. Then, the probability measure over Ω is equivalent to the Lebesgue measure μ on the unit interval. Hence, the statement $\mu(\{A \mid \mathsf{P}^A = \mathsf{NP}^A\}) = 0$ is equivalent to the statement that probability (over A) of $\mathsf{P}^A \neq \mathsf{NP}^A$ is equal to 1. And this is equivalent to the statement of Theorem 3.10.

Baker and Gill [BG81] also conjectured the *Random oracle hypothesis*. Roughly speaking, it says that "a relativized statement which is true with probability 1 with the random oracle is true for the empty oracle". Kurtz [Kur83] disproved a certain version of this conjecture by double relativization (two oracle sets are available for the machine).

Recently, another approach has been investigated. The Hausdorff dimension is used for classification of the sets of measure 0 (for sets of measure > 0, the Hausdorff dimension is always 1; for sets of measure 0, the Hausdorff dimension $\in [0, 1]$). Note that for every dimension $d \in [0, 1]$, there exists a set of measure 0 with dimension d. Hitchcock [Hit04] proved that the class $\{A \mid \mathsf{P}^A = \mathsf{NP}^A\}$ has Hausdorff dimension 1.

3.4 Positive relativization

In recursion theory the *relativization principle* holds: If an assertion holds in an unrelativized case, then it holds in the presence of an arbitrary oracle. Theorems 3.3 and 3.6 show that this principle does not hold in a polynomial time world.

The relativizing principle is a restriction on the machines or the oracles which allows one to show that the equality/inequality among unrelativized complexity classes is equivalent to the equality/inequality in any arbitrary relativization of a restricted model. For solving an unrelativized case, it is therefore sufficient to prove a relationship in arbitrary relativization of a restricted model. See [BDG90] for example of relativizing principle.

The relativizing principle is a special and stronger case of something which we meet in the structural complexity theory more often and what we call positive relativization. The *Positive relativization* is a restriction on the machines or the oracles which allows one to show that the equality among unrelativized complexity classes is equivalent to the equality in any arbitrary relativization of a restricted model. We do not require equivalence between the inequality in unrelativized case and the inequality in arbitrary relativization of restricted model anymore.

Negative relativization can be defined similarly. We still do not know much about relationships among complexity classes. However, in most cases we suppose a separation, i.e. an inequality between complexity classes. That is the reason why positive relativization has been studied so much. When we have positive relativization of the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem, it is sufficient to separate restricted classes with an arbitrary oracle in order to separate P from NP.

There are basically two types of positive relativization. The first of them puts restrictions on the machines. Namely, it restricts access to the oracle.

See [BDG90] for an example of this kind of positive relativization (nondeterministic machine can query only polynomial many words in the entire computation tree). The second one puts restrictions on the type of the oracle. The most studied property of the oracle is density, and special attention is focused on tally and sparse sets.

Long and Selman [LS86] showed that sparse sets yield positive relativization of particular levels of the polynomial time hierarchy from the second level up. We show the idea of their proof in case of the first level of the polynomial time hierarchy. But instead of sparse sets, we consider sets that are "easily enumerable".

But before that, one can ask how it is with the first level of the polynomial time hierarchy and sparse sets. Does the following proposition hold: $\mathsf{P} = \mathsf{NP}$ if and only if $\mathsf{P}^S = \mathsf{NP}^S$ for every sparse set S? This is probably hard to prove. Theorem 3.6 shows the existence of the sparse oracle B with $\mathsf{P}^B \neq \mathsf{NP}^B$. Proving the foregoing proposition would immediately separate P from NP. On the other hand, showing $\mathsf{P}^S = \mathsf{NP}^S$ for some sparse set S would collapse the polynomial time hierarchy $\mathsf{PH} = \Theta_2^p$.

Definition 3.11 ([LS86]) *For a set A, define function $enum_A(0^n)$ which returns the characteristic function of $A^{\leq n}$.*

Theorem 3.12 ([LS86]) $\mathsf{P} = \mathsf{NP}$ *if and only if* $\mathsf{P}^T = \mathsf{NP}^T$ *for every T such that $enum_T \in \mathsf{PF}^T$.*

Proof The if part is obvious since for an empty set $\mathrm{enum}_\emptyset$ is clearly in $\mathsf{PF}^\emptyset = \mathsf{PF}$. Only if part. Take an arbitrary $L \in \mathsf{NP}^T$, then there is a nondeterministic polynomial time Turing machine with an oracle such that $L = L(M, T)$. Let p be the polynomial which bounds the running time of M. We describe the machine in P^T which recognizes L. Denote by M' the Turing machine that works exactly the same as M, but instead of the oracle uses the characteristic function of the oracle set on one of its working tapes. The language accepted by M' is in NP, and therefore in P by the assumption $\mathsf{P} = \mathsf{NP}$. Consider the machine N on input of length n. First of all, N computes $\mathrm{enum}_T(0^{p(|n|)})$ with the help of the oracle T. Afterwards, N simulates the machine M' with initial segment of T computed in the first stage. N recognizes L and belongs to P^T. Since L is arbitrary, $\mathsf{P}^T = \mathsf{NP}^T$. □

Clearly, $\mathrm{enum}_T \in \mathsf{PF}^T$ holds for every tally set T. That gives us the next corollary.

Corollary 3.13 ([LS86]) $\mathsf{P} = \mathsf{NP}$ *if and only if* $\mathsf{P}^T = \mathsf{NP}^T$ *for every* $T \in \mathsf{TALLY}$.

Note a general approach of the so-called *oracle replacement*. The oracle is not too complicated and the class P is powerful enough to compute the initial segment of its oracle (i.e. $\text{enum}_T \in \mathsf{P}^T$). The class NP is robust enough for the oracle replacement: Instead of using the oracle T, the characteristic function of (the needed part of) the oracle can be used.

Hartmanis extended the result from Theorem 3.12 to the oracles that have the (desired) property that their initial segments can be computed deterministically in polynomial time relative to the set itself.

Theorem 3.14 ([Har83]) $\mathsf{P} = \mathsf{NP}$ *if and only if* $\mathsf{P}^A = \mathsf{NP}^A$ *for every* $A \subseteq \mathsf{K}[\log n, n^{O(c)}]$.

Finally, we prove that sets from the first level of the extended low level hierarchy provide positive relativization of the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem.

Proposition 3.15 ([BDG90]) $\mathsf{P} = \mathsf{NP}$ *if and only if* $\mathsf{P}^A = \mathsf{NP}^A$ *for every* $A \in \mathsf{EL}_1$.

Proof Since the empty set is in EL_1, the if implication is immediate. Only if implication. Suppose $\mathsf{P} = \mathsf{NP}$ and $A \in \mathsf{EL}_1$. By the definition of EL_1, $\mathsf{NP}^A \subseteq \mathsf{P}^{A \oplus SAT}$. But if $\mathsf{P} = \mathsf{NP}$, the answer for the oracle queries to SAT can be computed in polynomial time, and hence $\mathsf{P}^{A \oplus SAT} = \mathsf{P}^A$. Thus $\mathsf{NP}^A \subseteq \mathsf{P}^A$. □

3.5 Bibliographical remarks

We have surveyed some basic facts from the theory of relativization. Many results are presented in [BDG90] and [HO02], where the reader can find many references. All involved concepts, namely relativization, separation, the Random oracle hypothesis, the relativizing principle and positive relativization, do not concern only the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem. All of them have been used in many other problems concerning relationships among complexity classes.

Vereshchagin [Ver94] observed that all constructions of the separating oracles have two parts: Standard diagonalization and the combinatorial part which allows to perform the diagonalization. In his work, he introduced a general framework for the construction of the separating oracles.

Hartmanis [Har83] proved Theorem 3.6 (the existence of the separating oracle relative to the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem) in terms of Kolmogorov complexity.

Balcázar, Book, and Schöning [BBS86], using a different technique than [LS86], showed that sparse sets yield positive relativization of particular levels

of the polynomial time hierarchy. Moreover, they showed that sparse sets provide the relativizing principle for (non)collapse of the polynomial time hierarchy and for equality of the polynomial time hierarchy and PSPACE .

Both Balcázar, Book, and Schöning [BBS86] and Long and Selman [LS86] presented positive relativizations of other complexity classes with sparse sets.

Using the oracle replacement approach, Köbler et al. [KSTT89] showed that sparse sets provide positive relativization of relationships between complexity classes concerning the number of accepting computations paths.

Long [Lon85] compared restricting the density of the oracle (sparse sets) with restricting the access to the oracle (polynomial number of queries in the whole computation tree of nondeterministic computation).

Bovet, Crescenzi, and Silvestri [BCS95] presented a connection between oracles and *leaf languages*. Their concept of a leaf language is a machine independent characterization of complexity classes by a pair of languages. Their work refines oracle replacement and generalizes all previously known results and besides that gives us some new ones.

[All90], [For94], [HCCC$^+$] are devoted to discussions about relativization. [HZR95] is an open-questions paper in the theory of relativization.

Chapter 4

Collapsing Oracles

In this chapter, we investigate properties of the oracles relative to which $\mathsf{P} = \mathsf{NP}$.

Definition 4.1 (Class of collapsing oracles)

$$\mathcal{X} = \{A \mid \mathsf{P}^A = \mathsf{NP}^A\}.$$

First of all, we show a connection between the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem and the class $\mathcal{X}$.

Next we show basic structural properties of $\mathcal{X}$ concerning reductions, boolean operations and the question of density.

Then we show which sets belong to $\mathcal{X}$. We show that complete problems for the polynomial time hierarchy belong to $\mathcal{X}$ if and only the if the polynomial time hierarchy collapses. We present the known results about complete sets for deterministic/nondeterministic exponential time/space complexity classes which belong to $\mathcal{X}$. This allows us to show that $\mathcal{X}$ is not closed under $\leq_m^p$-reductions downward. We strengthen results concerning exponential classes to double exponential classes, triple exponential classes and so on. Then we consider subexponential classes. We show that one of those classes does not have complete problems under polynomial time reductions. We also show that if this class has a complete problem under any reductions, then it has a complete problem which is not in $\mathcal{X}$.

Further, we show the known result that hard problems of complexity classes whose complete problems belong to $\mathcal{X}$ do not generally belong to $\mathcal{X}$. We conclude that $\mathcal{X}$ is not closed under $\leq_m^p$-reductions upward.

We present two characterizations of the class $\mathcal{X}$. The first one claims that sets from $\mathcal{X}$ are those which are hard for NP relativized to the set itself. The second one shows that sets from $\mathcal{X}$ are exactly those which lie in the

intersection of the first level of the extended low and the zeroth level of the extended high hierarchy

Next we show that $\mathcal{X}$ is not closed under any of these boolean operations: intersection, union and symmetric difference.

Then we investigate the relationship between $\mathcal{X}$ and the extended hierarchies.

Finally, we consider sets reducible to $\mathcal{X}$ and mention a few open problems and related results.

4.1 $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem in terms of $\mathcal{X}$

First we show that the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem is equivalent to the membership of certain sets to $\mathcal{X}$. Sets that are concerned are those that give a positive relativization of the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem.

Theorem 4.2 *The following statements are equivalent:*

1. $\mathsf{P} = \mathsf{NP}$.
2. $\emptyset \in \mathcal{X}$.
3. $(\exists A \in \mathsf{P})[A \in \mathcal{X}]$.
4. $(\forall A \in \mathsf{P})[A \in \mathcal{X}]$.
5. $(\forall T \in \mathsf{TALLY})[T \in \mathcal{X}]$.
6. $(\forall A \; s.t. \; enum_A \in \mathsf{PF}^A)[A \in \mathcal{X}]$.
7. $(\forall A \subseteq \mathsf{K}[\log n, n^{O(c)}])[A \in \mathcal{X}]$.
8. $(\forall A \in \mathsf{EL}_1)[A \in \mathcal{X}]$ *(i.e.* $\mathsf{NP}^A \subseteq \mathsf{P}^{A \oplus SAT}$).

Proof Statement (1) is equivalent with the statements (2) to (8), because of the definition of $\mathcal{X}$, the fact that $\mathsf{P}^{\mathsf{P}} = \mathsf{P}$ and $\mathsf{NP}^{\mathsf{P}} = \mathsf{NP}$, Corollary 3.13, Theorem 3.12 and 3.14 and Proposition 3.15 respectively.

Note that example of set A in (6) is an exponential padding of an arbitrary set. □

4.2 Basic properties of $\mathcal{X}$

Here we present some basic structural properties of $\mathcal{X}$ which mostly follow from known results from the relativization theory.

First we show that $\mathcal{X}$ is neither empty nor contains all languages.

Proposition 4.3 *$\mathcal{X}$ is neither the empty class ($\mathcal{X} \neq \emptyset$) nor the class of all sets.*

Proof Nonemptiness follows from Theorem 3.3. The second statement follows from Theorem 3.6. □

Further we present the result that $\mathcal{X}$ is small by means of the measure theory (see Theorem 3.10).

Proposition 4.4 ([BG81]) $\mu(\mathcal{X}) = 0$.

Note that this is a non-trivial result! But $\mu(\{\mathcal{X} \cap \text{computable}\}) = 0$ is trivial because even $\mu(\{A \mid A \text{ computable}\}) = 0$. Mehlhorn [Meh73] showed that $\{\mathcal{X} \cap \text{computable}\}$ is effectively meager.

Simple switching the answers from the oracle yields next result.

Proposition 4.5 *$\mathcal{X}$ is closed under complements.*

Next we show that $\mathcal{X}$ is closed under Turing equivalences and consequently under many other computational weaker reductions and under polynomial time isomorphisms.

Proposition 4.6 *$\mathcal{X}$ is closed under $\equiv_T^p$.*

Proof Let $A \in \mathcal{X}$, $A \equiv_T^p B$ and $L \in \mathsf{NP}^B$ be arbitrary. Since $B \leq_T^p A$, $L \in \mathsf{NP}^A$ and by assumption, $L \in \mathsf{P}^A$. From $A \leq_T^P B$, $L \in \mathsf{P}^B$. Since L is arbitrary, $\mathsf{NP}^B \subseteq \mathsf{P}^B$, and $B \in \mathcal{X}$. □

Later on we show that $\mathcal{X}$ is closed under many-one reductions neither downward nor upward.

Afterwards we also show that $\mathcal{X}$ is not closed under unions, intersections and symmetric differences. Now we prove that if $\mathcal{X}$ was closed under unions (or intersections or symmetrical differences), it would mean that $\mathsf{P} = \mathsf{NP}$.

Proposition 4.7 *If $\mathcal{X}$ is closed under unions, or intersections or symmetric differences then $\mathsf{P} = \mathsf{NP}$.*

Proof

From Proposition 4.3 consider an arbitrary $L \in \mathcal{X}$. From Proposition 4.5 coL is also in $\mathcal{X}$ and clearly L and coL are disjoint. $L \cup coL = \Sigma^*$, $L \cap coL = \emptyset$ and $L \triangle coL = \Sigma^*$. By assumption, $\emptyset$ or the set of all words Σ^* belongs to $\mathcal{X}$. Both of these sets belong to P and Theorem 4.2 (3) finishes the proof. □

We do not know whether $\mathcal{X}$ is closed under disjoint unions or not.

An interesting property concerning many problems is the question of density. Sparse sets, i.e. sets that have only polynomial many words of given

length, play an important role in many areas of structural complexity. As mentioned in Section 3.4, sparse set in $\mathcal{X}$ would collapse the polynomial time hierarchy $\mathsf{PH} = \Theta_2^p$. Because we do not suppose a collapse of the polynomial time hierarchy, we do not expect sparse sets in $\mathcal{X}$.

4.3 Inside of $\mathcal{X}$

What sets are in $\mathcal{X}$? We saw in Theorem 4.2 that in some sense "computable simple" sets (like sets from P, tally sets, sets from the first level of the extended low hierarchy and others) are in $\mathcal{X}$ if and only if $\mathsf{P} = \mathsf{NP}$. It is widely believed that $\mathsf{P} \neq \mathsf{NP}$ and therefore we do not suppose these sets to be in $\mathcal{X}$.

In next sections, we present results about other sets and their relationships with $\mathcal{X}$. First we show that the sets from the polynomial time hierarchy are in $\mathcal{X}$ if and only if the polynomial time hierarchy collapses. Then we show that complete sets for all standard, i.e. deterministic/nondeterministic, time/space, exponential complexity classes belong to $\mathcal{X}$. After that we strengthen results about exponential complexity classes to double, triple and so on exponential classes. At the end, we investigate some classes which lie between polynomial and exponential complexity classes.

4.3.1 Polynomial time

First of all, we show that sets from the polynomial time hierarchy are in $\mathcal{X}$ if and only if the polynomial time hierarchy collapses. It is the same case as the one mentioned in the previous paragraph. The polynomial hierarchy is believed to be infinite, so we do not suppose sets from the polynomial time hierarchy to be in $\mathcal{X}$.

This proposition is given in [BDG88] as an exercise.

Proposition 4.8 ([BDG88]) *The following statements are equivalent:*

1. $(\exists A \in \mathsf{PH})[A \in \mathcal{X}]$.
2. $(\exists A \in \mathsf{PH})[\mathsf{PH}^A$ *collapses*$]$.
3. $(\forall A \in \mathsf{PH})[\mathsf{PH}^A$ *collapses*$]$.
4. PH *collapses.*

Proof

"$1 \Rightarrow 2$":

If there exists $A \in \mathsf{PH}$ with $\mathsf{P}^A = \mathsf{NP}^A$, then PH^A collapses to the first level $\mathsf{PH}^A = \Sigma_1^{p,A} = \mathsf{NP}^A$.

"$2 \Leftrightarrow 3 \Leftrightarrow 4$":

From $\emptyset \leq_T^p A$, $\Sigma_1^p = \Sigma_1^{p,\emptyset} \subseteq \Sigma_1^{p,A} \subseteq \Sigma_{k+1}^p$ for $A \in \Sigma_k^p$. Further, $\Sigma_1^{p,A} = \Sigma_{k+1}^p$ for $A \in C_T^p(\Sigma_k^p)$.

By mathematical induction over i, we obtain for all i that $\Sigma_i^{p,A} \subseteq \Sigma_{k+i}^p \subseteq \Sigma_{k+i}^{p,A}$ for all $A \in \Sigma_k^p$ and $\Sigma_i^{p,A} = \Sigma_{k+i}^p$ for all $A \in C_T^p(\Sigma_k^p)$. Hence, if $\Sigma_i^{p,A} = \Sigma_{i+1}^{p,A} = \mathsf{PH}^A$ for some $A \in \Sigma_k^p$, then $\Sigma_{k+i}^p = \mathsf{PH}$, and if $\Sigma_k^p = \mathsf{PH}$, then $\Sigma_{k+i}^{p,A} = \mathsf{PH}^A$ for all $A \in \Sigma_i^p$ and all i.

"$4 \Rightarrow 1$":

Let $\mathsf{PH} = \Sigma_k^p$. Take an arbitrary $A \in C_T^p(\Sigma_k^p)$. Then $\mathsf{NP}^A = \Sigma_{k+1}^p = \Delta_{k+1} = \mathsf{P}^A$. □

4.3.2 Exponential deterministic time

Next lemma states a well known simulation of nondeterministic computation. This allows us to show that complete problems for deterministic exponential time complexity classes belong to $\mathcal{X}$.

Lemma 4.9 (folklore) *Let* $A \in \mathsf{DTIME}[t(n)]$ *and* $\mathsf{DTIME}[2^{n^{O(c)}} t(n^{O(c)})] \subseteq \mathsf{P}^A$. *Then* $A \in \mathcal{X}$.

Proof Take an arbitrary $L \in \mathsf{NP}^A$. We show that $L \in \mathsf{P}^A$. Since $L \in \mathsf{NP}^A$, there exists a nondeterministic polynomial time Turing machine M with an oracle such that $L = L(M, A)$. Let $p(n)$ be the polynomial which bounds the running time of M on input of length n. We can simulate computation of M by depth-first search in the computation tree of M. For input of length n, this tree has depth $p(n)$ and at most $c^{p(n)}$ branches for a constant c. Therefore, the simulation can be performed deterministically in time $O(c^{p(n)}p(n))$. Since $p(n) \geq n$, we can rewrite this as $O(2^{q(n)})$ for some polynomial $q(n)$. We also have to take into account the oracle queries whose length is bounded by $p(n)$. Together, we get a deterministic simulation which works in time $O(2^{n^{O(c)}} t(n^{O(c)}))$. From $\mathsf{DTIME}[2^{n^{O(c)}} t(n^{O(c)}] \subseteq \mathsf{P}^A$, it follows that $L \in \mathsf{P}^A$, and thus $\mathsf{P}^A = \mathsf{NP}^A$. □

Corollary 4.10

1. $C_T^p(\mathsf{DEXT}) \subseteq \mathcal{X}$.
2. $C_T^p(\mathsf{EXP}) \subseteq \mathcal{X}$.

Proof

1. Take an arbitrary $A \in C_T^p(\mathsf{DEXT})$, i.e. $A \in \mathsf{DTIME}[2^{cn}]$ for a constant c. Clearly, $O(2^{n^{c_1}} 2^{cn^{c_2}}) = O(2^{n^{c_3}})$ for some constants c_1, c_2 and c_3. Clearly, $\mathsf{DTIME}[2^{n^{c_3}}] \subseteq \mathsf{EXP}$. Applying the previous Lemma 4.9 with Remark 2.23 (2) prove the first statement.

2. Take an arbitrary $A \in C_T^p(\mathsf{EXP})$, i.e. $A \in \mathsf{DTIME}[2^{n^c}]$ for a constant c. Clearly, $O(2^{n^{c_1}} 2^{n^{c_2}}) = O(2^{n^{c_3}})$ for some constants c_1, c_2 and c_3. Clearly, $\mathsf{DTIME}[2^{n^{c_3}})] \subseteq \mathsf{EXP}$. Applying Lemma 4.9 finishes the proof.

□

Using the just presented result that complete problems for DEXT belong to $\mathcal{X}$, we can prove that $\mathcal{X}$ is not closed under many-one reductions downward.

Theorem 4.11 *$\mathcal{X}$ is not closed under $\leq_m^p$ downward ($A \leq_m^p B$ and $B \in \mathcal{X} \not\Rightarrow A \in \mathcal{X}$).*

Proof By Theorem 3.6, there exists $B \notin \mathcal{X}$ and by Theorem 3.7, $B \in \mathsf{DEXT}$. For an arbitrary $A \in C_m^p(\mathsf{DEXT})$, $B \leq_T^p A$ and by Corollary 4.10 (1), $A \in \mathcal{X}$. □

A simple corollary is that $\mathcal{X}$ can not be closed downward under stronger reductions than many-one, for example bounded truth-table, truth-table and Turing reductions.

4.3.3 Exponential nondeterministic time

The case of the nondeterministic exponential time complexity class is not so obvious as its deterministic counterpart.

Consider an arbitrary $L \in \mathsf{NP}^{\mathsf{NEXT}}$ via the machine M. The deterministic simulation of computation of the machine M which is used in Lemma 4.9 has complexity $O(2^{n^{O(c)}} 2^{2^{n^{O(c)}}})$ which is not in NEXP much less in NEXT.

Hemachandra [Hem89] presented a much smarter strategy how to simulate computation of M. He showed that $\mathsf{P}^{\mathsf{NEXT}}$ can construct increasingly accurate partial census information about the number of yes responses that NEXT makes to queries from NP in $\mathsf{NP}^{\mathsf{NEXT}}$ computation.

By showing that $\mathsf{P}^{\mathsf{NEXT}} = \mathsf{NP}^{\mathsf{NEXT}}$, Hemachandra proved that the strong exponential hierarchy collapses to its second level, $\mathsf{SEH} = \Delta_2^{\mathsf{SEH}} = \mathsf{P}^{\mathsf{NEXT}}$.

This result gives us an analogy of Corollary 4.10 for nondeterministic case. NEXP case is obtained by a padding argument. We do not proof the next theorem now, but we present the proof technique later for obtaining a more general result.

Theorem 4.12 ([Hem89])

1. $C_T^p(\mathsf{NEXT}) \subseteq \mathcal{X}$.
2. $C_T^p(\mathsf{NEXP}) \subseteq \mathcal{X}$.

4.3.4 Polynomial and exponential space

The first sets known to belong to $\mathcal{X}$ are complete problems for PSPACE (see Theorem 3.4).

By the same technique (the simple simulation of nondeterministic computation), which is used in the previous sections, we can prove that complete problems for exponential space complexity classes belong to $\mathcal{X}$ also.

Lemma 4.13 (folklore) *Let* $A \in \mathsf{DSPACE}[s(n)]$ *and* $\mathsf{DSPACE}[n^{O(c)} + s(n^{O(c)})] \subseteq \mathsf{P}^A$. *Then* $A \in \mathcal{X}$.

Proof Take an arbitrary $L \in \mathsf{NP}^A$. We show that $L \in \mathsf{P}^A$. Since $L \in \mathsf{NP}^A$ there exists a nondeterministic polynomial time Turing machine M with an oracle A such that $L = L(M, A)$. Let $p(n)$ be the polynomial which bounds the running time of M on input of length n. Note that $p(n)$ also bounds space used by M on every input of length n because any branch of nondeterministic computation does not have time to use more space than its working time. We can simulate computation of M by depth-first search in the computation tree of M. For input of length n, every branch of the computation tree has space complexity $O(p(n))$ and there are at most $c^{p(n)}$ branches for a constant c. Therefore, the simulation of all branches can be performed in deterministic space $O(p^2(n))$ (it can be performed in space $O(p(n))$, but we do not need it since the estimate is polynomial). We also have to take into account oracle queries, which can be at most $p(n)$ long. Together, we get a deterministic simulation working in space $O(n^{O(c)} + s(n^{O(c)}))$. From $\mathsf{DSPACE}[n^{O(c)} + s(n^{O(c)})] \subseteq \mathsf{P}^A$, it follows that $L \in \mathsf{P}^A$, and thus $\mathsf{P}^A = \mathsf{NP}^A$. □

Corollary 4.14

1. $C_T^p(\mathsf{PSPACE}) \subseteq \mathcal{X}$.
2. $C_T^p(\mathsf{DEXTSPACE}) \subseteq \mathcal{X}$.
3. $C_T^p(\mathsf{EXPSPACE}) \subseteq \mathcal{X}$.

Proof

1. Take an arbitrary $A \in C_T^p(\mathsf{PSPACE})$, i.e. $A \in \mathsf{DSPACE}[n^c]$ for a constant c. Clearly, $O(n^{c_1} + (n^{c_2})^c) = O(n^{c_3})$ for some constants c_1, c_2 and c_3. Clearly, $\mathsf{DSPACE}[n^{c_3}] \subseteq \mathsf{PSPACE}$. Lemma 4.13 proves the first statement.
2. Take an arbitrary $A \in C_T^p(\mathsf{DEXTSPACE})$, i.e. $A \in \mathsf{DSPACE}[2^{cn}]$ for some a c. Clearly, $O(n^{c_1} + 2^{cn^{c_2}}) = O(2^{n^{c_3}})$ for some constants c_1, c_2 and c_3. Clearly, $\mathsf{DSPACE}[2^{n^{c_3}}] \subseteq \mathsf{EXPSPACE}$. Space analogy of Remark 2.23 (2) (padding of EXPSPACE problems to DEXTSPACE) and Lemma 4.13 prove the second statement.

3. Take an arbitrary $A \in C_T^p(\mathsf{EXPSPACE})$, i.e. $A \in \mathsf{DSPACE}[2^{n^c}]$ for a constant c. Clearly, $O(n^{c_1} + 2^{(n^{c_2})^c}) = O(2^{n^{c_3}})$ for some constants c_1, c_2 and c_3. Clearly, $\mathsf{DSPACE}[2^{n^{c_3}}] \subseteq \mathsf{EXPSPACE}$. Lemma 4.13 finishes the proof.

□

In space complexity classes, the nondeterministic case is not interesting because before mentioned classes PSPACE, DEXTSPACE and EXPSPACE are closed under nondeterminism because of Savitch Theorem 2.11.

4.3.5 Beyond exponential classes

We can strengthen the previous results, by the same technique, to the complete problems for double, triple and so on exponential complexity classes.

Definition 4.15 *For natural number $k \geq 1$, define:*

1. $\exp_1 = \cup_c \{f \mid f = O(2^{n^c})\}$.
2. $\exp_{k+1} = \cup_g \{f \mid f = O(2^g) \wedge g \in \exp_k\}$.

Definition 4.16 *For natural number $k \geq 1$, define:*

1. $\mathsf{EXP}_k = \cup_f \{\mathsf{DTIME}[f(n)] \mid f \in \exp_k\}$.
2. $\mathsf{NEXP}_k = \cup_f \{\mathsf{NTIME}[f(n)] \mid f \in \exp_k\}$.
3. $\mathsf{EXPSPACE}_k = \cup_f \{\mathsf{DSPACE}[f(n)] \mid f \in \exp_k\}$.

Clearly, $\mathsf{EXP}_1 = \mathsf{EXP}$, $\mathsf{NEXP}_1 = \mathsf{NEXP}$ and $\mathsf{EXPSPACE}_1 = \mathsf{EXPSPACE}$.

In case of deterministic time complexity classes, the strengthening is easy. The same simulation of nondeterministic computation that is used in proof of Corollary 4.10 gives us next result.

Theorem 4.17 *For natural number $k \geq 1$, $C_T^p(\mathsf{EXP}_k) \subseteq \mathcal{X}$.*

Proof For fixed $k \geq 1$, take an arbitrary $A \in C_T^p(\mathsf{EXP}_k)$, i.e. $A \in \mathsf{DTIME}[f(n)]$ for $f \in \exp_k$, and $\mathsf{DTIME}[f'(n)] \subseteq \mathsf{P}(A)$ for every $f'(n) \in \exp_k$. Clearly, $2^{n^{c_1}} f(n^{c_2}) = O(f'(n))$ for some constants c_1 and c_2 and some $f'(n) \in \exp_k$. Lemma 4.9 finishes the proof. □

In case of space complexity classes, it is very similar.

Theorem 4.18 *For natural number $k \geq 1$, $C_T^p(\mathsf{EXPSPACE}_k) \subseteq \mathcal{X}$.*

Proof For fixed $k \geq 1$, take an arbitrary $A \in C_T^p(\mathsf{EXPSPACE}_k)$, i.e. $A \in \mathsf{DSPACE}[f(n)]$ for $f \in \exp_k$, and $\mathsf{DSPACE}[f'(n)] \subseteq \mathsf{P}(A)$ for every $f'(n) \in \exp_k$. Clearly $n^{c_1} + f(n^{c_2}) = O(f'(n))$ for some constants c_1 and c_2 and some $f'(n) \in \exp_k$. Lemma 4.13 finishes the proof. □

The situation is more difficult in case of nondeterministic time complexity classes. As mentioned in Section 4.3.3, presenting Theorem 4.12, a simple simulation of the nondeterministic Turing machine is not sufficient. We describe the technique of Hemachandra ([Hem89]), which is used in the proof of Theorem 4.19.

Consider the computation tree of a machine from $\mathsf{NP}^{\mathsf{NEXT}}$. The idea is that the machine from $\mathsf{P}^{\mathsf{NEXT}}$ can compute the number of query strings receiving yes answers from the oracle at each level of the computation tree.

We can not simply compute the number of yes answers at some level in the tree. To even know which string are queried at a particular level, we must primarily know the answers to queries at the previous levels of the computation tree. So, we have to proceed gradually, first of all compute the number of yes responses at the first level of the tree. Using this information, we can compute the number of yes responses at the second level of the tree and so on. At each level, we use knowledge of the previous levels to help us binary search for the number of yes responses at the current level.

A typical example of foregoing computation is this one: We already know that the computation tree has exactly 1, 0, 2, 3 yes responses at levels 1, 2, 3, 4 and between 4 and 8 at level 5. Our machine (from P) asks the oracle (from NEXP) this question: Given input x and assuming 1, 0, 2, and 3 are the correct numbers of yes answers at levels 1, 2, 3, and 4, are there at least 6 yes strings at level 5?

This question can be answered by NEXP machine: It guesses the first five levels of computation tree, checks that the tree corresponds to the considered NP machine, checks that guessed yes queries are really in NEXP (by guessing the proofs), checks that there are 1, 0, 2, 3 yes responses at levels 1, 2, 3, 4 and at least 6 yes responses queried at level 5.

Finally, we know the number of yes responses at each level. The final query of the machine from P to its NEXT oracle is about acceptance of the input word.

Note that the machine from P can remember just the number of yes responses at each level of computation tree. There is no way of how the polynomial time machine could remember all the queries, whose number may be exponential at a single level.

Theorem 4.19 *For natural number $k \geq 1$, $C_T^p(\mathsf{NEXP}_k) \subseteq \mathcal{X}$.*

Proof For fixed $k \geq 1$, we show that $\mathsf{P}^{\mathsf{NEXP}_k} = \mathsf{NP}^{\mathsf{NEXP}_k}$. Consequently, $\mathsf{P}^U = \mathsf{NP}^U$ for $U \in \mathcal{C}_T^p(\mathsf{NEXP}_k)$.

Take an arbitrary $L \in \mathsf{NP}^{\mathsf{NEXP}_k}$. Then there exists a nondeterministic Turing machine M_1 with an oracle such that its running time is bounded by $f(n) \in \exp_k$ and there exists a nondeterministic polynomial time Turing machine M_2 with an oracle such that $L = L(M_2, L(M_1))$. Let denote by L' the language accepted by the machine M_1, i.e. $L' = L(M_1)$. Then $L = L(M_2, L')$. Let $p(n)$ be the polynomial which bounds the running time of the machine M_2. Consider the computation tree of the machine M_2 with the oracle L'. We refer to it later as to the computation tree. Since the running time of M_2 is bounded by $p(n)$, its computation tree on input of length n has at most $p(n)$ levels and at each level M_2 asks at most $2^{p(n)}$ queries which are at most $p(n)$ long.

We describe a nondeterministic Turing machine M_3 such that $L(M_3) \in \mathsf{NEXP}_k$ and a deterministic polynomial time Turing machine M_4 such that $L = L(M_4, L(M_3))$.

We define M_3 so that $L(M_3)$ fulfils following:

$L(M_3) = \{\langle \text{final}, x, c_1, c_2, \ldots, c_l\rangle \mid$ There exist sets $C_1, C_2, \ldots, C_l$ of strings such that:

1. $|C_i| = c_i$ and $\bigcup_i C_i \subseteq L'$, $1 \leq i \leq l$.
2. If we simulate M_2 on input x, answering each oracle query q at level i of computation tree with a yes response $\Leftrightarrow q \in C_i$, then each y in C_i is actually queried at level i in this simulation for every $1 \leq i \leq l$.
3. If final is 1, there is an accepting path in the simulation mentioned in 2 above$\}$.

Define (nondeterministic) Turing machine M_3 such that on input x':

1. If the input word x' is not a form of $\langle \text{final}, x, c_1, c_2, \ldots, c_l\rangle$, then reject. Otherwise, denote by $n = |x|$.
2. If $l > p(n)$ or $c_i > 2^{p(n)}$ for some i, $1 \leq i \leq l$, then reject.
3. For every i, $1 \leq i \leq p(n)$, nondeterministically guess the set C_i of at most $2^{p(n)}$ words of length up to $p(n)$.
4. If there is some i, $1 \leq i \leq l$, such that $|C_i| \neq c_i$, then reject.
5. For every i, $1 \leq i \leq p(n)$, for every word $w \in C_i$, $m = |w|$, do the following:
   (a) Nondeterministically guess p, the proof string of length up to $f(m)$.

(b) Simulate M_1 on input w in such way that in the i-th step of the simulation, use the i-th bit of the string p to choose the next state. Continue with the simulation until M_1 accepts or rejects.

(c) If M_1 accepts, then stop the simulation of M_1 and we continue in Step 5 with another word w.

(d) If M_1 rejects, then reject.

6. Simulate M_2 on input x by depth-first search of its computation tree until M_2 asks the oracle a word w or the simulation enters a leaf of the computation tree.
7. If the simulation enters a leaf, then check whether it is an accepting or rejecting leaf. If it is an accepting leaf, then remember the visiting accepting leaf.
8. If M_2 asks the oracle a word w in the simulation at the i-th level of the computation tree, then the answer is yes if and only if $w \in C_i$. If $w \in C_i$, then remember that the word $w \in C_i$ was queried. We continue on the simulation in Step 6.
9. If there is a i, $1 \leq i \leq l$, and $w \in C_i$ such that w was not queried in the simulation of M_2, then reject.
10. If final is 0, then accept.
11. If final is 1, then accept if and only if an accepting leaf was reached during the simulation of M_2 in Step 7.

Note that the language $L(M_3)$ accepted by the machine M_3 fulfils the conditions imposed on $L(M_3)$ above. Now we consider the complexity of $L(M_3)$.

Claim 1 $L(M3) \in \mathsf{NEXP}_k$.

Running time of M_1 is bounded by $f \in \exp_k$, i.e. f is a form of $2^{2^{\cdot^{\cdot^{2^{q(n)}}}}}$ for some polynomial $q(n)$ and there are at most k powers of 2. Let us define polynomial $r(n) = q(p(n))^2$, and let $f'(n) \in \exp_k$ be the same form as $f(n)$ only $q(n)$ is replaced by $r(n)$ at the top of k powers of 2. We show that $L(M_3) \in \mathsf{NTIME}[f'(n)]$.

Suppose input x' of length n.

Step 1 can be performed in time $O(n) = O(f'(n))$.

Step 2 can be performed in time $O(p(n)) = O(f'(n))$.

Step 3 can be performed in time $O(p^2(n)2^{p(n)}) = O(f'(n))$.

Step 4 can be performed in time $O(p(n)2^{p(n)}) = O(f'(n))$.

Steps 5a - 5d are performed at most $p(n)2^{p(n)}$ times.

Step 5a can be performed in time $O(f(m))$, $m = O(p(n))$, which is $O(f(p(n)))$.

Steps 5b - 5d together consume the time corresponding to the computation of M_1 on input of length $m = O(p(n))$. Since the running time of M_1 is bounded by $f(n)$, Steps 5b - 5d consume time $O(f(p(n))$.

Together, Steps 5a - 5d can be performed in time $O(p(n)2^{p(n)}f(p(n))f(p(n))) = O(f'(n))$.

Since the running time of M_2 is bounded by $p(n)$, Steps 6 - 9 can be performed in time $O(2^{p(n)}) = O(f'(n))$.

Steps 10 - 11 can be performed in time $O(1) = O(f'(n))$.

Therefore, the running time of M_3 is bounded by $f'(n)$. Since $f'(n) \in \exp_k$ and $\mathsf{NTIME}[f'(n)] \subseteq \mathsf{NEXP}_k$, Claim 1 is proved. □

Define the (deterministic) Turing machine M_4 such that on input x of length n:

1. Let $i = 1$ and $s = \lambda$ (empty string).
2. Let $k = 0$ and $l = 2^{p(n)}$.
3. If $k + 1 = l$, then go to Step 5.
4. Let $m = \lfloor (k + l)/2 \rfloor$ and $q = \langle 0, x, s, m \rangle$.
   Ask the oracle for a word q.
   If the answer is yes, then let $k = m$. Otherwise, let $l = m$.
   Go to Step 3.
5. Let $q = \langle 0, x, s, l \rangle$.
   Ask the the oracle for a word q.
   If the answer is yes, then let $c_i = l$. Otherwise, let $c_i = k$.
6. Increase i and let $s = c_1, c_2, \cdots, c_{i-1}$.
   If $i \leq p(n)$, then go to Step 2.
7. Let $q = \langle 1, x, c_1, c_2, \ldots, c_{p(n)} \rangle$.
   Ask the oracle for word a q.
   Accept if and only if the answer is yes.

Claim 2 $L = L(M_4, L(M_3))$.

Proof First note that M_4 is a deterministic polynomial time Turing machine with an oracle. Consider an input word of length n. For every i, $1 \leq i \leq p(n)$, M_4 computes by standard binary search method the number of yes responses at the i-th level of the computation tree of the machine M_2. In the i-th stage, binary search takes time $O(\log 2^{i-1}) = O(i)$. Since there is $p(n)$ stages, M_4 works in time $O(p(n)^2)$, i.e. in polynomial time.

It remains to show that with the oracle $L(M_3)$, M_4 accepts L.

An important observation is that if $c_0, c_1, \ldots, c_{i-1}$ are correct, then we find correct c_i. This is because some branch of nondeterminism computation in $L(M_3)$ will guess the true yes strings $C_0, C_1, \ldots, C_{i-1}$.

On the other hand, if some branch of computation in $L(M_3)$ does not guess sets $C_0, C_1, \ldots, C_{i-1}$ correctly, then it will not accept. Let us say the branch guess sets $C'_0, C'_1, \ldots, C'_{i-1}$ such that $|C_j| = |C'_j| = c_j$, $1 \leq j \leq i-1$, and let m be the minimal $i \leq j-1$ such that $C_m \neq C'_m$. Then there is a string $w \in C'_m$ which is not a yes string in the computation tree of M_2. Since C'_i are correct at levels $1, 2, \ldots, m-1$, the strings queried at level m in the simulation are exactly those queried at level m in the computation tree. Thus, the condition 1 imposed on $L(M_3)$ is violated and this branch will not accept. If w is not queried at level m, then the condition 2 imposed on $L(M_3)$ is violated and this branch will not accept.

After the last $p(n)$-th stage, we know the correct values $c_1, c_2, \ldots, c_{p(n)}$ and M_4 accepts if and only if the oracle answers yes for query $\langle 1, x, c_1, c_2, \ldots, c_{p(n)} \rangle$, which means M_2 accepts with the oracle L'. □

(Theorem 4.19) □

4.3.6 Between polynomial and exponential

In the previous sections, we see that complete problems for strong (i.e. exponential and above) complexity classes belong to $\mathcal{X}$. On the other hand, we do not know about complete problems for the polynomial time hierarchy. This section investigates how it is with complete problems for complexity classes which lie between polynomial and exponential complexity classes.

First of all, we strengthen the result of Theorem 3.6 and Theorem 3.7. Recall Theorems 3.6 and 3.7 which state existence of an oracle $B \in \mathsf{DEXT}$ such that $B \notin \mathcal{X}$.

Consider the enumeration $\{M_i\}_{i \geq 1}$ from Fact 2.27.

Recall that $\leq^b_a$, for appropriate strings a and b, denotes reduction. In case b is equal to p, $\leq^p_a$ denotes polynomial time reduction whose type is determined by string a (f.e. $\leq^p_m$ stands for polynomial time many-one reduction and $\leq^p_T$ stands for polynomial time Turing reduction).

Theorem 4.20 *Let $\mathcal{F}$ be a class of functions. Let $\mathcal{C} = \cup_{f \in \mathcal{F}} \mathsf{DTIME}[f]$ be a complexity class. If $\mathcal{C}$ has a complete problem $U \in \mathsf{DTIME}[f]$ under $\leq^p_a$, $f \in \mathcal{F}$ non-decreasing, and $\overline{f}(n) = \log n \sum_{i=1}^{\log n} [p_i(n) f(p_i(n))] \in \mathcal{F}$, then $\mathcal{C}$ has a complete problem V under $\leq^p_a$ with $V \notin \mathcal{X}$.*

Proof We show the construction of the oracle V that is complete for $\mathcal{C}$ and $V \notin \mathcal{X}$.

On words of even length, we encode U to ensure that V is hard for $\mathcal{C}$. On words of odd length, we diagonalize against P^V in the same way we do in the proof of Theorem 3.6.

We construct V in stages. Denote by $k(n)$ an increasing sequence of odd natural numbers: $k(n)$ is the length of word which is used in the n-th stage to ensure that M_n does not accept $L(V)$. After n stages, we denote the so far constructed oracle set by V_{n-1}. In the n-th stage, we add at most one word of length $k(n)$ to ensure that $L(V) \neq L(M_n, V)$. So, all words of length smaller than or equal to $k(n)$ are decided after the n-th stage. We show that either $0^{k(n)} \in L(M_n, V)$ and $V \cap \Sigma^{=k(n)} = \emptyset$, or $0^{k(n)} \notin L(M_n, V)$ and $V \cap \Sigma^{=k(n)} \neq \emptyset$. Figure 4.1 presents the n-the stage in the construction of V. The final oracle $V = \cup_n V_n$.

---

stage 0:
  $k(0) = 0$
  $V_0 = \{uu \mid u \in U\}$
stage n:
  Let $k(n)$ be the smallest odd natural number such that
    $k(n) > p_{n-1}(k(n-1))$ and $p_n(k(n)) < 2^{k(n)}$.
  If $0^{k(n)} \in L(M_n, V_{n-1})$ then $V_n = V_{n-1}$.
  If $0^{k(n)} \notin L(M_n, V_{n-1})$ then $V_n = V_{n-1} \cup \{y(n)\}$,
  where $y(n)$ is the first word, in lexicographic order, of length $k(n)$
  such that $y(n)$ is not queried in the computation of M_n on input $0^{k(n)}$
  with the oracle V_{n-1}.

Figure 4.1 Construction of $V \in \mathcal{X}$ with $U \leq_m^p V$.

From the condition on $k(n)$, an appropriate word $y(n)$ of length $k(n)$ always exists if needed. There are $2^{k(n)}$ words of length $k(n)$ and $p_n(k(n)) < 2^{k(n)}$.

The condition on $k(n)$ also ensures that $k(n)$ is long enough not to disturb, by possible adding of $y(n)$ to V, any computation from the previous stages. It means that $0^{k(n)} \in L(M_n, V_{n-1}) \Leftrightarrow 0^{k(n)} \in L(M_n, V)$. This holds for every n. Therefore, $0^{k(n)} \in L(V) \Leftrightarrow 0^{k(n)} \notin L(M_n, V)$ for every n. That means $L(V) \notin \mathsf{P}^V$. Since $L(V) \in \mathsf{NP}^V$, we obtain $V \notin \mathcal{X}$.

input $x, |x| = n$
if n is even **and** $x \neq ww$ **then**
  reject x
endif
if n is even **and** $x = ww$ **then**
  simulate M_U on w and accept $x \Leftrightarrow M_U$ accepts w
endif
$V' = \emptyset$
for every $m = 1, \ldots, n$ **do**
  i=Find(m)
  if $(i > 0)$ **then**
    simulate M_i on input 0^m
    if M_i queries a word w of odd length,
     then the answer is yes $\Leftrightarrow w \in V'$
    if M_i queries a word w of even length of form $w = vv$
     then simulate M_U on v,
     and the answer is yes $\Leftrightarrow M_U$ accepts v
    if M_i queries a word w of even length not of form $w = vv$
     then the answer is no
    if M_i rejects 0^m, then $V' = V' \cup \{y\}$,
     where y is the lexicographically first word of length m
     which was not queried in the performed simulation
    endif
  endif
endfor
accepts $x \Leftrightarrow x \in V'$

Figure 4.2 Decision algorithm for V.

From the construction of V, $U \leq_m^p V$, i.e. $V \in H_a^p(\mathcal{C})$. It remains to show that $V \in \mathcal{C}$.

Words of even length do not make problems. Since $U \in \mathcal{C}$, $U \in \mathsf{DTIME}[f]$ for $f \in \mathcal{C}$, and so there exists a deterministic Turing machine M_U such that $U = L(M_U)$ and $f(n)$ bounds its running time. We have to deal with words of odd length, i.e. diagonalization words. Figure 4.2 describes an algorithm for V. For input of odd length n, it constructs $V' = V \cap \Sigma^{\leq n}$ by performing stages $0, 1, \ldots l$ from Figure 4.1 such that $k(l) \leq n$.

Consider time complexity of the algorithm from Figure 4.2. If input has even length n, then we reject immediately or simulate the machine M_U which recognizes U. It takes $O(f(n))$ time. Suppose input has odd length n.

Note that we use the procedure Find from Figure 3.2 in order to shorten the description of the algorithm. The procedure Find for an argument n returns stage $i \leq n$ such that $k(i) = n$ (or return 0 if such a stage does not exist).

We call the procedure Find n times. Since Find works in linear time (see description of the procedure Find in the proof of Theorem 3.7), this takes time $O(n^2)$. This can be done in time $O(n)$ by straight generating of the sequence $k(.)$.

We claim that the procedure Find returns nonzero value at most $l = \log n$ times. It is sufficient to prove that $k(i+1) > 2k(i)$ for every $i > 1$. This is proved in the same way as in the proof of Theorem 3.7. From the construction of V (Figure 4.1), $k(1) = 3$ and $k(2) = 5$. From the condition on $k(i+1)$, $k(i+1) > p_i(k(i)) = k(i)^i + i > k(i)^2 > 2k(i)$ for $i > 1$ (note that $i > 1$ implies $k(i) \geq 5$).

The algorithm from Figure 4.2 simulates at most l deterministic polynomial time Turing machines, i.e. machines $M_1, \ldots, M_l$, on inputs of length at most n. Recall that the running time of the machine M_i is bounded by $p_i(n) = n^i + i$, the running time of the machine M_u is bounded by $f(n)$ and $f(n)$ is non-decreasing. So, we obtain a time bound $O(p_i(n)f(p_i(n)))$ for the machine M_i.

During the simulation of the machine M_i on input $O^{k(i)}$, we need to store all queries of length $k(i)$. This takes time $O(p_i(n))$. After the simulation, we need to find the word $y(i)$ (the first word of length $k(i)$, in lexicographical order, which is not queried during the simulation). This can be done in time $O(np_i(n)\log(np_i(n)))$, which is $O(np_i(n)\log n)$. Since f is, as a time bound, at least linear, this is $O(p_i(n)f(p_i(n)\log n)$.

From $l \leq \log n$, time complexity of the algorithm from Figure 4.2 is $\overline{f}(n) = \log n \sum_{i=1}^{\log n}[p_i(n)f(p_i(n))]$. By assumption $\overline{f}(n) \in \mathcal{F}$, i.e. $\mathsf{DTIME}[\overline{f}(n)] \subseteq \mathcal{C}$. Hence, $V \in \mathcal{C}$. We show before that $V \in H_a^p(\mathcal{C})$, so we obtain $V \in C_a^p(\mathcal{C})$. □

We define the hierarchy of complexity classes and further we apply Theorem 4.20 on it.

Definition 4.21

1. *For natural number* $k \geq 1$, $\mathsf{D}_k = \mathsf{DTIME}[n^{O(\log^k n)}]$.
2. $\mathsf{DH} = \bigcup_k \mathsf{D}_k$.

Corollary 4.22 *If* DH *has a complete problem under* $\leq_a^p$, *then* DH *has a complete problem* V *under* $\leq_a^p$ *with* $V \notin \mathcal{X}$.

Proof We apply Theorem 4.20 with $\mathcal{F} = \cup_{c,k}\{n^{c\log^k n}\}$ and $\mathcal{C} = \mathsf{DH}$. Let the complete problem for DH belongs to $\mathsf{DTIME}[f]$ where f is a form of $n^{c\log^k n}$ for some constants c and k. Consider

$$\begin{aligned}
\overline{f}(n) &= \log n \textstyle\sum_{i=1}^{\log n}[p_i(n)f(p_i(n))] \\
&= \log n \textstyle\sum_{i=1}^{\log n}[(n^i+i)f(n^i+i)] \\
&\leq \log n \textstyle\sum_{i=1}^{\log n}[(c_1n^i)f(c_1n^i)] \\
&= \log n \textstyle\sum_{i=1}^{\log n}[(c_1n^i)(c_1n^i)^{c\log^k(c_1n^i)}] \\
&\leq \log n \textstyle\sum_{i=1}^{\log n}[(c_1n)^{i+ci\log^k(c_1n^i)}] \\
&= \log n \textstyle\sum_{i=1}^{\log n}[(c_1n)^{i+ci(\log^k c_1+i^k\log^k n)}] \\
&\leq (\log^2 n)(c_1n)^{\log n+c\log n(\log^k c_1+\log^k n\log^k n)} \\
&\leq (\log^2 n)(c_1n)^{c_2\log^{2k+1} n} \\
&\leq (c_1n)^{c_3\log^{2k+1} n}
\end{aligned}$$

for constants c_1, c_2 and c_3 and $n > 1$.

So, $\overline{f}(n)$ is a form of $O(n^{O(\log^{2k+1} n)})$ and since $\mathsf{DTIME}[.]$ is closed under $O(.)$, $\overline{f}(n) \in \mathcal{F}$ and we can apply Theorem 4.20.

Note that the exponent of logarithm changes from k to $2k+1$ (but still constant) and this is why this proof does not work for case of D_k. Nevertheless, for union of all D_k, i.e. DH, it works. □

It is not difficult to show that each level of DH has a standard complete problem under $\leq^p_m$-reductions: For natural number k, the complete problem for D_k is $K_k = \{\langle M, x\rangle \mid M$ is a deterministic Turing machine that accepts x, $|x| = n$, in at most $n^{O(\log^k n)}$ steps$\}$.

On the contrary, the next proposition shows that DH does not have complete problems under $\leq^p_T$-reductions.

Proposition 4.23 DH *does not have a complete problem under polynomial time Turing reductions.*

Proof By the Time Hierarchy Theorem 2.10, $\mathsf{D}_i \neq \mathsf{D}_j$ for $i \neq j$. Assume DH has a complete problem U under $\leq^p_T$-reductions with $U \in \mathsf{D}_k$. Denote by d the constant such that $U \in \mathsf{DTIME}[n^{d\log^k n}]$. By the Time Hierarchy Theorem 2.10, there exists $V \in \mathsf{D}_{k+1}$ such that $V \notin \mathsf{D}_k$. From $V \in \mathsf{DH}$ and U is complete for DH , $V \in \mathsf{P}^U$ via the deterministic Turing machine whose running time is bounded by polynomial $p(n)$. Therefore, $V \in \mathsf{DTIME}[p(n) + p(n)^{d\log^k n)}]$ and hence, $V \in \mathsf{DTIME}[n^{O(\log^k n)}]$ which contradicts the assumption that $V \notin \mathsf{D}_k$. □

Note that the proof of nonexistence of complete problems for DH is similar to the proof that polynomial time hierarchy PH does not have a complete

problem unless PH collapses. In case of PH, we do not know whether it collapses or not. In case of DH, we know it does not.

Consider the proof of Theorem 4.20 once again. We never use the fact that the complete problem U for the class $\mathcal{C}$ is complete under polynomial time reductions. We encode U on words of even length to V by doubling the word ($ww \in V \Leftrightarrow w \in U$) and every reduction that is powerful enough to double the input word can be considered in Theorem 4.20 and Corollary 4.22. Hence, if DH has a complete problem under appropriate, possibly not polynomial, reductions, then it also has a complete problem under these reductions does not belong to $\mathcal{X}$.

Consider for example many-one reductions that can be computed in deterministic time $O(n^{O(\log^{O(c)} n)})$. Let us denote such reductions by $\leq_m^{dh}$. There is a standard complete problem for DH under $\leq_m^{dh}$-reductions, namely $K_{dh} = \{\langle M, x, k\rangle \mid M$ is a deterministic Turing machine that accepts x, $|x| = n$, in at most $n^{O(\log^k n)}$ steps$\}$. From Theorem 4.20, there exists a complete problem for DH under $\leq_m^{dh}$-reductions that does not belong to $\mathcal{X}$.

4.4 Hard problems of complexity classes

We have seen that complete problems for strong complexity classes like for example PSPACE, DEXT, EXP, EXPSPACE and others belong to $\mathcal{X}$. Now we show that this does not generally hold for hard problems for these classes. It means that hardness is not sufficient for sets being in $\mathcal{X}$. We use the result by Hartmanis [Har85] which simply said states that a relativization method can be relativized, i.e. relativization works for relativized computations too.

Lemma 4.24 ([Har85]) *For every oracle C, there exists oracles A and B such that $\mathsf{P}^{C\oplus A} = \mathsf{NP}^{C\oplus A}$ and $\mathsf{P}^{C\oplus B} \neq \mathsf{NP}^{C\oplus B}$.*

Proof First we prove the existence of A. Let be $A = KS(C)$, i.e. a standard complete problem under $\leq_m^p$-reductions from Definition 2.30. We want to show that $\mathsf{P}^{C\oplus A} = \mathsf{NP}^{C\oplus A}$.

Clearly,

$$\mathsf{P}^{C\oplus A} = \mathsf{P}^{C\oplus KS(C)} \subseteq \mathsf{NP}^{C\oplus KS(C)} \subseteq \mathsf{PSPACE}^{C\oplus KS(C)}.$$

Further,

$$\mathsf{P}^{C\oplus KS(C)} = \mathsf{P}^{KS(C)} = \mathsf{PSPACE}^{C}$$

and

$$\mathsf{PSPACE}^{C\oplus KS(C)} = \mathsf{PSPACE}^{KS(C)} = \mathsf{PSPACE}^{C}.$$

It follows that in the foregoing string of inclusions all inclusions are actually equalities.

To show the existence of B such that $\mathsf{P}^{C\oplus B} \neq \mathsf{NP}^{C\oplus B}$, we use standard Baker-Gill-Solovay [BGS75] technique, but instead of diagonalization against P we need to perform diagonalization against P^C.

Recall that $L(B) = \{0^n \mid (\exists x \in B)[|x| = n]\}$. Clearly, $L(B) \in \mathsf{NP}^B$ for every B, and hence $L(B) \in \mathsf{NP}^B \subseteq \mathsf{NP}^{C\oplus B}$.

Recall the enumeration $\{M_i\}_{i\geq 1}$ from Fact 2.27.

We construct B in stages. Denote by $k(n)$ an increasing sequence of natural numbers: $k(n)$ is the length of word which is used in the n-th stage to ensure that M_n does not accept $L(B)$. After n stages, we denote the so far constructed oracle set by B_{n-1}. In the n-th stage, we add at most one word of length $k(n)$ to ensure that $L(B) \neq L(M_n, C \oplus B)$. We show that either $0^{k(n)} \in L(M_n, C \oplus B)$ and $B \cap \Sigma^{=k(n)} = \emptyset$, or $0^{k(n)} \notin L(M_n, C \oplus B)$ and $B \cap \Sigma^{=k(n)} \neq \emptyset$. Figure 4.3 presents the n-the stage in the construction of B. The final oracle $B = \cup_n B_n$.

---

stage 0:
  $k(0) = 0$
  $B_0 = \emptyset$
stage n:
  Let $k(n)$ be the smallest natural number such that
    $k(n) > p_{n-1}(k(n-1))$ and $p_n(k(n)) < 2^{k(n)}$.
  If $0^{k(n)} \in L(M_n, C \oplus B_{n-1})$ then $B_n = B_{n-1}$.
  If $0^{k(n)} \notin L(M_n, C \oplus B_{n-1})$ then $B_n = B_{n-1} \cup \{y(n)\}$,
  where $y(n)$ is the first word, in lexicographic order, of length $k(n)$
  such that $1y(n)$ is not queried in the computation of M_n on input $0^{k(n)}$
  with the oracle $C \oplus B_{n-1}$.

Figure 4.3 Construction of B such that $L(B) \notin \mathsf{P}^{C\oplus B}$.

From the condition on $k(n)$, an appropriate word $y(n)$ of length $k(n)$ in case (b) always exists if needed. There are $2^{k(n)}$ words of length $k(n)$ and $p_n(k(n)) < 2^{k(n)}$.

The condition on $k(n)$ also ensures that $k(n)$ is long enough not to disturb, by possible adding of $y(n)$ to B, any computation from the previous stages. It means that $0^{k(n)} \in L(M_n, C \oplus B_{n-1}) \Leftrightarrow 0^{k(n)} \in L(M_n, C \oplus B)$. This holds for every n. Therefore, $0^{k(n)} \in L(B) \Leftrightarrow 0^{k(n)} \notin L(M_n, C \oplus B)$ for every n. That means $L(B) \notin \mathsf{P}^{C\oplus B}$. Since $L(B) \in \mathsf{NP}^{C\oplus B}$, we get $\mathsf{P}^{C\oplus B} \neq \mathsf{NP}^{C\oplus B}$. □

We show why the same technique does not give us a similar result for union. Recall the construction of the separating oracle from the proof of Lemma 4.24. A potential problem is this one: In the n-th stage we have $B(n-1)$ from the previous stages. We want to diagonalize against $\mathsf{P}^{C \cup B(n-1)}$ by adding of one string of particular length m to $B(n-1)$ or leaving $B(n-1)$ unchanged. But there can be words of length m in C and this might cause problems.

We can say even more. A similar result for operation union can not hold without solving the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem. Consider an alphabet Σ and $X = \Sigma^*$, i.e. X is the set of all words over Σ. Clearly, $X \cup Y = X$ holds for all $Y \subseteq \Sigma^*$. Since X is obviously in P, a result like for X exists Y such that $\mathsf{P}^{X \cup Y} \neq \mathsf{NP}^{X \cup Y}$ would immediately separate P from NP by Theorem 4.2. Similarly, a result like for X exists Y such that $\mathsf{P}^{X \cup Y} = \mathsf{NP}^{X \cup Y}$ would immediately collapse NP to P from Theorem 4.2.

The previous lemma has simple corollary: Every set can be encoded into an oracle that is not in $\mathcal{X}$, i.e. that separates P from NP relativized.

Corollary 4.25 *For every oracle C, there exists an oracle D with $D \notin \mathcal{X}$ such that $C \leq_m^p D$.*

Proof Given oracle C, take oracle B which is ensured by Lemma 4.24, i.e. $C \oplus B \notin \mathcal{X}$. Desired oracle $D = C \oplus B$. Clearly, $C \leq_m^p C \oplus B$ and hence, $C \leq_m^p D$. □

Finally, we can prove that not all hard problems of complexity classes belong to $\mathcal{X}$.

Theorem 4.26 *Let C be a complexity class and $U \in C_a^b(\mathsf{C})$. Then, there exists $V \in H_a^b(\mathsf{C})$ with $U \leq_m^p V$ and $V \notin \mathcal{X}$.*

Proof By Lemma 4.24, there exists B such that $U \oplus B \notin \mathcal{X}$. From $U \in C_a^b(\mathsf{C})$, $V = U \oplus B \in H_a^b(\mathsf{C})$. □

Theorem 4.27 *$\mathcal{X}$ is not closed under $\leq_m^p$ upward ($A \leq_m^p B$ and $A \in \mathcal{X} \not\Rightarrow B \in \mathcal{X}$).*

Proof From Theorem 3.3, there exists $A \in \mathcal{X}$. From Lemma 4.24, there exists B such that $A \oplus B \notin \mathcal{X}$. □

Simple corollary is that $\mathcal{X}$ cannot be closed upward under stronger reductions than many-one, for example bounded truth-table, truth-table and Turing reductions.

Let us mention a different proof of Theorem 4.27. Take an arbitrary $A \in \mathcal{X}$. Clearly, $A \leq_m^p A \oplus L$ for an arbitrary L. But $\mu(\{A \oplus L \mid$

L arbitrary}) > 0 and that violates the statement of Proposition 4.4 which claims $\mu(\mathcal{X}) = 0$.

Finally, note why the result of Corollary 4.25 is not in conflict with the previous results that complete problems for exponential time complexity classes belong to $\mathcal{X}$.

Consider $C \in C_T^p(\mathsf{EXP})$. By Corollary 4.25, there exists an oracle D such that $C \leq_m^p D$, i.e. $D \in H_m^p(\mathsf{EXP})$, and $D \notin \mathcal{X}$. Construction of D from Lemma 4.24 does not ensure that $D \in \mathsf{EXP}$ (which would mean $D \in C_m^p(\mathsf{EXP})$).

Consider the algorithm from Figure 4.2, which was used in the proof of Theorem 4.20. In the simulation of the machine M_i , we have to deal with queries of even length which encode the set U. For D we can use an algorithm similar to the one from Figure 4.2. In the simulation of the machine M_i, we have to deal with queries which encode the set $C \in C_m^p(\mathsf{EXP})$. Let $C \in \mathsf{DTIME}[2^{p(n)}]$ form some polynomial $p(n)$. Then the complexity of the above described algorithm for D is $\log n \Sigma_{i=1}^{\log n}[p_i(n) f(p_i(n))] = \log n \Sigma_{i=1}^{\log n}[p_i(n) 2^{p(p_i(n))}]$, which is not bounded by any function of form $2^{q(n)}$ for some fixed polynomial $q(n)$.

4.5 Characterization of $\mathcal{X}$

We have seen that complete problems for strong complexity classes belong to $\mathcal{X}$. What do these sets have in common? This section provides two characterizations of the class $\mathcal{X}$. First we show that sets in $\mathcal{X}$ are exactly those sets A which are hard for NP^A and even for PH^A under Turing reductions.

Theorem 4.28 (Characterization of $\mathcal{X}$ I.) *Following statements are equivalent:*

1. $A \in \mathcal{X}$.
2. $A \equiv_T^p K(A)$.
3. $A \in H_T^p(\mathsf{NP}^A)$.
4. $A \in H_T^p(\mathsf{PH}^A)$.

Proof From Lemma 3.2 ($\mathsf{P}^A = \mathsf{NP}^A \Leftrightarrow K(A) \in \mathsf{P}^A$) and the fact that $A \leq_T^p K(A)$ for every A, the statements (1) and (2) are equivalent.

From Lemma 3.1 ($K(A) \in C_m^p(\mathsf{NP}^A)$) the statements (2) and (3) are equivalent.

The statement (4) follows from (1) and (3). In that case PH^A collapses to the first level $\Sigma_1^{p,A} = \mathsf{NP}^A$.

From $K(A) \in \mathsf{PH}^A$ for every A, the statement (4) implies (2). □

Note that from $A \in \mathsf{NP}^A$ for every A, completeness and hardness coincides in case of NP^A. Therefore, we can replace $H^p_T(\mathsf{NP}^A)$ in Theorem 4.28 (3), by $C^p_T(\mathsf{NP}^A)$. Similarly, with $C^p_T(\mathsf{PH}^A)$ in the statement (4).

Note also that $K(A)$ can be substituted in the second statement of Theorem 4.28 by an arbitrary problem from $C^p_T(\mathsf{NP}^A)$.

Another interesting characterization connects $\mathcal{X}$ with the extended low and high hierarchies. First of all, we have the folloving basic observation.

Proposition 4.29 $A \in \mathcal{X} \Rightarrow A \in \mathsf{EL}_1$.

Proof Let $A \in \mathcal{X}$, i.e. $\mathsf{P}^A = \mathsf{NP}^A$. Clearly, $\mathsf{NP}^A \subseteq \mathsf{P}^A \subseteq \mathsf{P}^{A \oplus SAT}$. □

Does the converse statement hold? Because sets from EL_1 provide a positive relativization of the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem this would immediately give a collapse of NP to P. Showing the existence of $A \in \mathsf{EL}_1$ such that $A \notin \mathcal{X}$ would give a separation P from NP. However, we can prove the converse statement with additional an assumption that A is hard for NP under Turing reductions.

Proposition 4.30 $A \in \mathcal{X} \Leftrightarrow (A \in \mathsf{EL}_1 \wedge SAT \leq^p_T A)$.

Proof

"$\Rightarrow$" :

Let $A \in \mathcal{X}$, i.e. $\mathsf{P}^A = \mathsf{NP}^A$. By the previous proposition $A \in \mathsf{EL}_1$. From $SAT \in \mathsf{NP}^A = \mathsf{P}^A$, it follows that $SAT \leq^p_T A$.

"$\Leftarrow$" :

Let $A \in \mathsf{EL}_1$ and $SAT \leq^p_T A$. The former one means that $\mathsf{NP}^A \leq^p_T A \oplus SAT$. Since $SAT \leq^p_T A$, we get $A \oplus SAT \leq^p_T A$. Together, $\mathsf{NP}^A \leq^p_T A$ and hence $\mathsf{P}^A = \mathsf{NP}^A$. □

An interesting question about EL_1 is whether there exists $A \in \mathsf{EL}_1$ such that A is Turing incomparable with SAT. The just proved proposition shows that this question is hard to solve: If such A did exist, then it would separate P from NP because EL_1 yields a positive relativization of the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem. Even showing an existence of $A \in \mathsf{EL}_1$ which is not hard for NP, i.e. $SAT \not\leq^p_T A$, would separate P from NP.

Corollary 4.31 *$\mathcal{X}$ is closed under disjoint unions $\Leftrightarrow$ EL_1 is closed under disjoint unions.*

Proof Note that hard sets for NP under Turing reductions are closed under disjoint unions. Precisely, if $SAT \leq^p_T A$ and $SAT \leq^p_T B$, then $SAT \leq^p_T A \oplus B$

(note that only one of the statements $SAT \leq_T^p A$ and $SAT \leq_T^p B$ is enough to guarantee $SAT \leq_T^p A \oplus B$). Proposition 4.30 finishes the proof. □

As we mention in Section 4.2, the class $\mathcal{X}$ is not known to be either closed or not closed under disjoint unions. Equivalence from Corollary 4.31 shows that solving the relationship between $\mathcal{X}$ and a disjoint union would immediately give us a relationship between EL_1 and a disjoint union. Unfortunately, this question remains still open.

Next we prove that sets in the zeroth level of the extended high hierarchy are exactly those sets which are hard for NP.

Proposition 4.32 $A \in \mathsf{EH}_0 \Leftrightarrow SAT \leq_T^p A$.

Proof

"$\Rightarrow$" :

If $A \in \mathsf{EH}_0$, then $SAT \in \mathsf{P}^{A \oplus SAT} \subseteq \mathsf{P}^A$ and hence $SAT \leq_T^p A$.

"$\Leftarrow$" :

If $SAT \leq_T^p A$, then there exists a deterministic polynomial time Turing machine M with an oracle such that $SAT = L(M, A)$. Consider an arbitrary $B \in \mathsf{P}^{A \oplus SAT}$, then there exists a deterministic polynomial time Turing machine M_1 with an oracle such that $B = L(M_1, A \oplus SAT)$. Let us define a Turing machine M_2 such that on input x:

1. Simulate M_1 on input x until M_1 asks an oracle a word y or accepts or rejects.
2. Accept whenever M_1 accepts, reject whenever M_1 rejects and stops.
3. If M_1 asks $y = 0y'$, then M_2 asks y' and the answer is yes if and only if the answer for M_2 is yes and we continue on the simulation in Step 1.
4. If M_1 asks $y = 1y'$, then we simulate M on input y and the answer is yes exactly when M accepts y'. Then we continue on the simulation in Step 1.

If the oracle of M_2 is A, then M_2 accepts B, i.e. $B = L(M_2, A)$. M_2 is deterministic because M and M_1 are deterministic and uses polynomial time because both M and M_1 work in polynomial time. Thus $B \in \mathsf{P}^A$. Assuming $SAT \leq_T^p A$, we show that $\mathsf{P}^{A \oplus SAT} \subseteq \mathsf{P}^A$. Therefore, $A \in \mathsf{EH}_0$ and the statement is proved. □

Finally, we have proved all needed propositions to present connection between sets in $\mathcal{X}$ and the extended hierarchies.

Theorem 4.33 (Characterization of $\mathcal{X}$ II.)

$$A \in \mathcal{X} \Leftrightarrow A \in \mathsf{EL}_1 \cap \mathsf{EH}_0.$$

Proof Immediately from Propositions 4.30 and 4.32. □

We investigate the relationship between $\mathcal{X}$ and the extended hierarchies later.

4.6 $\mathcal{X}$ is not closed under $\cap$, $\cup$ and $\triangle$

In Section 4.2, we show basic structural properties of the class $\mathcal{X}$. Here we continue with more involved properties. We prove that $\mathcal{X}$ is not closed under any of these operations: unions, intersections and symmetric differences.

Proofs for all three operations are similar. We formulate all of them in one theorem. In the proof, we use the following notation.

Definition 4.34 (coding convention)

1. *For a word β, let β' be the word β without the last symbol (bit) and let β'' be the word β without the last two symbols (bits).*
2. *We say that a word is* triplicate *if it consists of a sequence of triples where all three bits are the same* (0 *or* 1) *for each particular triple.*
3. *Define coding function h as follows: $h(0) = 000$ and $h(1) = 111$. We extend this definition to the words $h(a_1a_2 \cdots a_k) = h(a_1)h(a_2) \cdots h(a_k)$ and to the sets $h(B) = \{\alpha \mid (\exists \beta \in B)[h(\beta) = \alpha]\}$.*

Consider the construction of $B \notin \mathcal{X}$ from the proof of Theorem 3.6. Recall that $L(B) = \{0^n \mid (\exists x \in B)[|x| = n]\} \in \mathsf{NP}^B \setminus \mathsf{P}^B$, and by Theorem 3.7, $B \in \mathsf{DEXT}$.

Theorem 4.35 *$\mathcal{X}$ is not closed under intersections, unions and symmetric differences.*

Proof We prove that $\mathcal{X}$ is not closed under all three operations simultaneously. Take an arbitrary $K \in C_m^p(\mathsf{DEXT})$. Our goal is to encode both sets B and K into X_i and into Y_i to ensure that $X_i \in \mathcal{X}$ and $Y_i \in \mathcal{X}$, $1 \leq i \leq 3$ but $X_1 \cap Y_1 \notin \mathcal{X}$, $X_2 \cup Y_2 \notin \mathcal{X}$, $X_3 \triangle Y_3 \notin \mathcal{X}$.

Before giving the formal definition of the sets X_i and Y_i, $1 \leq i \leq 3$, we show the idea of the proof. It is represented by Figures 4.4 - 4.6. Figure 4.4 shows how X_1 and Y_1 are constructed. Both X_1 and Y_1 are divided into three disjoint sections. The first section encodes the words of length $3k$, the second section encodes the words of length $3k+1$ and finally the third section encodes the words of length $3k + 2$. In the first section, we encode the set B in both X_1 and Y_1. The second section of X_1 encodes the set K and the

third section of X_1 is empty. The second section of Y_1 is empty and the third section of Y_1 encodes the set K. The set $X_1 \cap Y_1$ has both the second and the third sections empty and the first section encodes the set B. It is not difficult to show that if the set $X_1 \cap Y_1$, which encodes the set B, was in $\mathcal{X}$, then the set $L(B)$ would be in P^B which is a contradiction to our choice of B.

The same idea is used to show that $\mathcal{X}$ is not closed under unions. Look at Figure 4.5. In this case the set $X_2 \cup Y_2$ encodes the set B in the first section and other two sections contain all words of given length. Again, is is not difficult to show that if the set $X_2 \cup Y_2$, which encodes the set B, was in $\mathcal{X}$, then the set $L(B)$ would be in P^B which is a contradiction to our choice of B.

Figure 4.6 shows the same for the case of symmetric difference.

$\mathbf{X_1}$		$\mathbf{Y_1}$		$\mathbf{X_1 \cap Y_1}$
B	$\cap$	B	$=$	B
K		$\emptyset$		$\emptyset$
$\emptyset$		K		$\emptyset$

Figure 4.4 $X_1, Y_1 \in \mathcal{X}$ but $X_1 \cap Y_1 \notin \mathcal{X}$.

$\mathbf{X_2}$		$\mathbf{Y_2}$		$\mathbf{X_2 \cup Y_2}$
B	$\cup$	$\emptyset$	$=$	B
K		coK		Σ^*
coK		K		Σ^*

Figure 4.5 $X_2, Y_2 \in \mathcal{X}$ but $X_2 \cup Y_2 \notin \mathcal{X}$.

$\mathbf{X_3}$		$\mathbf{Y_3}$		$\mathbf{X_3 \triangle Y_3}$
B	$\triangle$	$\emptyset$	$=$	B
$\emptyset$		$\emptyset$		$\emptyset$
K		K		$\emptyset$

Figure 4.6 $X_3, Y_3 \in \mathcal{X}$ but $X_3 \triangle Y_3 \notin \mathcal{X}$.

Define the sets X_i, $1 \leq i \leq 3$, as follows.

$|\beta| = 3k + 0$:
$\beta \in X_i \Leftrightarrow (\exists \alpha \in B)[h(\alpha) = \beta]$ $(1 \leq i \leq 3)$
$|\beta| = 3k + 1$:
$\beta \in X_i \Leftrightarrow (\exists \alpha \in K)[h(\alpha) = \beta']$ $(1 \leq i \leq 2)$
$(\forall \beta)[\beta \notin X_3]$
$|\beta| = 3k + 2$:
$(\forall \beta)[\beta \notin X_1]$
$\beta \in X_2 \Leftrightarrow (\exists \alpha \notin K)[h(\alpha) = \beta'']$
$\beta \in X_3 \Leftrightarrow (\exists \alpha \in K)[h(\alpha) = \beta'']$

Definition of X_i, $1 \leq i \leq 3$, by the length of word β.

Define the sets Y_i, $i \leq i \leq 3$, as follows.

$|\beta| = 3k + 0$:
$\beta \in Y_1 \Leftrightarrow (\exists \alpha \in B)[h(\alpha) = \beta]$
$(\forall \beta)[\beta \notin Y_i]$ $(2 \leq i \leq 3)$
$|\beta| = 3k + 1$:
$(\forall \beta)[\beta \notin Y_i]$ $(i = 1, 3)$
$\beta \in Y_2 \Leftrightarrow (\exists \alpha \notin K)[h(\alpha) = \beta']$
$|\beta| = 3k + 2$:
$\beta \in Y_i \Leftrightarrow (\exists \alpha \in K)[h(\alpha) = \beta'']$ $(1 \leq i \leq 3)$

Definition of Y_i, $1 \leq i \leq 3$, by the length of word β.

Claim 1 *X_i, $Y_i \in \mathcal{X}, 1 \leq i \leq 3$.*

Proof From the definitions of X_i and Y_i, $K \leq^p_m X_i$ and $K \leq^p_m Y_i$, $1 \leq i \leq 3$. Since $B \in \mathsf{DEXT}$ and K is complete for DEXT under $\leq^p_m$-reductions, $X_i \leq^p_m K$ and $Y_i \leq^p_m K$, $i = 1, 3$. In X_2 and Y_2, also *coK* is encoded. But since $K \in C^p_m(\mathsf{DEXT})$, clearly $coK \in \mathsf{DEXT}$ and therefore, there exist a $\leq^p_m$-reduction from *coK* to K. Hence, $X_2 \leq^p_m K$ and $Y_2 \leq^p_m K$. Together, $X_i \equiv^p_m K$ and $Y_i \equiv^p_m K$, $1 \leq i \leq 3$. □

Denote by $B_1 = X_1 \cap Y_1$, $B_2 = X_2 \cup Y_2$ and by $B_3 = X_3 \cap Y_3$.

Claim 2 $B_1 = h(B) = \{\alpha \mid (\exists \beta \in B)[h(\beta) = \alpha]\}$.

Proof From the definitions of X_1 and Y_1, there are no words of length $3k+1$ and $3k+2$ in B_1. The only words in B_1 are of length $3k$. Clearly, $B_1 = h(B) = \{\alpha \mid (\exists \beta \in B)[h(\beta) = \alpha]\}$. □

Claim 3

$$\begin{aligned} B_2 = \quad & \{\alpha \mid |\alpha| = 3k \wedge (\exists \beta \in B)[h(\beta) = \alpha]\} \\ \cup \quad & \{wb \mid |w| = 3k \wedge b \in \{0,1\} \wedge w \textit{ is triplicate}\} \\ \cup \quad & \{wb_1b_2 \mid |w| = 3k \wedge b_1 \in \{0,1\} \wedge b_2 \in \{0,1\} \wedge w \textit{ is triplicate}\}. \end{aligned}$$

Proof From the definitions of X_2 and Y_2, words of length $3k$ in B_2 are exactly those from the claim. From the definitions of X_2 and Y_2, words of length $3k+1$ in B_2 are $\{\beta \mid (\exists \alpha \in K)[h(\alpha) = \beta']\} \cup \{\beta \mid (\exists \alpha \notin K)[h(\alpha) = \beta']\}$. Those are exactly all triplicate words of length $3k$ with appended one more bit at the end. From the definitions of X_2 and Y_2, words of length $3k+2$ in B_2 are $\{\beta \mid (\exists \alpha \notin K)[h(\alpha) = \beta'']\} \cup \{\beta \mid (\exists \alpha \in K)[h(\alpha) = \beta']\}$. Those are exactly all triplicate words of length $3k$ with appended two more bits at the end. □

Claim 4 $B_3 = h(B) = \{\alpha \mid (\exists \beta \in B)[h(\beta) = \alpha]\}$.

Proof From the definitions of X_3 and Y_3, words of length $3k+1$ in B_3 are $(\emptyset \cup \emptyset) \setminus (\emptyset \cap \emptyset) = \emptyset$, i.e. there are no words of length $3k+1$ in B_3. From the definitions of X_3 and Y_3, words of length $3k+2$ in B_3 are $(K \cup K) \setminus (K \cap K) = \emptyset$, i.e. there are no words of length $3k+2$ in B_3. The only words in B_3 are of length $3k$. Those are $h(B) = \{\alpha \mid (\exists \beta \in B)[h(\beta) = \alpha]\}$. □

Claim 5 B_1, B_2, $B_3 \notin \mathcal{X}$.

Proof From Claim 2 and 3, $B_1 = B_3$, so we need to prove only B_1, $B_2 \notin \mathcal{X}$. Suppose the opposite. We show that $B_i \in \mathcal{X}$ implies $L(B) \in \mathsf{P}^B$, $1 \leq i \leq 2$, which is a contradiction to our choice of B. For $1 \leq i \leq 2$, define:

$$L_h(B_i) = \{0^n \mid n = 3k \wedge (\exists \beta \in B_i)[|\beta| = n]\}.$$

Clearly, $L_h(B_i) \in \mathsf{NP}^{B_i}$, and therefore $L_h(B_i) \in \mathsf{P}^{B_i}$ by our assumption, $1 \leq i \leq 2$. Hence, there exist deterministic polynomial time Turing machines M_i with oracles such that $L_h(B_i) = L(M_i, B_i)$, $1 \leq i \leq 2$.

Define Turing machines M_1' such that on input x:

1. If input x is not a form of 0^n, then reject.
2. If input x is a form of 0^n, then simulate M_1 on input 0^{3n} until M_1 asks an oracle a word w or accepts or rejects.

3. Accept whenever M_1 accepts, reject whenever M_1 rejects and stops.
4. If M_1 asks w which is not a triplicate word, then the answer is no and we continue on the simulation in Step 2.
5. If M_1 asks w which is a triplicate word, then M_1' asks $h^{-1}(w)$ and the answer is yes if and only if the answer for M_1' is yes and we continue on the simulation in Step 2.

M_1' is deterministic because M_1 is deterministic and uses polynomial time because M_1 works in polynomial time. If the oracle of M_1' is B, then M_1' accepts $L(B)$, i.e. $L(B) = L(M_1', B)$. Thus $L(B) \in \mathsf{P}^B$, which is a contradiction to our choice of B, so $B_1 \notin \mathcal{X}$.

Define Turing machines M_2' such that on input x:

1. If input x is not a form of 0^n, then reject.
2. If input x is a form of 0^n, then simulate M_2 on input 0^{3n} until M_2 asks an oracle a word w or accepts or rejects.
3. Accept whenever M_2 accepts, reject whenever M_2 rejects and stops.
4. If M_2 asks w of length $3k$, and w is not a triplicate word, then the answer is no. Otherwise, M_2' asks $h^{-1}(w)$ and the answer is yes if and only if the answer for M_2' is yes. Then we continue on the simulation in Step 2.
5. If M_2 asks w of length $3k+1$, and $w = vb$ for v triplicate and $b \in \{0,1\}$, then the answer is yeas. Otherwise, the answer is no. Then we continue on the simulation in Step 2.
6. If M_2 asks w of length $3k+2$, and $w = vb_1b_2$ for v triplicate and $b_1, b_2 \in \{0,1\}$, then the answer is yeas. Otherwise, the answer is no. Then we continue on the simulation in Step 2.

M_2' is deterministic because M_2 is deterministic and uses polynomial time because M_2 works in polynomial time. If the oracle of M_2' is B, then M_2' accepts $L(B)$, i.e. $L(B) = L(M_2', B)$. Thus $L(B) \in \mathsf{P}^B$, which is a contradiction to our choice of B, so $B_2 \notin \mathcal{X}$. Claim 5 is proved. □

(Theorem 4.35) □

4.7 $\mathcal{X}$ and the extended hierarchies

Recall Theorem 4.33, which claims $A \in \mathcal{X} \Leftrightarrow A \in \mathsf{EL}_1 \cap \mathsf{EH}_0$. From Fact 2.26, $\mathsf{EL}_1 \cap \mathsf{EH}_0 \subseteq \mathsf{EL}_1 \cap \mathsf{EH}_1$. Are there some conditions under which $\mathsf{EL}_1 \cap$

$\mathsf{EH}_0 = \mathsf{EL}_1 \cap \mathsf{EH}_1$ (or $\mathsf{EL}_1 \cap \mathsf{EH}_0 \neq \mathsf{EL}_1 \cap \mathsf{EH}_1$)? We show that $\mathsf{EL}_1 \cap \mathsf{EH}_0 = \mathsf{EL}_1 \cap \mathsf{EH}_1$ provided the polynomial time hierarchy does not collapse and a special hypothesis holds.

The next proposition connects $\mathsf{EL}_1 \cap \mathsf{EH}_1$ with the class $\mathcal{X}$.

Proposition 4.36 $A \in \mathsf{EL}_1 \cap \mathsf{EH}_1 \Rightarrow A \oplus SAT \in \mathcal{X}$.

Proof From the definition of EL_1, $\mathsf{NP}^A \subseteq \mathsf{P}^{A \oplus SAT}$. From the definition of EH_1, $\mathsf{NP}^{A \oplus SAT} \subseteq \mathsf{NP}^A$. Combining these facts with $\mathsf{P}^{A \oplus SAT} \subseteq \mathsf{NP}^{A \oplus SAT}$, we obtain $\mathsf{P}^{A \oplus SAT} = \mathsf{NP}^{A \oplus SAT} = \mathsf{NP}^A$, i.e. $A \oplus SAT \in \mathcal{X}$. □

The next corollary shows that $\mathcal{X}$ is closed under joins with SAT.

Corollary 4.37 $A \in \mathcal{X} \Rightarrow A \oplus SAT \in \mathcal{X}$.

Proof Take an arbitrary $A \in \mathcal{X}$. From Theorem 4.33, $A \in \mathsf{EL}_1 \cap \mathsf{EH}_0$. From Fact 2.26, $\mathsf{EH}_0 \subseteq \mathsf{EH}_1$. So, $A \in \mathsf{EL}_1 \cap \mathsf{EH}_1$, and $A \oplus SAT \in \mathcal{X}$ by Proposition 4.36. □

We are interested in the opposite implication. The first step is the following proposition.

Proposition 4.38 $A \oplus SAT \in \mathcal{X} \Rightarrow A \in \mathsf{EL}_1$.

Proof From $A \oplus SAT \in \mathcal{X}$, $A \oplus SAT \in \mathsf{EL}_1 \cap \mathsf{EH}_0$ by Theorem 4.33. Since $A \oplus SAT \in \mathsf{EL}_1$, $\mathsf{NP}^{A \oplus SAT} \subseteq \mathsf{P}^{(A \oplus SAT) \oplus SAT}$. Clearly, $\mathsf{P}^{A \oplus SAT} \equiv_T^p \mathsf{P}^{(A \oplus SAT) \oplus SAT}$. Since $\mathsf{NP}^A \subseteq \mathsf{NP}^{A \oplus SAT}$, $\mathsf{NP}^A \subseteq \mathsf{P}^{A \oplus SAT}$, i.e. $A \in \mathsf{EL}_1$. □

Corollary 4.39 $A \in \mathsf{EH}_0 \wedge A \oplus SAT \in \mathcal{X} \Rightarrow A \in \mathcal{X}$.

Proof From assumptions and Proposition 4.38, $A \in \mathsf{EH}_0 \cap \mathsf{EL}_1$. Therefore, $A \in \mathcal{X}$ by Theorem 4.33. □

From Theorem 4.33, we know that all sets from $\mathcal{X}$ belong to EH_0. One can ask whether the first assumption of Corollary 4.39 can be omitted, i.e. whether holds $A \oplus SAT \in \mathcal{X} \Rightarrow A \in \mathcal{X}$ for every A.

Let us analyse the relationship between A and the polynomial time hierarchy PH provided $A \oplus SAT \in \mathcal{X}$.

1. $A \in \mathsf{PH}$: Since $A \in \mathsf{PH}$, $A \oplus SAT \in \mathsf{PH}$. From $A \oplus SAT \in \mathcal{X}$ and Proposition 4.8, PH collapses.
2. $A \notin \mathsf{PH}$ and $SAT \leq_T^p A$: Clearly, $A \leq_T^p A \oplus SAT$ for every A. Since $SAT \leq_T^p A$, $A \oplus SAT \leq_T^p A$. Together, $A \oplus SAT \equiv_T^p A$, and $A \in \mathcal{X}$ from Theorem 4.6 ($\mathcal{X}$ is closed under $\equiv_T^p$).

3. $A \notin \mathsf{PH}$ and $SAT \not\leq_T^p A$: Since $SAT \not\leq_T^p A$, even $\mathsf{PH} \not\leq_T^p A$. But From Theorem 4.28, if A was in $\mathcal{X}$, then A would be in $H_T^p(\mathsf{PH}^A)$. Therefore, $\mathsf{PH} \subseteq \mathsf{PH}^A \leq_T^p A$. That means that A can not be in $\mathcal{X}$. The question is if this case can actually happen.

Hypothesis T: There is not a set A such that $A \oplus SAT \in \mathcal{X}$, $A \notin \mathsf{PH}$ and $SAT \not\leq_T^p A$. Note a special case of Hypothesis T: There is not a set A such that $A \oplus SAT \in \mathsf{EL}_1$ and $SAT \not\leq_T^p A$.

The foregoing analysis proves the following proposition.

Proposition 4.40 *If* PH *does not collapse and Hypothesis T holds, then* $A \oplus SAT \in \mathcal{X} \Rightarrow A \in \mathcal{X}$.

Now we can prove the main theorem of this section.

Theorem 4.41 *If* PH *does not collapse and Hypothesis T holds, then* $\mathsf{EL}_1 \cap \mathsf{EH}_0 = \mathsf{EL}_1 \cap \mathsf{EH}_1$.

Proof From Fact 2.26, the inclusion from left to right follows immediately. We prove the inclusion from right to left. Take an arbitrary $L \in \mathsf{EL}_1 \cap \mathsf{EH}_1$. From proposition 4.36, $L \oplus SAT \in \mathcal{X}$. Assumptions of Proposition 4.40 are fulfilled, we get $L \in \mathcal{X}$. From Theorem 4.33, $L \in \mathsf{EL}_1 \cap \mathsf{EH}_0$. □

4.8 Sets reducible to $\mathcal{X}$

As far as we know, Balcázar [Bal84], as the first one, proposed to investigate the oracles relative to which $\mathsf{P} = \mathsf{NP}$. One idea mentioned in [Bal84] is to investigate the following class.

Definition 4.42 ([Bal84])

$$H = \bigcap_{O \in \mathcal{X}} R_T^p(O) = \{L \mid (\forall O \in \mathcal{X})[L \in \mathsf{P}^O]\}.$$

In the same paper, Balcázar claims a few simple propositions without proofs.

Proposition 4.43 ([Bal84])

1. $\mathsf{PH} \subseteq H \subseteq \mathsf{PSPACE}$.
2. $H = \mathsf{PSPACE} \Rightarrow (\forall Q \in \mathsf{PSPACE})[Q \in \mathcal{X} \Rightarrow Q \in C_T^p(\mathsf{PSPACE})]$.

Proof

1. Take an arbitrary $A \in \mathcal{X}$, i.e. $\mathsf{P}^A = \mathsf{NP}^A$. From Theorem 4.28 (4), $\mathsf{PH}^A \leq_T^p A$. Clearly, $\mathsf{PH} \subseteq \mathsf{PH}^A$ and we obtain $\mathsf{PH} \leq_T^p A$, and so $\mathsf{PH} \subseteq H$.

   Take an arbitrary complete problem Q for PSPACE under $\leq_T^p$-reductions (f.e. QBF). By Theorem 3.4, $Q \in \mathcal{X}$, and obviously $\mathsf{P}^Q = \mathsf{PSPACE}$. From the definition of the class H, $H \subseteq \mathsf{P}^Q = \mathsf{PSPACE}$.

2. Assume $H = \mathsf{PSPACE}$. Take an arbitrary $A \in \mathsf{PSPACE}$ with $A \in \mathcal{X}$. From the definition of H, we get $\mathsf{PSPACE} = H \subseteq \mathsf{P}^A$, i.e A is hard for PSPACE. Since A is from PSPACE, we get $A \in C_T^p(\mathsf{PSPACE})$.

□

Regan [Reg83] also investigated the class H and proved the following proposition.

Proposition 4.44 ([Reg83])
If $\mathsf{PH} = \Sigma_k^p$ *for some* k*, then* $H = \mathsf{PH}$.

Proof $\mathsf{PH} \subseteq H$ follows from Proposition 4.43 (1). Assume $\mathsf{PH} = \Sigma_k^p$. For $B \in C_m^p(\Sigma_k^p)$, $\mathsf{P}^B = \mathsf{NP}^B = \Sigma_k^p$, i.e. $B \in \mathcal{X}$. Take an arbitrary $L \in H$. From the definition of H, $(\forall A \in \mathcal{X})[L \in \mathsf{P}^A]$. Therefore, $L \in \mathsf{P}^B$, i.e. $B \in \mathsf{PH}$. Since L is arbitrary, $H \subseteq \mathsf{PH}$. □

Corollary 4.45 $H \neq \mathsf{PH} \Rightarrow \mathsf{PH} \neq \mathsf{PSPACE}$ *and* PH *does not collapse.*

Proof If PH collapses, then $H = \mathsf{PH}$ by Proposition 4.44. If $\mathsf{PH} = \mathsf{PSPACE}$, then PH collapses. □

Proposition 4.46 $H = \mathsf{PH} \neq \mathsf{PSPACE}$ *implies the existence of an oracle* $A \in \mathcal{X}$ *with* $\mathsf{P}^A = \mathsf{NP}^A \subsetneq \mathsf{PSPACE}$.

Proof Since $H = PH$, there exists an oracle $A \in \mathcal{X}$ such that $\mathsf{PH} = \mathsf{P}^A$. □

Conclusion of the previous proposition, i.e. the existence of an oracle $A \in \mathcal{X}$ with $\mathsf{P}^A = \mathsf{NP}^A \subsetneq \mathsf{PSPACE}$, is an open problem. Note that a similar problem, the existence of an oracle A such that $\mathsf{P}^A = \mathsf{NP}^A \neq \mathsf{PSPACE}^A$, was resolved by Ko [KO89] by using lower bounds on the size of constant depth circuits.

Balcázar also claimed the opposite implication of Proposition 4.43 (2). This was probably typo. Consider the "proof" of that implication, i.e. we want to prove $(\forall A \in \mathsf{PSPACE})[A \in \mathcal{X} \Rightarrow A \in C_T^p(\mathsf{PSPACE})] \Rightarrow H = \mathsf{PSPACE}$.

Suppose, the assumption of the implication holds. Take an arbitrary $L \in \mathsf{PSPACE}$, we want to show that $L \in \mathsf{P}^A$ for every $A \in \mathcal{X}$. Take an arbitrary $A \in \mathcal{X}$. For $A \in \mathsf{PSPACE}$, the task is simple: From the assumption, $A \in C_T^P(\mathsf{PSPACE})$. Therefore, $L \in \mathsf{P}^A$. Now consider $A \notin \mathsf{PSPACE}$. We do not know whether $\mathsf{PSPACE} \subseteq \mathsf{P}^A$. There is not known any set in $\mathcal{X}$ from PSPACE which is not hard for PSPACE. There is not even known any set in $\mathcal{X}$ which is not hard for PSPACE.

Note once more that the existence of a set $A \in \mathsf{PH}$ such that $A \in \mathcal{X}$ would collapse the polynomial time hierarchy PH (Proposition 4.8). From Corollary 4.14 (1), complete sets for PSPACE belong to $\mathcal{X}$. Whether there exists an oracle $A \in \mathcal{X}$, $A \in \mathsf{PSPACE}$ which is not complete for PSPACE is an open problem. What is between PH and PSPACE? Toda answered this question: PP is polynomial time Turing hard for PH, i.e. $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{PP}}$, where PP is the class of languages accepted by probabilistic polynomial time Turing machines (see [HO02]) and clearly $\mathsf{PP} \subseteq \mathsf{PSPACE}$. Nowadays, there is not known whether $\mathsf{P}^{\mathsf{PP}} = \mathsf{NP}^{\mathsf{PP}}$ or not. There is also not known any oracle A such that $\mathsf{P}^{\mathsf{PP}^A} \neq \mathsf{NP}^{\mathsf{PP}^A}$, which would mean, in some sense, a difficulty in proving $\mathsf{P}^{\mathsf{PP}} = \mathsf{NP}^{\mathsf{PP}}$, i.e $\mathsf{PP} \in \mathcal{X}$.

Chapter 5

Separating Oracles

In this chapter, we investigate properties of the oracles relative to which $\mathsf{P} \neq \mathsf{NP}$.

Definition 5.1 (Class of separating oracles)

$$\mathcal{Z} = \{A \mid \mathsf{P}^A \neq \mathsf{NP}^A\}.$$

Note that $\mathcal{Z}$ is the complement of the class $\mathcal{X}$, which is studied in the previous chapter. Therefore, many properties of $\mathcal{X}$ pass to $\mathcal{Z}$. We show that $\mathcal{Z}$ is not closed under intersections, unions and symmetric differences. The proof technique is similar to the one used for proving that $\mathcal{X}$ is not closed under these operations. Unlike the case of $\mathcal{X}$, where the connection with disjoint union is open, we prove that $\mathcal{Z}$ is not closed under disjoint unions. We conclude that disjoint union can lower complexity measured in terms of extended lowness.

5.1 Basic properties of $\mathcal{Z}$

Proposition 5.2

1. $\mathsf{P} \neq \mathsf{NP} \Leftrightarrow \emptyset \in \mathcal{Z} \Leftrightarrow (\exists A \in \mathsf{P})[A \in \mathcal{Z}] \Leftrightarrow (\forall A \in \mathsf{P})[A \in \mathcal{Z}]$.
2. *$\mathcal{Z}$ is neither the empty class nor the class of all sets.*
3. $\mu(\mathcal{Z}) = 1$.
4. *$\mathcal{Z}$ is closed under complements and under $\equiv_T^p$.*
5. *$\mathcal{Z}$ is closed under $\leq_m^p$ neither upward nor downward.*
6. *$\mathcal{Z}$ is closed under unions, intersection or symmetric difference* $\Rightarrow$ $\mathsf{P} \neq \mathsf{NP}$.

Proof The first four statements are just a rephrasing of Theorem 4.2 and Propositions 4.3, 4.4, 4.5 and 4.6. The statement (5) is rephrasing of Theorem 4.11 and 4.27. The last statement is proved in the same way as Proposition 4.7, by means of the foregoing propositions. □

Concerning the question of density, the first known oracle which belongs to $\mathcal{Z}$ is B from Theorem 3.6. This oracle B is obviously sparse. Even more, for every length k there is at most one word of length k in B. It seems to be hard to strengthen this to tally sets because tally sets yield a positive relativization of the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem (see Corollary 3.13).

5.2 $\mathcal{Z}$ is not closed under $\cap$, $\cup$ and $\triangle$

In this section, we prove that $\mathcal{Z}$ is not closed under any of these operations: union, intersection and symmetric differences. The proof is similar to the proof that $\mathcal{X}$ is not closed under the same operations.

Theorem 5.3 *$\mathcal{Z}$ is not closed under intersections, unions and symmetric differences.*

Proof We construct sets $X_i, Y_i \in \mathcal{Z}$, $1 \leq i \leq 3$, such that $X_1 \cap Y_1 \notin \mathcal{Z}$, $X_2 \cup Y_1 \notin \mathcal{Z}$ and $X_3 \triangle Y_3 \notin \mathcal{Z}$.

Recall the coding function h from Definition 4.34. Take an arbitrary $K \in C_m^p(\mathsf{DEXT})$. From Corollary 4.10 (1), $K \in \mathcal{X}$, i.e. $K \notin \mathcal{Z}$.

Applying Lemma 4.24 on the set K, we obtain a set B such that $K \oplus B \in \mathcal{Z}$. The way of how we construct the sets X_i, Y_i, $1 \leq i \leq 3$, is represented by Figures 5.1 - 5.3.

$\mathbf{X_1}$		$\mathbf{Y_1}$		$\mathbf{X_1 \cap Y_1}$
K	$\cap$	K	$=$	K
B		$\emptyset$		$\emptyset$
$\emptyset$		B		$\emptyset$

Figure 5.1 $X_1, Y_1 \in \mathcal{Z}$ but $X_1 \cap Y_1 \notin \mathcal{Z}$.

$\mathbf{X_2}$		$\mathbf{Y_2}$		$\mathbf{X_2 \cup Y_2}$
K	$\cup$	K	$=$	K
B		Σ^*		Σ^*
Σ^*		B		Σ^*

Figure 5.2 $X_2, Y_2 \in \mathcal{Z}$ but $X_2 \cup Y_2 \notin \mathcal{Z}$.

$\mathbf{X_3}$		$\mathbf{Y_3}$		$\mathbf{X_3 \triangle Y_3}$
K	$\triangle$	$\emptyset$	$=$	K
$\emptyset$		K		K
B		B		$\emptyset$

Figure 5.3 $X_3, Y_3 \in \mathcal{Z}$ but $X_3 \triangle Y_3 \notin \mathcal{Z}$.

Define the sets X_i, $1 \leq i \leq 3$, as follows.

$|\beta| = 3k + 0$:
$\beta \in X_i \Leftrightarrow (\exists \alpha \in K)[h(\alpha) = \beta]$ $(1 \leq i \leq 3)$
$|\beta| = 3k + 1$:
$\beta \in X_i \Leftrightarrow (\exists \alpha \in B)[h(\alpha) = \beta']$ $(1 \leq i \leq 2)$
$(\forall \beta)[\beta \notin X_3]$
$|\beta| = 3k + 2$:
$(\forall \beta)[\beta \notin X_1]$
$(\forall \beta)[\beta \in X_2]$
$\beta \in X_3 \Leftrightarrow (\exists \alpha \in B)[h(\alpha) = \beta'']$

Definition of X_i, $1 \leq i \leq 3$, by the length of β.

Define the sets Y_i, $i \leq i \leq 3$, as follows.

$|\beta| = 3k + 0$:
$\beta \in Y_i \Leftrightarrow (\exists \alpha \in K)[h(\alpha) = \beta]$ $(1 \leq i \leq 2)$
$(\forall \beta)[\beta \notin Y_3]$
$|\beta| = 3k + 1$:
$(\forall \beta)[\beta \notin Y_1]$
$(\forall \beta)[\beta \in Y_2]$
$\beta \in Y_3 \Leftrightarrow (\exists \alpha \in K)[h(\alpha) = \beta']$
$|\beta| = 3k + 2$:
$\beta \in Y_i \Leftrightarrow (\exists \alpha \in B)[h(\alpha) = \beta'']$ $(1 \leq i \leq 3)$

Definition of Y_i, $1 \leq i \leq 3$, by the length of β.

Denote $Z_1 = X_1 \cap Y_1$, $Z_2 = X_2 \cup Y_2$ and $Z_3 = X_3 \triangle Y_3$.

Since $K \in C_m^p(\mathsf{DEXT})$, $K \neq \emptyset$ and $coK \neq \emptyset$. Since the complete sets for DEXT under $\leq_m^p$-reductions are closed under complements, $K = \emptyset$ or $coK = \emptyset$ would imply $\mathsf{P} = \mathsf{DEXT}$, which is in contradiction with the Time Hierarchy Theorem 2.10. Hence, we can take fixed words $u \in K$ and $v \notin K$.

Claim 1 *Z_1, Z_2, $Z_3 \in C_m^p(\mathsf{DEXT})$.*

Proof Recall that $K \in C_m^p(\mathsf{DEXT})$. From the definitions of Z_i, $K \leq_m^p Z_i$ via function $f(x) = h(x)$, $1 \leq i \leq 3$. Therefore, $Z_i \in H_m^p(\mathsf{DEXT})$.

From the definition of Z_1, $Z_1 \leq_m^p K$ via function

$$f_1(x) = \begin{cases} h^{-1}(x), & \text{if } |x| = 3k \text{ and } x \text{ is triplicate} \\ v, & \text{otherwise} \end{cases}$$

From the definition of Z_2, $Z_2 \leq_m^p K$ via function

$$f_2(x) = \begin{cases} h^{-1}(x), & \text{if } |x| = 3k \text{ and } x \text{ is triplicate} \\ v, & \text{if } |x| = 3k \text{ and } x \text{ is not triplicate} \\ u, & \text{otherwise} \end{cases}$$

From the definition of Z_3, $Z_3 \leq_m^p K$ via function

$$f_3(x) = \begin{cases} h^{-1}(x), & \text{if } |x| = 3k \text{ and } x \text{ is triplicate} \\ h^{-1}(y), & \text{if } |x| = 3k+1 \text{ and } x \text{ is a form of } yb, \\ & \text{y triplicate and } b \in \{0,1\} \\ v, & \text{otherwise} \end{cases}$$

Since DEXT is closed under $\leq_m^p$-reductions downward, i.e. $L_1 \leq_m^p L_2$ and $L_2 \in \mathsf{DEXT}$ imply $L_1 \in \mathsf{DEXT}$, we obtain $Z_i \in \mathsf{DEXT}$, $1 \leq i \leq 3$. Consequently, $Z_i \in C_m^p(\mathsf{DEXT})$, $1 \leq i \leq 3$. □

Corollary 1 *Z_1, Z_2, $Z_3 \notin \mathcal{Z}$.*

Proof From Claim 1 and Corollary 4.10 (1), $Z_i \in \mathcal{X}$, i.e. $Z_i \notin \mathcal{Z}$, $1 \leq i \leq 3$. □

Claim 2 *$X_1 \equiv_m^p K \oplus B \equiv_m^p Y_1$.*

Proof From the definition of X_1, $K \oplus B \leq_m^p X_1$ via function

$$g_1(x) = \begin{cases} h(y), & \text{if } x = 0y \\ h(y)0, & \text{if } x = 1y \end{cases}$$

Note that in case of $x = 1y$, g_1 can be defined equivalently as $g_1(x) = h(y)1$.

From the definition of X_1, $X_1 \leq_m^p K \oplus B$ via function

$$g_2(x) = \begin{cases} 0h^{-1}(x), & \text{if } |x| = 3k \text{ and } x \text{ is triplicate} \\ 1h^{-1}(y), & \text{if } |x| = 3k+1 \text{ and } x \text{ is a form of } yb, \\ & \text{y triplicate and } b \in \{0,1\} \\ 0v, & \text{otherwise} \end{cases}$$

From the definition of Y_1, $K \oplus B \leq_m^p Y_1$ via function

$$g_3(x) = \begin{cases} h(y), & \text{if } x = 0y \\ h(y)00, & \text{if } x = 1y \end{cases}$$

Note that in case of $x = 1y$, g_3 can be defined equivalently as $g_3(x) = h(y)01$, or $g_3(x) = h(y)10$ or $g_3(x) = h(y)11$.

From the definition of Y_1, $Y_1 \leq_m^p K \oplus B$ via function

$$g_4(x) = \begin{cases} 0h^{-1}(x), & \text{if } |x| = 3k \text{ and } x \text{ is triplicate} \\ 1h^{-1}(y), & \text{if } |x| = 3k+2 \text{ and } x \text{ is a form of } yb_1b_2, \\ & \text{y triplicate and } b_1, b_2 \in \{0,1\} \\ 0v, & \text{otherwise} \end{cases}$$

□

Corollary 2 *X_1, $Y_1 \in \mathcal{Z}$.*

Proof From Claim 2 and Proposition 5.2 (4). □

Claim 3 *$X_2 \equiv_m^p K \oplus B \equiv_m^p Y_2$.*

Proof From the definition of X_2, $K \oplus B \leq_m^p X_2$ via the foregoing function g_1.

From the definition of X_2, $X_2 \leq_m^p K \oplus B$ via function

$$g_5(x) = \begin{cases} 0h^{-1}(x), & \text{if } |x| = 3k \text{ and } x \text{ is triplicate} \\ 1h^{-1}(y), & \text{if } |x| = 3k+1 \text{ and } x \text{ is a form of } yb, \\ & \text{y triplicate and } b \in \{0,1\} \\ 0u, & \text{if } |x| = 3k+2 \\ 0v, & \text{otherwise} \end{cases}$$

From the definition of Y_2, $K \oplus B \leq_m^p Y_2$ via the foregoing function g_3.

From the definition of Y_2, $Y_2 \leq_m^p K \oplus B$ via function

$$g_6(x) = \begin{cases} 0h^{-1}(x), & \text{if } |x| = 3k \text{ and } x \text{ is triplicate} \\ 0u, & \text{if } |x| = 3k+1 \\ 1h^{-1}(y), & \text{if } |x| = 3k+2 \text{ and } x \text{ is a form of } yb_1b_2, \\ & \text{y triplicate and } b_1, b_2 \in \{0,1\} \\ 0v, & \text{otherwise} \end{cases}$$

□

Corollary 3 *X_2, $Y_2 \in \mathcal{Z}$.*

Proof From Claim 3 and Proposition 5.2 (4). □

Claim 4 *$X_3 \equiv_m^p K \oplus B \equiv_m^p Y_3$.*

Proof

From the definitions of Y_1 and X_3, $Y_1 = X_3$. Therefore, $X_3 \equiv_m^p K \oplus B$ from Claim 2.

From the definition of Y_3, $K \oplus B \leq_m^p Y_3$ via function

$$g_7(x) = \begin{cases} h(y)0, & \text{if } x = 0y \\ h(y)00, & \text{if } x = 1y \end{cases}$$

Note that in case of $x = 0y$, g_7 can be defined equivalently as $g_7(x) = h(y)1$. Note that in case of $x = 1y$, g_7 can be defined equivalently as $g_7(x) = h(y)01$, or $g_7(x) = h(y)10$ or $g_7(x) = h(y)11$.

From the definition of Y_3, $Y_3 \leq_m^p K \oplus B$ via function

$$g_8(x) = \begin{cases} 0h^{-1}(y), & \text{if } |x| = 3k+1 \text{ and } x \text{ is a form of } yb \\ & \text{y triplicate and } b \in \{0,1\} \\ 1h^{-1}(y), & \text{if } |x| = 3k+2 \text{ and } x \text{ is a form of } yb_1b_2, \\ & \text{y triplicate and } b_1, b_2 \in \{0,1\} \\ 0v, & \text{otherwise} \end{cases}$$

□

Corollary 4 *X_3, $Y_3 \in \mathcal{Z}$.*

Proof From Claim 4 and Proposition 5.2 (4). □

(Theorem 5.3) □

5.3 $\mathcal{Z}$ is not closed under $\oplus$

Theorem 5.4 *$\mathcal{Z}$ is not closed under disjoint unions.*

Proof We construct A and B such that A, $B \in \mathcal{Z}$, but $A \oplus B \notin \mathcal{Z}$. Before giving the formal proof, we show the idea of the proof. To ensure that A, $B \in \mathcal{Z}$, i.e. $\mathsf{P}^A \neq \mathsf{NP}^A$ and $\mathsf{P}^B \neq \mathsf{NP}^B$, we diagonalize against deterministic polynomial time Turing machines with oracles in the same way we do in the proof of Theorem 3.6. Besides the diagonalization, we also encode a complete problem K for PSPACE into A and B which ensures that both A and B are hard for PSPACE. Of course, neither A nor B can belong to PSPACE (in that case, they would be complete for PSPACE and therefore in $\mathcal{X}$ by Corollary 4.14 (1)). We encode some information about A into B and vice versa, so that $A \oplus B$ is powerful enough to recognize strings which we use in A and B for diagonalization, and so $A \oplus B \in \mathcal{X}$.

stage 0:
$k(0) = 0$
$l(0) = 0$
$A_0 = \{xx \mid x \in K\}$
$B_0 = \{xx \mid x \in K\}$
stage n:
Let $k(n)$ be the smallest odd natural number such that
$k(n) > \max\{p_{n-1}(k(n-1)), p_{n-1}(l(n-1))\}$ and $p_n(k(n)) < 2^{k(n)}$.
If $0^{k(n)} \in L(M_n, A_{n-1})$ then $A'_n = A_{n-1}$ and $B'_n = B_{n-1}$.
If $0^{k(n)} \notin L(M_n, A_{n-1})$ then $A'_n = A_{n-1} \cup \{y_1(n)\}$ and $B'_n = B_{n-1} \cup S_1$,
where $y_1(n)$ is the first word, in lexicographic order, of length $k(n)$ such that $y_1(n)$ is not queried in the computation of M_i on input $0^{k(n)}$ with the oracle A_{n-1} and $S_1 = \{x \mid |x| = k(n) \wedge x \leq_{lex} y_1(n)\}$.
Let $l(n)$ be the smallest odd natural number such that
$l(n) > \max\{p_{n-1}(l(n-1)), p_n(k(n))\}$ and $p_n(l(n)) < 2^{l(n)}$.
If $0^{l(n)} \in L(M_n, B'_n)$ then $B_n = B'_n$ and $A_n = A'_n$.
If $0^{l(n)} \notin L(M_n, B'_n)$ then $B_n = B'_n \cup \{y_2(n)\}$ and $A_n = A'_n \cup S_2$,
where $y_2(n)$ is the first word, in lexicographic order, of length $l(n)$ such that $y_2(n)$ is not queried in the computation of M_i on input $0^{l(n)}$ with the oracle B'_n and $S_2 = \{x \mid |x| = l(n) \wedge x \leq_{lex} y_2(n)\}$.

Figure 5.4 Construction of A, B such that $L(A) \notin \mathsf{P}^A$ and $L(B) \notin \mathsf{P}^B$.

Recall the enumeration $\{M_i\}_{i\geq 1}$ from Fact 2.27. We construct A and B in stages. Denote $k(n)$ and $l(n)$ increasing sequences of natural numbers: $k(n)$ is the length of word which is used in the n-th stage to ensure that M_n does not accept $L(A)$ and $l(n)$ is defined similarly for M_n and $L(B)$. After n stages, we denote the the so far constructed oracles A_{n-1} and B_{n-1}. In the n-th stage, we add at most one word of length $k(n)$ to A and at most one word of length $l(n)$ to B to ensure that $L(A) \neq L(M_n, A)$ and $L(B) \neq L(M_n, B)$. If a word w is added to A, then some words are added to B in order to ensure $A \oplus B \in \mathcal{X}$. Similarly, for B and A the other way around. We show that either $0^{k(n)} \in L(M_n, A)$ and $A \cap \Sigma^{=k(n)} = \emptyset$, or $0^{k(n)} \notin L(M_n, A)$ and $A \cap \Sigma^{=k(n)} \neq \emptyset$. Similarly, we show that either $0^{l(n)} \in L(M_n, B)$ and $B \cap \Sigma^{=l(n)} = \emptyset$, or $0^{l(n)} \notin L(M_n, B)$ and $B \cap \Sigma^{=l(n)} \neq \emptyset$.

Figure 5.4 describes the n-th stage in the construction of A and B.

The final oracles are $A = \cup_n A_n$ and $B = \cup_n B_n$.

From the conditions on $k(n)$ and $l(n)$, appropriate words $y_1(n)$ of length $k(n)$ and $y_2(n)$ of length $l(n)$ always exist if needed. There are $2^{k(n)}$ words of length $k(n)$ but $p_n(k(n)) < 2^{k(n)}$ and there are $2^{l(n)}$ words of length $l(n)$

but $p_n(l(n)) < 2^{l(n)}$.

Conditions on $k(n)$ and $l(n)$ also ensure that $k(n)$ and $l(n)$ are long enough not to disturb, by possible adding of some words to A and B, any computation from the previous stages. Condition on $l(n)$, $l(n) > p_n(k(n))$ ensures that the second part of the n-th stage does not disturb, by possible adding of some words to A and B, the first part of the n-th stage. It means that $0^{k(n)} \in L(M_n, A_{n-1}) \Leftrightarrow 0^{k(n)} \in L(M_n, A)$. Similarly, $0^{l(n)} \in L(M_n, B_{n-1}) \Leftrightarrow 0^{l(n)} \in L(M_n, B)$. This holds for every n. Therefore $0^{k(n)} \in L(A) \Leftrightarrow 0^{k(n)} \notin L(M_n, A)$ for every n. Similarly $0^{l(n)} \in L(B) \Leftrightarrow 0^{l(n)} \notin L(M_n, B)$ for every n. Therefore, $L(A) \notin \mathsf{P}^A$ and $L(B) \notin \mathsf{P}^B$. Since $L(A) \in \mathsf{NP}^A$ and $L(B) \in \mathsf{NP}^A$, we obtain $\mathsf{P}^A \neq \mathsf{NP}^A$ and $\mathsf{P}^B \neq \mathsf{NP}^B$, i.e. $A \in \mathcal{Z}$ and $B \in \mathcal{Z}$.

Note the structure of $A \oplus B$. From the construction of A and B follows: If w is a word of even length from $A \oplus B$, then either w is the only word of length n in $0A$ (we call this word the *unique word* of length n in $A \oplus B$) and $1B$ consist of all words of length n lexicographically smaller than w, or the other way around (interchange A and B).

Claim 1 *There is a deterministic polynomial time algorithm (with the oracle $A \oplus B$) that for a given odd n finds out whether there are no words of length n in $A \oplus B$, or it finds the unique word w of length n and also it finds out whether $w \in A$ or $w \in B$. The algorithm needs only $O(n)$ queries to the oracle.*

Proof Binary search. □

Claim 2 $A \oplus B \notin \mathcal{Z}$.

Proof Take an arbitrary $L \in \mathsf{NP}^{A \oplus B}$, then there exists a nondeterministic polynomial time Turing machine M_1 with an oracle such that $L = L(M_1, A \oplus B)$. Let polynomial $p(n)$ bound the running time of M_1.

We define Turing machine M_2 which has on its input tape, besides the input word x of length n, the description of words of even length in $A \oplus B$. Note that this description is polynomial in n: For every even number k between 2 and $p(n)$ (or $p(n) - 1$, if $p(n)$ is odd), it consists of the unique word w of length k (if it exists) and a single bit indicating whether $w \in A$ or $w \in B$. Define Turing machine M_2 such that on input x (and its extended input):

1. Simulate M_1 on input x until M_1 asks an oracle a word w or accepts or rejects.

2. Accept whenever M_1 accepts, reject whenever M_1 rejects and stops.
3. If M_1 asks w of odd length, then the answer is yes if and only if the answer for M_2 is yes and we continue on the simulation in Step 1.
4. If M_1 asks w of even length, then use the extended input for the answer and we continue on the simulation in Step 1.

M_2 is nondeterministic because M_1 is nondeterministic and uses polynomial time because M_1 works in polynomial time. Note that M_2 does not query words of even length. If the oracle of M_2 is $A \oplus B$ and its extended input agrees with the structure of words of even length in $A \oplus B$, then M_2 accepts L, i.e. $L = L(M_2, A \oplus B)$.

Because M_2 does not query words of even length, the only information that M_2 can obtain from the oracle $A \oplus B$ on odd words are words of even length from A and B, which encode $K \in C_m^p(\mathsf{PSPACE})$. From Corollary 4.14 (1), there exists a deterministic polynomial time Turing machine M_3 with an oracle such that $L = L(M_3, A \oplus B)$ with an extended input. Let polynomial $q(n)$ bounds the running time of M_3.

We define a Turing machine M_4 such that on input x of length n:

1. By binary search with help of the oracle, compute in polynomial time the description of words of even length in the oracle set.
2. Simulate M_3 on input x until M_3 asks a word w or accepts or rejects.
3. Accept whenever M_3 accepts, reject whenever M_3 rejects and stops.
4. If M_3 asks w of odd length, then the answer is yes if and only if the answer for M_4 is yes and we continue on the simulation in Step 2.
5. If M_3 asks w of even length to its extended input, use the extended input of the machine M_4 and we continue on the simulation in Step 2.

M_4 is deterministic because M_3 is deterministic and uses polynomial time because M_3 works in polynomial time and Step 1 takes polynomial time: For polynomial many words apply binary search (Claim 1). If the oracle of M_4 is $A \oplus B$, then M_4 accepts L, i.e. $L = L(M_4, A \oplus B)$. Therefore, $L \in \mathsf{P}^{A \oplus B}$ and since L is arbitrary, $\mathsf{NP}^{A \oplus B} \subseteq \mathsf{P}^{A \oplus B}$, i.e. $A \oplus B \notin \mathcal{Z}$. □

(Theorem 5.4) □

We can prove that both $\mathcal{X}$ and its complement $\mathcal{Z}$ are not closed under intersections, unions and symmetric differences. The reason is that under these operations, we can construct the oracle sets in the way that some of the information disappear and some other information reveals under these operations. In case of $\mathcal{Z}$, we also manage to show that $\mathcal{Z}$ is not closed under

disjoint unions. In case of A and B from Theorem 5.4, the disjoint union $A \oplus B$ reveals some information so that we could show collapse of NP to P relativized to $A \oplus B$. The same technique seems to be insufficient for proving that $\mathcal{X}$ is not closed under disjoint unions. There, we need to lose information, but disjoint union does not forget.

Corollary 5.5 *There exist sets* $A \notin \mathsf{EL}_1$ *and* $B \notin \mathsf{EL}_1$ *such that* $A \oplus B \in \mathsf{EL}_1$.

Proof From Theorem 4.33, $A \in \mathcal{X} \Leftrightarrow A \in \mathsf{EL}_1 \cap \mathsf{EH}_0$. In the proof of Theorem 5.4, we construct A, $B \in \mathcal{Z}$ such that $A \oplus B \notin \mathcal{Z}$, i.e. A, $B \notin \mathcal{X}$ and $A \oplus B \in \mathcal{X}$, i.e. $(A \notin \mathsf{EL}_1 \vee A \notin \mathsf{EH}_0)$ and $(B \notin \mathsf{EL}_1 \vee B \notin \mathsf{EH}_0)$. Since $K \leq_T^p A$ and $K \leq_T^p B$, where K is a complete problem for PSPACE, it follows that $SAT \leq_T^p A$ and $SAT \leq_T^p B$ from Proposition 4.32. Consequently, $A \notin \mathsf{EL}_1$ and $B \notin \mathsf{EL}_1$. □

Let us comment the meaning of Corollary 5.5. In the classical complexity theory, based on reductions, the join operator can not lower the complexity. We just present a result that join operator can lower complexity measured in terms of extended-lowness. This, with a similar result for EL_2 [HJRW99], shows that the extended low hierarchy is not a natural measure of complexity.

Chapter 6

Conclusion

We conclude this thesis with summarizing the main results and we also outline further possible research.

This thesis investigates characteristics and properties of the oracles relative to the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem. Recall that $\mathcal{X}$ is the class of sets relative to which $\mathsf{P} = \mathsf{NP}$ relativized and $\mathcal{Z}$ the class of sets relative to which $\mathsf{P} \neq \mathsf{NP}$.

We show that complete problems for various complexity classes belong to $\mathcal{X}$ and also the consequences of having complete sets in $\mathcal{X}$ for some other complexity classes. Naturally, this can be investigated for other complexity classes. The most interesting case is whether complete problems for PP, which lies between PH and PSPACE, are in $\mathcal{X}$ or not. In Section 4.3.6, we show that there are complete problems for DH under any reductions stronger than polynomial, if they ever exist, that do not belong to $\mathcal{X}$. We show an example of such reduction and we obtain a complete problem for DH which is out of $\mathcal{X}$. Another approach is to investigate different complexity classes (nondeterministic, space complexity, different time bound and so on) than DH which lie between polynomial and exponential world.

The foregoing problems are connected with the following two problems:
1. Is there a set $A \in \mathcal{X}$ from PSPACE such that A is not complete for PSPACE?
2. Is there a set $A \in \mathcal{X}$ such that A is not hard for PH?

We show two characterizations of the class $\mathcal{X}$. Further characterizations are desired to understand the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem better. Besides the characterization of the class $\mathcal{X}$ in terms of the extended hierarchies, we also show in Section 4.7 a connection between $\mathcal{X}$ and the extended hierarchies in the following sense: If PH does not collapse and Hypothesis T about $\mathcal{X}$ (from Section 4.7) holds, then $\mathsf{EL}_1 \cap \mathsf{EH}_0 = \mathsf{EL}_1 \cap \mathsf{EH}_1$. The question of proving or disproving the Hypothesis T remains open. Its special case, whether there exists a set A such that $A \oplus SAT \in \mathsf{EL}_1$ and A is not hard for NP, is also open.

We investigate closedness of $\mathcal{X}$ under reductions and boolean operations. We show that $\mathcal{X}$ is closed under $\leq_m^p$-reductions (and therefore under any stronger reductions) neither upward nor downward. We prove that $\mathcal{X}$ is not closed under unions, intersections and symmetric differences. The question about relationships between the class $\mathcal{X}$ and disjoint union is still open. Answering this question would explain the connection between EL_1 and disjoint union.

We show a close connection between the class $\mathcal{X}$ and $\mathcal{Z}$. We prove that $\mathcal{Z}$ is not closed under unions, intersections and symmetric differences.

We prove that $\mathcal{Z}$ is not closed under disjoint unions. This shows that disjoint union can lower complexity measured in terms of extended lowness. This result extends a similar result about EL_2 ([HJRW99]). Closedness of EL_1, which is closely connected to $\mathcal{X}$, under boolean operations is still open.

Different operations can also be investigated. The question of closedness of $\mathcal{X}$ under homomorphisms seems to be connected with our results.

All foregoing open questions and problems are left for further investigation.

Bibliography

[All90] E. Allander. Oracles versus proof techniques that do not relativize. In *Proceedings of the 1990 International Symposium on Algorithms*, pages 39-52. Springer-Verlag *Lecture Notes in Computer Science* ♯450, August 1990.

[Bal84] J. Balcázar. Separating, strongly separating, and collapsing relativized complexity classes. In *Proceedings of Mathematical Foundations of Computer Science 1984*, pages 1-16. Springer-Verlag *Lecture Notes in Computer Science* ♯176, 1984.

[BBS86] J. Balcázar, R. Book, and U. Schöning. The polynomial-time hierarchy and sparse oracles. *Journal of the Association for Computing Machinery*, 33(3):603-617, 1986.

[BCS95] D. Bovet, P. Crescenzi, and R. Silvestri. Complexity classes and sparse oracles. *Journal of Computer and System Sciences*, 50(3):382-390, 1995.

[BDG88] J. Balcázar, J. Diáz, and J. Gabarró. *Structural Complexity I*, volume 11 of EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1988.

[BDG90] J. Balcázar, J. Diáz, and J. Gabarró. *Structural Complexity II*, volume 22 of EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1990.

[BG81] C. Bennett and J. Gill. Relative to a random oracle A, $P^A \neq NP^A \neq coNP^A$ with probability 1. *SIAM Journal on Computing*, 10(1):96-113, 1981.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question. *SIAM Journal on Computing*, 4(4):431-442, 1975.

[BHL95] H. Buhrman, E. Hemaaspaandra, and L. Longpré. SPARSE reduces conjunctively to TALLY. *SIAM Journal on Computing*, 24(4):673-681, 1995.

[For94] L. Fortnow. The role of relativization in complexity theory. *Bulletin of the EATCS*, 52:229-244, 1994.

[Har83] J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations. In *Proceedings of the 24th Symposium on the Foundations of Computer Science*, IEEE, New York, 439-445, 1983.

[Har85] J. Hartmanis. Solvable problems with conflicting relativizations. *Bulletin of the EATCS*, 27:40-49, Oct 1985.

[Hem89] L. Hemachandra. The strong exponential hierarchy collapses. *Journal of Computer and System Sciences*, 39(3):299-322, 1989.

[HCCC$^+$] J. Hartmanis, R. Chang, S. Chari, D. Ranjan, and P. Rohatgi. Relativization: A revisionistic retrospective. *Bulletin of the EATCS*, 47:144-153, 1992.

[Hit04] John M. Hitchcock. Hausdorff Dimension and Oracle Constructions, *Electronic Colloquium on Computational Complexity*, Report No. 72 (2004).

[HJRW99] L. Hemaspaandra, Z. Jiang, J. Rothe, O. Watanabe. Boolean Operations, Joins, and the Extended Low Hierarchy. *Theoretical Computer Science*, 205:317-327, 1998.

[HO02] L. Hemaspaandra, M. Ogihara. *The Complexity Theory Companion*. EATCS Texts in Theoretical Computer Science. Springer-Verlag, 2002.

[HZR95] L. Hemaspaandra, A. Ramachandran, and M. Zimand. Worlds to die for. *SIGACT News*, 26(4):5-15, 1995.

[KO89] K. Ko. Relativized Polynomial Time Hierarchies Having Exactly K Levels. *SIAM Journal on Computing* 18(2):392-408, 1989.

[KSTT89] J. Köbler, Uwe Schöning, S. Toda, and J. Torán. Turing machines with few accepting computations and low sets for PP. In *Proceedings, 4th IEEE Structure in Complexity Theory Conf.*, pp. 205-215, 1989.

[Kur83] S. Kurtz. On the random oracle hypothesis. *Information and Control*, 57(1):40-47, April 1983.

[Lon85] T. Long. On restricting the size of oracles compared with restricting access to oracles. *SIAM Journal on Computing*, 14:585-597, 1985.

[LS86] T. Long and A. Selman. Relativizing complexity classes with sparse oracles. *Journal of the Association for Computing Machinery*, 33(3):618-627, 1986.

[Meh73] K. Mehlhorn. On size of sets of computable functions. *Proceedings of the 14th IEEE Symposium on Switching and automata Theory*, Iowa City, IO, 1973, pp.190-196.

[Reg83] K. Regan. Arithmetical degrees of index sets for complexity classes. In *Logic and Machines: Decision Problems and Complexity, Proceedings of the Symposium "Rekursive Kombinatorik" 1983*, pages 118-130. Springer-Verlag *Lecture Notes in Computer Science* ♯171, 1984.

[Ver94] N. Vereshchagin. Relativizable and nonrelativizable theorems in the polynomial theory of algorithms. *Russian Academy of Sciences-Izvestiya-Mathematics*, 42(2):261-298, 1994.