Operators of Higher Order

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Bayerischen Julius – Maximilians – Universität Würzburg

vorgelegt von

Herbert Alexander Baier Saip

aus

Concepción (Chile)

Würzburg, 1998

Eingereicht am: 12.02.98
bei der Fakultät für Mathematik und Informatik

1. Gutachter: *Prof. Dr. Klaus W. Wagner*
2. Gutachter: *Prof. Dr. Gerd Wechsung*

Tag der mündlichen Püfung: 02.07.98

I dedicate this thesis to my grandfather, Pablo Saip, an outstanding and inspiring person, my good friend. He will always be with me.

Acknowledgments

I would like to express my sincere acknowledgments to my friends and colleagues, who in various ways helped me throughout this work:

- My advisor, Klaus W. Wagner, for his valuable discussions, interesting ideas, encouragement and support throughout this work.
- Heribert Vollmer for his helpful discussions and suggestions.
- My colleagues, Sven Kosub, Steffen Reith and Heinz Schmitz, for reading this manuscript.
- Gerhard Buntrock, Ulrich Hertrampf, Gisela Hoppe, Sven Kosub, Gundula Niemann, Steffen Reith, Diana Rooß, Heinz Schmitz, Heribert Vollmer and Klaus W. Wagner for providing me an enjoyable working atmosphere.

This work was supported by **CNPq** grant 290020/92.1 (Brazil) and in part by **DAAD** (Germany).

Danksagung

Ich möchte gern allen meinen Freunden und Kollegen danken, die zum Gelingen dieſer Arbeit beigetragen haben:

- Meinem Doktorvater Klaus W. Wagner für intereſſante Diſkuſſionen und Ideen, Ermutigung und Unterſtützung während dieſer Arbeit.
- Heribert Vollmer für intereſſante Diſkuſſionen und Vorſchläge.
- Meinen Kollegen Sven Koſub, Steffen Reith und Heinz Schmitz für das Leſen des Manuſkripts.
- Gerhard Buntrock, Ulrich Hertrampf, Giſela Hoppe, Sven Koſub, Gundula Niemann, Steffen Reith, Diana Rooß, Heinz Schmitz, Heribert Vollmer und Klaus W. Wagner für die gute Arbeitsatmoſphäre.

Diese Arbeit entstand im Rahmen des **CNPq**-Stipendiums 290020/92.1 (Brasilien) und wurde teilweise durch den **DAAD** (Deutschland) unterstützt.

Abstract

Motivated by results on interactive proof systems we investigate the computational power of quantifiers applied to well-known complexity classes. In special, we are interested in existential, universal and probabilistic bounded error quantifiers ranging over words and sets of words, i.e. oracles if we think in a Turing machine model. In addition to the standard oracle access mechanism, we also consider quantifiers ranging over oracles to which access is restricted in a certain way.

We first examine an $\exists$-$\forall$-hierarchy over *P* using words quantifiers as well as two types of set quantifiers. This hierarchy of classes is called the analytic polynomial-time hierarchy. We show that each class of this hierarchy coincides with one of the classes Σ_k^p and Π_k^p $(k \geq 0)$ of the (arithmetic) polynomial-time hierarchy, *PSPACE*, or one of the classes Σ_k^{exp} and Π_k^{exp} $(k \geq 1)$ of the exponential-time alternation hierarchy and vice versa.

We next consider a hierarchy which refines the analytic polynomial-time hierarchy by considering restrictions on the number of oracle queries, the so called bounded analytic polynomial-time hierarchy. We characterize classes of this hierarchy by well-known complexity classes. In particular, we show that each class from this hierarchy having a certain normal form coincides with one of the classes *NP*, co*NP*, *PSPACE*, Σ_k^{exp} or Π_k^{exp} $(k \geq 1)$ and vice versa. All these characterizations remain valid if the queries are asked in a nonadaptive form, i.e. in “parallel”.

We also study a hierarchy which can intuitively be interpreted as the analytic polynomial-time hierarchy defined over *L* instead of *P*, i.e. an $\exists$-$\forall$-hierarchy over *L* using word quantifiers as well as two types of set quantifiers. This hierarchy is called the analytic logarithmic-space hierarchy. We show that every class of this hierarchy can be represented in a certain normal form and characterize such classes by well-known complexity classes. In particular, each class whose last quantifier is a word quantifier coincides with one of the classes *L*, Σ_k^p or Π_k^p $(k \geq 1)$ and vice versa.

Furthermore, we examine probabilistic bounded error quantifiers. For instance, using the restricted oracle access mechanism we characterize (one prover) interactive proof systems by an existential set quantifier and a probabilistic bounded error word quantifier applied to *P*, and show that a bounded error set quantifier applied to *PSPACE* can be eliminated without changing the class in question.

Finally, we discuss the relativizability of the results.

Zusammenfassung

Angeregt durch die Resultate über interaktive Beweissysteme untersuchen wir Quantoren in Anwendung auf bereits bekannte Komplexitätsklassen hinsichtlich ihrer dadurch gegebenen Berechnungsmächtigkeit. Von besonderem Interesse sind dabei existentielle und universelle Quantoren sowie Quantoren mit begrenzter Fehlerwahrscheinlichkeit, die alle über Wörter oder Wortmengen (Orakel im Kontext der Turingmaschinen) quantifizieren. Außer in bezug auf den Standardmechanismus eines Orakelzugriffs werden auch Quantifizierungen über Orakel, für deren Zugriff gewisse Beschränkungen bestehen, betrachtet.

Zuerst beschäftigen wir uns mit einer $\exists$-$\forall$-Hierarchie über *P*, wobei sowohl Wortquantoren als auch zwei verschiedene Typen von Mengenquantoren verwendet werden. Die so entstehende Klassenhierarchie nennen wir die *analytische Polynomialzeit-Hierarchie*. Es zeigt sich, daß jede Klasse dieser Hierarchie mit einer der Klassen Σ_k^p oder Π_k^p ($k \geq 1$) der (arithmetischen) Polynomialzeit-Hierarchie, mit *PSPACE* oder mit einer der Klassen Σ_k^{exp} oder Π_k^{exp} ($k \geq 1$) der alternierenden Exponentialzeit-Hierarchie zusammenfällt. Auch die Umkehrung gilt; jede der aufgeführten Klassen läßt sich durch eine der Klassen aus der analytischen Polynomialzeit-Hierarchie ausdrücken.

Als nächstes wird eine Hierarchie betrachtet, die die analytische Polynomialzeit-Hierarchie durch die Einbeziehung von Anzahlbegrenzungen der Orakelfragen verfeinert: die sogenannte *beschränkte analytische Polynomialzeit-Hierarchie*. Wir charakterisieren die Klassen dieser Hierarchie durch bekanntere Komplexitätsklassen, und zeigen insbesonders, daß jede Klasse der Hierarchie, die einer bestimmten Normalform genügt, einer der Klassen *NP*, co*NP*, *PSPACE*, Σ_k^{exp} oder Π_k^{exp} ($k \geq 1$) entspricht. Auch hier ist die Umkehrung der Aussage ebenfalls richtig. Darüber hinaus bleiben alle Charaktierisierungen gültig, wenn Orakelfragen ausschließlich nicht-adaptiv, also in gewissem Sinne parallel gestellt werden können.

Wir studieren auch eine Hierarchie, die intuitiv als analytische Polynomialzeit-Hierarchie über *L* anstelle von *P* interpretiert werden kann, d.h. die $\exists$-$\forall$-Hierarchie über *L* sowohl bezüglich der Wortquantoren als auch bezüglich der zwei Typen von Mengenquantoren. Diese Hierarchie wird die *analytische Hierarchie über logarithmischem Raum* genannt. Wir zeigen, daß jede Klasse dieser Hierarchie in eine bestimmte Normalform gebracht werden kann, und charakterisieren solche Klassen dann mit Hilfe bereits bekannter Komplexitätsklassen. Dabei stellt sich heraus, daß jede Klasse, deren letzter Quantor ein Wortquantor ist, mit *L*, Σ_k^p oder Π_k^p ($k \geq 1$) identisch ist, und umgekehrt.

Weiterhin untersuchen wir Quantoren mit begrenzter Fehlerwahrscheinlichkeit. Beispielsweise ist die Klasse der mittels interaktiven Beweissystemen in Polynomialzeit entscheidbaren Mengen, die Klasse *IP*, im Kontext polynomieller Zeitressourcen durch einen Existenzquantor in Verbindung mit einem Quantor mit begrenzter Fehlerwahrscheinlichkeit ausdrückbar, wenn man den Mechanismus für die Orakelzugriffe einschränkt. Es zeigt sich, daß ein Mengenquantor mit begrenztem Fehler, angewendet auf *PSPACE*, eliminiert werden kann, ohne die Klasse zu verändern.

Abschließend gehen wir auf die Relativierbarkeit der Resultate ein.

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

"Todo esfuerzo que no se sostiene se pierde."

Gabriela Mistral

Quantifiers play an important role in the complexity theory. Take for example the classes of the (arithmetical) polynomial-time hierarchy, which can be characterized by polynomial length bounded existential and universal word quantifiers on the base of *P*. The main subject of this thesis is the investigation of the computational power of quantifiers applied to well-known complexity classes. In special, we are interested in existential, universal and probabilistic bounded error quantifiers ranging over words and sets of words (oracles if we think in a Turing machine model). In addition to the standard oracle access mechanism, we consider also quantifiers ranging over oracles whose access is in a certain way restricted.

This chapter is organized as follows: We start giving a brief overview on the theme of this thesis (§1.1). Then, an outline of the results is exhibited (§1.2). Finally, we present papers related to this thesis and some interesting results (§1.3).

1.1 A Brief Overview

Complexity theory is the area of computer science that tries to classify computational problems in terms of the amount of computational resources needed to solve them. Intuitively, this is the field which deals with the reasons why certain problems are hard to be solved by computers. A traditional way to accomplish this task has been to consider computational structures (normally as computational models) which capture computational problems and generate complexity classes. Then, properties and relationships among these complexity classes are investigated. This approach provides a more or less abstract framework to study the nature of these problems.

Interesting complexity classes can be defined (or characterized) by quantifiers on the base of some other complexity class. A classical example are the classes of the (arithmetic) polynomial-time hierarchy [SM73, Sto77, Wra77], which are characterized by the existential and the universal quantifier on the base of *P*. These quantifiers vary over words whose lengths are polynomially bounded in the length of the input.

However, quantifiers of higher types, i.e. quantifiers ranging over sets of words, have also called the attention of the research community. Using the Turing machine model this means that the quantifiers vary over oracles. To our knowledge, Orponen [Orp83] was the first who studied in 1983 quantifiers of higher types in complexity theory. He related a hierarchy defined by existential and universal set quantifiers on the base of the class *PH* with the classes of the exponential-time alternation hierarchy.

In 1988, Fortnow, Rompel, and Sipser [FRS88] characterized the power of multi-prover interactive proof systems (*MIP*) by an existential set quantifier on the base of the polynomial-time bounded error probability class *BPP*. This characterization of *MIP* has motivated us to answer the question of whether a Fortnow-Rompel-Sipser like result could also be established for (one-prover) interactive proof systems (*IP*). In cooperation with Wagner [BW96] we showed in 1996 that this is possible by restricting the oracle access mechanism as follows: every query must be an extension of previous query. In other words, the series of queries in any computation has the form $u_1, u_1u_2, u_1u_2u_3, \ldots$. The quantifiers varying over oracles with this kind of restricted access are called quantifiers of type 1 whereas quantifiers varying over oracles with unrestricted access are called quantifiers of type 2. Word quantifiers[1] are called quantifiers of type 0. These characterizations of *MIP* and *IP* motivated us to study the set quantifiers of higher order in more detail. Thus, in the same work [BW96] we continued Orponen's investigations and defined a hierarchy over *P* using all three types of existential and universal quantifiers, the so called analytic polynomial-time hierarchy. It was shown that each class of this hierarchy coincides with one of the classes Σ_k^p and Π_k^p $(k \geq 0)$ of the (arithmetic) polynomial-time hierarchy, *PSPACE*, or one of the classes Σ_k^{exp} and Π_k^{exp} $(k \geq 1)$ of the exponential-time alternation hierarchy and vice versa. These results tighten up Orponen's result on quantifiers of type 2, and we proved relations like $\exists^1$co*NP* $=$ *PSPACE* and $\exists^1\exists^1$co*NP* $= \exists^2$co*NP* $=$ *NEXPTIME* giving the type of quantifiers in the exponent. Generally, for the classes of the analytic polynomial-time hierarchy two quantifiers of type 1 are as powerful as one quantifier of type 2. In 1990, Shamir [Sha90] proved *IP* $=$ *PSPACE* and Babai, Fortnow, and Lund [BFL90] showed *MIP* $=$ *NEXPTIME*. Comparing these results with the oracle characterization of interactive proof systems we get *IP* $= \exists^1$*BPP* $= \exists^1$co*NP* $=$ *PSPACE* and *MIP* $= \exists^2$*BPP* $= \exists^2$co*NP* $=$ *NEXPTIME*. However, the *BPP* part of these results is not relativizable [FS88, FRS88] whereas the co*NP* part is valid under every relativization.

In 1992, Arora and Safra [AS92] introduced the notion of probabilistically checkable proofs (*PCP*) to "scale down" the Babai, Fortnow, and Lund's result. The class *PCP*$(r(n), q(n))$ can be defined as an existential set quantifier applied to *BPP*, where an underlying machine is allowed to use $O(r(n))$ random bits for its computation and queries the oracle $O(q(n))$ times. Arora and Safra proved *NP* $=$ *PCP*$\big(\log n, (\log\log n)^{O(1)}\big)$ and few weeks later Arora, Lund, Motwani, Sudan, and Szegedy [ALM$^+$92] improved this result showing *NP* $=$ *PCP*$(\log n, O(1))$. In 1997, Vollmer and Wagner [VW97] gave a detailed discussion of scaling down results in this area.

[1]In this thesis, we consider word quantifiers ranging over words whose lengths can be polynomially or logarithmically bounded in the length of the input. Thus, we will further distinguish type 0 quantifiers into type p and type log, respectively.

Some constructions in [BW96] result in a bounded number of oracle queries. This fact and the interesting results obtained in the study of the *PCP* classes [AS92, ALM$^+$92], which limit the number of oracle queries, motivated us to continue the study of the analytic polynomial-time hierarchy classes but now considering the number of oracle queries that an oracle machine can ask during its computation. This hierarchy is called bounded analytic polynomial-time hierarchy. In cooperation with Wagner [BW97] we showed that each class of this hierarchy having a certain normal form coincides with *NP*, co*NP*, *PSPACE*, or one of the classes Σ_k^{exp} and Π_k^{exp} $(k \geq 1)$ of the exponential-time alternation hierarchy and vice versa. In addition, we proved that all these characterizations remain valid if the oracle machines are allowed to make only parallel queries, i.e. they have to form a list of all queries before any of them is queried to the oracle. In particular, all the characterizations for the classes of the analytic polynomial-time hierarchy [BW96] remain also valid under the parallel queries restriction.

Finally, let us mention that in 1996, Book, Vollmer, and Wagner [BVW96] investigated the power of probabilistic quantifiers of type 2.

1.2 Outline of this Thesis

The aim of this thesis is to investigate complexity classes defined (or characterized) by existential, universal and probabilistic bounded error quantifiers applied to well-known complexity classes. We consider word quantifiers and two types of set quantifiers, namely these of type 1 and 2. Restrictions on the number of oracle queries are also examined. Next, we present an overview of the organization of this thesis.

Chapter 2: Preliminaries

We introduce basic notations and concepts as well as our computational models. Furthermore, some complexity classes are defined. However, quantifiers will be defined in the respective chapter as needed.

Chapter 3: The Analytic Polynomial-Time Hierarchy

We investigate a hierarchy defined by existential and universal quantifiers varying over words and oracles of type 1 and 2 on the base of the class *P*. This hierarchy, which extends the (arithmetic) polynomial-time hierarchy, is called the *analytic polynomial-time hierarchy*. It is shown that each class of this hierarchy coincides with one of the classes Σ_k^p and Π_k^p $(k \geq 0)$ of the (arithmetic) polynomial-time hierarchy, *PSPACE*, or one of the classes Σ_k^{exp} and Π_k^{exp} $(k \geq 1)$ of the exponential-time alternation hierarchy and vice versa. These results tighten up Orponen's result [Orp83] on quantifiers of type 2. An algorithm is established which allows to find out the corresponding well-known class in an easy way.

Chapter 4: Bounding Queries in the Analytic Polynomial-Time Hierarchy

We examine a hierarchy which refines the analytic polynomial-time hierarchy by considering restrictions on the number of oracle queries. This hierarchy is called *bounded analytic polynomial-time hierarchy*. We characterize classes of this hierarchy by well-known complexity classes. In particular, for classes from this hierarchy having a certain normal form we show that each of these classes coincides with one of the classes *NP*, co*NP*, *PSPACE*, Σ_k^{exp} or Π_k^{exp} ($k \geq 1$) and vice versa. All these characterizations remain valid if the queries are asked in a nonadaptive form, i.e. in "parallel". In special, all the characterizations for the classes of the analytic polynomial-time hierarchy (Chapter 3) also remain valid under the parallel queries restriction.

Chapter 5: The Analytic Logarithmic-Space Hierarchy

We investigate a logarithmic-space hierarchy built up by word and set quantifiers of type 1 and 2, which can intuitively be interpreted as the analytic polynomial-time hierarchy defined over *L* instead of *P*. This hierarchy is called the *analytic logarithmic-space hierarchy*. We show that every class of this hierarchy can be represented in a certain normal form, where the last quantifier is either a word quantifier or a set quantifier of type 2. Furthermore, we characterize classes of this hierarchy by well-known complexity classes. In particular, it is shown that each class in this normal form, whose last quantifier is a word quantifier, coincides with one of the classes *L*, Σ_k^p or Π_k^p ($k \geq 1$) and vice versa.

Chapter 6: Probabilistic Bounded Error Operators

We consider probabilistic bounded error quantifiers. We show under which general conditions the type 2 of a bounded error set quantifier can be reduced to type 1. Furthermore, interesting characterizations are presented. For example, we characterize (one prover) interactive proof systems by an existential set quantifier of type 1 and a probabilistic bounded error word quantifier applied to *P*, and show that a bounded error set quantifier of type 1 applied to *PSPACE* can be eliminated without changing the class in question. Finally, we discuss the relativizability of results presented so far.

Conclusions

Instead of making a separate chapter for conclusions, we prefer to discuss the results in the respective chapters.

1.3 Related Papers and Interesting Results

The papers related to this thesis are the following:

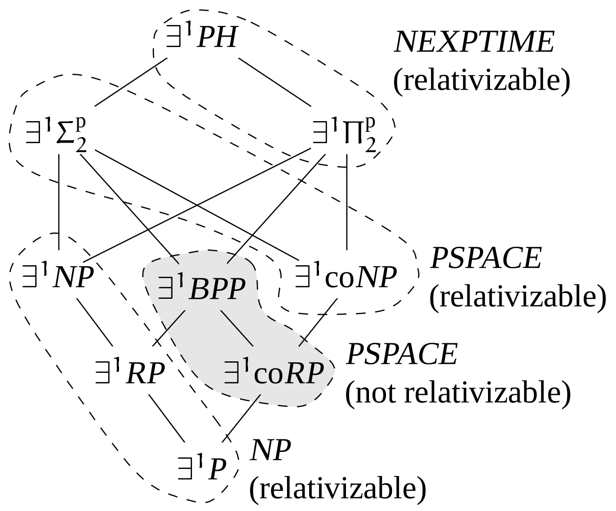
(1) The results in Chapter 3 and Theorem 6.15.

☞ H. Baier and K. W. Wagner. The analytic polynomial-time hierarchy. Technical Report 148, Institut für Informatik, Universität Würzburg, Germany, September 1996. To appear in Mathematical Logic Quarterly (formerly: Zeitschrift für Mathematische Logik und Grundlagen der Mathematik).

(2) A part of the results in Chapter 4 (§4.3 and §4.4).

☞ H. Baier and K. W. Wagner. Bounding queries in the analytic polynomial-time hierarchy. Technical Report 178, Institut für Informatik, Universität Würzburg, Germany, August 1997. To appear in Theoretical Computer Science.

Finally, Figures 1.1 and 1.2 give us a preview of some of our results.



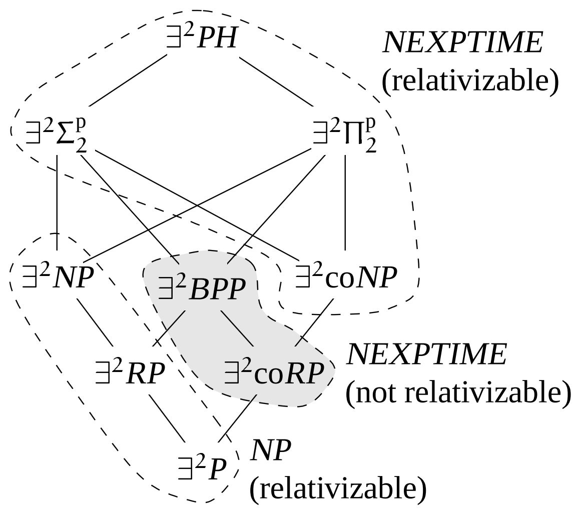
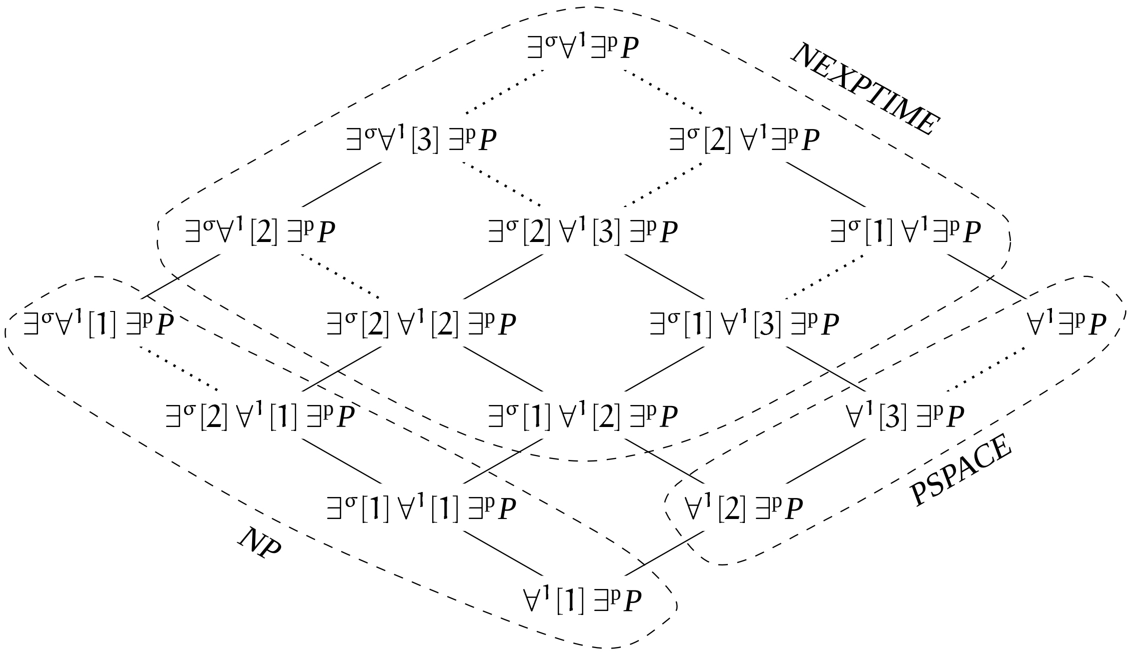


Figure 1.1: *Relativized world of classes $\exists^1\mathcal{K}$ and $\exists^2\mathcal{K}$, where $\mathcal{K}$ are interesting classes within the polynomial-time hierarchy.*


$\exists^\sigma\forall^1\exists^p P$
$\exists^\sigma\forall^1[3]\,\exists^p P$
$\exists^\sigma[2]\,\forall^1\exists^p P$
$\exists^\sigma\forall^1[2]\,\exists^p P$
$\exists^\sigma[2]\,\forall^1[3]\,\exists^p P$
$\exists^\sigma[1]\,\forall^1\exists^p P$
$\exists^\sigma\forall^1[1]\,\exists^p P$
$\exists^\sigma[2]\,\forall^1[2]\,\exists^p P$
$\exists^\sigma[1]\,\forall^1[3]\,\exists^p P$
$\forall^1\exists^p P$
$\exists^\sigma[2]\,\forall^1[1]\,\exists^p P$
$\exists^\sigma[1]\,\forall^1[2]\,\exists^p P$
$\forall^1[3]\,\exists^p P$
$\exists^\sigma[1]\,\forall^1[1]\,\exists^p P$
$\forall^1[2]\,\exists^p P$
$\forall^1[1]\,\exists^p P$
NEXPTIME
PSPACE
NP

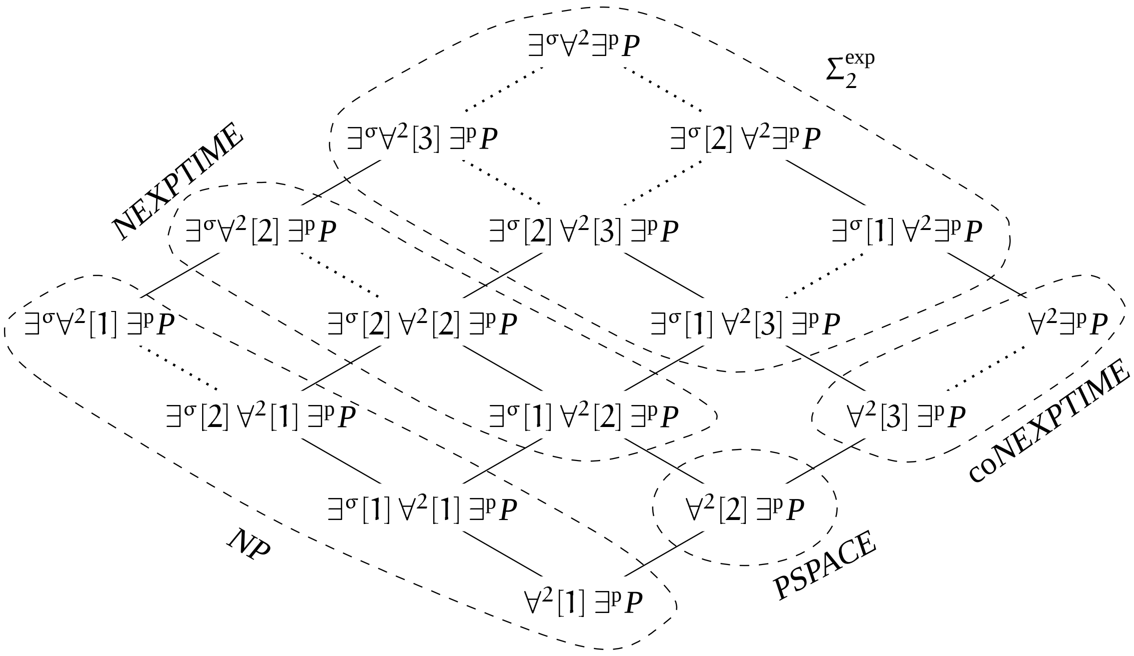

$\exists^\sigma\forall^2\exists^p P$
Σ_2^{exp}
$\exists^\sigma\forall^2[3]\,\exists^p P$
$\exists^\sigma[2]\,\forall^2\exists^p P$
$\exists^\sigma\forall^2[2]\,\exists^p P$
$\exists^\sigma[2]\,\forall^2[3]\,\exists^p P$
$\exists^\sigma[1]\,\forall^2\exists^p P$
$\exists^\sigma\forall^2[1]\,\exists^p P$
$\exists^\sigma[2]\,\forall^2[2]\,\exists^p P$
$\exists^\sigma[1]\,\forall^2[3]\,\exists^p P$
$\forall^2\exists^p P$
$\exists^\sigma[2]\,\forall^2[1]\,\exists^p P$
$\exists^\sigma[1]\,\forall^2[2]\,\exists^p P$
$\forall^2[3]\,\exists^p P$
$\exists^\sigma[1]\,\forall^2[1]\,\exists^p P$
$\forall^2[2]\,\exists^p P$
$\forall^2[1]\,\exists^p P$
NEXPTIME
coNEXPTIME
PSPACE
NP

Figure 1.2: *Classes* $\exists^\sigma[r]\,\forall^\tau[s]\,\exists^p P$ *with* $\sigma, \tau \in \{1, 2\}$ *and* $r, s : \mathbb{N} \to \mathbb{N}$*, such that* $r \geq 0$ *and* $s \geq 1$ *(the term* $[\cdot]$ *represents the number of oracle queries allowed). In the left direction we increase* r *and in the right direction we increase* s *(for short we write* $\forall^\tau[s]\,\exists^p P$ *instead of* $\exists^\sigma[0]\,\forall^\tau[s]\,\exists^p P$*).*

CHAPTER 2

Preliminaries

"Se não houver frutos,
valeu a beleza das flores,
se não houver flores,
valeu a sombra das folhas,
se não houver folhas,
valeu a intenção da semente."

Henfil

In this chapter, we present some notations and concepts which are required throughout this work. Basic familiarity with the most popular complexity theory notations and concepts is assumed (we refer the reader to standard textbooks of complexity theory such as [BDG95, Pap94]). Hence, only non-standard notations and concepts will be covered in detail.

This chapter is organized as follows: We start introducing some basic notations and concepts (§2.1). Next, we present our computational models (§2.2) and define some complexity classes (§2.3). Quantifiers will be defined in the respective chapter as needed.

2.1 Basic Notations and Concepts

We will study classes of languages whose instances consist of word and set of words. Next, some basic notations and concepts are presented, namely alphabets, words, languages, sets and functions.

An *alphabet* is any finite nonempty set Σ of symbols. A *word* over Σ is a finite sequence of symbols from Σ. In particular, ε represents the *empty word*, i.e. the word consisting of zero symbols. Given a word u over Σ containing n symbols, we say that the *length of* u, denoted by $|u|$, is n. For two words u and v over Σ, let uv (and sometimes $u \cdot v$) represent the concatenation of u and v. Given a word u over Σ and an integer n, define $u(n)$ as the n-th symbol of u, and u^n inductively by: $u^0 =_{\mathrm{df}} \varepsilon$ and $u^n =_{\mathrm{df}} u \cdot u^{n-1}$ for all $n \geq 1$. The set of all words over Σ including (not including) ε is denoted by Σ^* (Σ^+, respectively). Furthermore, define Σ^n ($\Sigma^{\leq n}$) as the set of all words over Σ of length n (at most n, respectively). A subset of Σ^* is also called a *language over* Σ and sometimes an *oracle over* Σ. The *complement of a language* $L \subseteq \Sigma^*$ is the language $\overline{L} =_{\mathrm{df}} \Sigma^* \setminus L$ and the *complement of a class* $\mathcal{K}$ *of languages* is the class co$\mathcal{K} =_{\mathrm{df}} \left\{\overline{L} : L \in \mathcal{K}\right\}$.

Let $\mathbb{N}$ denote the set of natural numbers and $\mathbb{N}_+$ the set of natural numbers greater than 0. For a set U, let $||U||$ be the cardinality of U and $\mathcal{P}(U)$ be the power set of U. The *characteristic function of a set* U is the function χ_U defined by

$$\chi_U(v) =_{\text{df}} \begin{cases} 1 & \text{if } v \in U, \\ 0 & \text{otherwise.} \end{cases}$$

There exists a natural bijection between $\{0,1\}^*$ and $\mathbb{N}$, when standard lexicographical order is used. For $i \in \mathbb{N}$, let $\text{lex}(i)$ ($\text{lex}_n(i)$) be the i-th word of $\{0,1\}^*$ ($\{0,1\}^n$, respectively) in lexicographical order. Thus, for a set $U \subseteq \{0,1\}^*$ we will also write $i \in U$. We then mean $\text{lex}(i) \in U$. Furthermore, for $u \in \{0,1\}^n$ let $\overleftrightarrow{u} \in \{0,1\}^n$ denote the predecessor of u in $\{0,1\}^n$ in lexicographical order ($\overleftrightarrow{0^n}$ is undefined).

For functions $f, g : \mathbb{N} \to \mathbb{N}$ we say that $f \in O(g)$ if there are positive integers c and n_0 such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$. Finally, the symbol $\circ$ denotes the traditional composition of functions $(f \circ g)(x) =_{\text{df}} f(g(x))$. Without loss of generality we restrict ourselves in this work to the standard alphabet $\Sigma = \{0,1\}$.

2.2 Computational Models

Standard Turing machines (§2.2.1) and their variations (§2.2.2) will be our formal computational models to accept languages.

2.2.1 Well-Known Computational Models

The standard Turing machine models are presumed to be known (see [BDG95, Pap94]): deterministic, nondeterministic, probabilistic, alternating and oracle Turing machines. We shortly restate these notions. Each non-final configuration of a deterministic Turing machine has only one successor whereas each non-final configuration of a nondeterministic Turing machine can have several successors and one of them is guessed. For simplicity, we suppose that each non-final configuration of a nondeterministic Turing machine has exactly two successors. A probabilistic Turing machine is similar to a nondeterministic Turing machine with the difference that in the former the successor configuration is chosen at random while in the second it is guessed.

An alternating Turing machine is a deterministic Turing machine which has in addition two special types of states (configurations): existential and universal. The acceptance of an alternating machine depends on these special states in the following way. At least one of the successor configurations of an existential configuration must lead to an accepting configuration, whereas all successor configurations of a universal configuration must lead to an accepting configuration. For $k \geq 1$, a Σ_k*-alternating* (Π_k*-alternating*) *Turing machine* is an alternating Turing machine starting with an existential (universal, respectively) state and having at most $k-1$ alternations between existential and universal states on every computation path. By convention the Σ_0-alternating and Π_0-alternating Turing machines are deterministic.

For a Turing machine M on input x being deterministic, nondeterministic, probabilistic or alternating, its *computation tree*, denoted $\beta_M(x)$, is a possibly infinite tree whose nodes are configurations, the root being the initial configuration, and for any node α, its sons are those configurations which are immediate successors of α. Note that for M being deterministic, its computation trees are also paths. For M not being deterministic, without loss of generality we assume that the trees are binary, i.e. each non-final configuration (node) has exactly two successors. An *accepting path* of $\beta_M(x)$ is a path in $\beta_M(x)$ which has the same root node and ends in an accepting state. Furthermore, for M being alternating, a *good subtree* of $\beta_M(x)$ is a subtree of $\beta_M(x)$ which has the same root node and includes both successors of an universal configuration and exactly one successor of an existential configuration. An *accepting subtree* of $\beta_M(x)$ is a good subtree of $\beta_M(x)$ which has only accepting paths.

An oracle Turing machine may ask queries to an oracle during its computation in the usual way: The machine writes a query on a special tape called *query tape*. When the machine transfers into a special *query state* then it switches automatically into a special state either *yes* or *no* depending on whether the current query belongs to the oracle or not. The oracle is required to be fixed previously to the computation of the machine.

In an interactive proof system (multi-prover interactive proof system) a probabilistic Turing machine interacts with a prover (provers, respectively) where every prover tries to convince the probabilistic Turing machine to accept the input. The input to the protocol (proof system) is known to the Turing machine and the provers, and the Turing machine interacts with a prover sending a message and receiving an answer. The provers can not interact with each other and there are no restrictions on their power [Pap94, pp. 289, 506].

A Turing machine M is called *polynomial-time* (*polynomial-space*) if there exists a polynomial p such that M halts on every path of every instance x in an accepting or rejecting state using no more than $p(|x|)$ steps (tape cells, respectively). Furthermore, M is called *logarithmic-space* if there exists a constant $c \in \mathbb{N}$ such that M halts on every path of every instance x in an accepting or rejecting state using no more than $c \cdot \log|x|$ tape cells, and M is called *exponential-time* if there exists a polynomial p such that M halts on every path of every instance x in an accepting or rejecting state using no more than $2^{p(|x|)}$ steps.

The above definitions can be combined. Thus, an oracle Turing machine may be e.g. deterministic, nondeterministic, probabilistic or alternating. Finally, for a Turing machine M define

$$\begin{array}{lll} M \text{ is deterministic} & \longrightarrow & L(M) =_{\text{df}} \{x : \beta_M(x) \text{ is an accepting path}\} \\ M \text{ is nondeterministic} & \longrightarrow & L(M) =_{\text{df}} \{x : \beta_M(x) \text{ contains an accepting path}\} \\ M \text{ is alternating} & \longrightarrow & L(M) =_{\text{df}} \{x : \beta_M(x) \text{ contains an accepting subtree}\} \end{array}$$

as the *language accepted by* M.

Where no confusion arises we write for simplicity Turing machine instead of deterministic Turing machine.

2.2.2 Turing Machines of Type $\sigma_1 \ldots \sigma_k$

Variations of standard Turing machines will also be used to accept languages.

We turn to hierarchies built up by word and set operators (defined by word and set quantifiers, respectively). Thus, we have to start with suitable classes of languages whose instances consist of words and sets of words. Oracle Turing machines will be used to accept these languages. Every word input is given on a separate input tape and every set input is given as an oracle. The machines have a special query tape for every oracle. The oracles can be classified according to the type of queries that can be made by an oracle machine. An oracle is an *input of type 1* (*input of type 2*) if the query on the corresponding query tape is not erased (erased, respectively) after each query. Hence, the next query made to an oracle of type 1 is an extension of the previous query. Note that formally inputs of type 1 and 2 are the same objects, namely sets of words. We will call a word an *input of type 0* (see Figure 2.1).

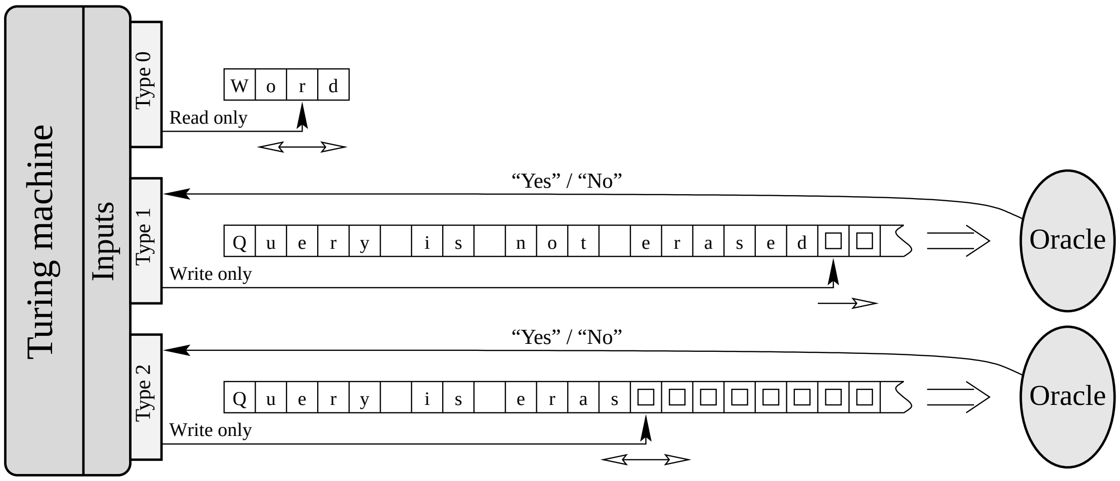


Figure 2.1: *Inputs of type 0, 1 and 2.*

For $k \geq 1$ and $\sigma_1, \ldots, \sigma_k \in \{0, 1, 2\}$, we say that a Turing machine is of *type* $\sigma_1 \ldots \sigma_k$ if it processes instances of the form $(X_1, \ldots, X_k)$, where X_i is an input of type σ_i for $i = 1, \ldots, k$. Such a machine has $\|\{i : \sigma_i = 0\}\|$ ordinary input tapes and $\|\{i : \sigma_i \in \{1, 2\}\}\|$ query tapes. The *length of an instance* $X = (X_1, \ldots, X_k)$, denoted $|X|$, is defined by $|X| =_{df} \sum_{1 \leq i \leq k, \sigma_i = 0} |X_i|$, i.e. the sum of the length of the word inputs. In what follows we assume that X contains at least one word input.

We define computation trees and accepting paths as in §2.2.1. For a Turing machine M of type $\sigma_1 \ldots \sigma_k$ on input $(X_1, \ldots, X_k)$ being deterministic or nondeterministic (X_i is an input of type σ_i), its *computation tree,* denoted $\beta_M(X_1, \ldots, X_k)$, is a possibly infinite tree whose nodes are configurations, the root being the initial configuration, and for any node α, its sons are those configurations which are immediate successors of α. Obviously, for M being deterministic, its computation trees are also paths. For M being nondeterministic, without loss of generality we also assume here that the trees are binary. An *accepting path* of $\beta_M(X_1, \ldots, X_k)$ is a path in $\beta_M(X_1, \ldots, X_k)$ which has the same root node and ends in an accepting configuration.

A Turing machine M of type $\sigma_1 \ldots \sigma_k$ is called *polynomial-time* (*polynomial-space*) if there exists a polynomial p such that M halts on every path of every instance $X = (X_1, \ldots, X_k)$ in an accepting or rejecting state using no more than $p(|X|)$ steps (tape cells, respectively). Furthermore, M is called *logarithmic-space* if there exists a constant $c \in \mathbb{N}$ such that M halts on every path of every instance $X = (X_1, \ldots, X_k)$ in an accepting or rejecting state using no more than $c \cdot \log |X|$ tape cells. Note that the space bound applies also to the length of oracle queries.

Finally, for a Turing machine M of type $\sigma_1 \ldots \sigma_k$ define

M is deterministic:

$$L(M) =_{\text{df}} \{(X_1, \ldots, X_k) : \beta_M(X_1, \ldots, X_k) \text{ is an accepting path}\}$$

M is nondeterministic:

$$L(M) =_{\text{df}} \{(X_1, \ldots, X_k) : \beta_M(X_1, \ldots, X_k) \text{ contains an accepting path}\}$$

as the *language accepted by* M.

2.3 Complexity Classes

Complexity classes are now defined using the Turing machine models presented in §2.2. The corresponding relativized classes can be obtained in a standard way. Let $\mathcal{K}$ be a complexity class defined by a suitable type of Turing machines and A be an oracle. The complexity class $\mathcal{K}^A$ is defined as the class of all languages which can be accepted by Turing machines whose type is the same as for the class $\mathcal{K}$ but having in addition access to the oracle A. For classes $\mathcal{K}_1$ and $\mathcal{K}_2$, the complexity class $\mathcal{K}_1^{\mathcal{K}_2}$ is the union of all classes $\mathcal{K}_1^A$ with $A \in \mathcal{K}_2$.

We will describe the most popular complexity classes intuitively (§2.3.1), and the nonstandard ones more precisely (§2.3.2).

2.3.1 Well-Known Complexity Classes

The complexity class *P* (*NP*) is defined as the class of all languages which can be accepted by polynomial-time deterministic (nondeterministic, respectively) Turing machines. The most investigated nondeterministic complexity class is *NP*, this class contains many important problems. A good overview of problems in this class can be found in Garey and Johnson [GJ79].

“Space” classes play also an important role in complexity theory. The complexity class *L* (*NL*) is defined as the class of all languages which can be accepted by logarithmic-space deterministic (nondeterministic, respectively) Turing machines. Another well-known “space” class is *PSPACE*, the class of all languages which can be accepted by polynomial-space deterministic Turing machines. It was shown [Sav70] that the class of all languages which can be accepted by polynomial-space nondeterministic Turing machines coincides with *PSPACE*.

The classes of the (arithmetic) polynomial-time hierarchy, introduced by Meyer and Stockmeyer [MS72], play an important role in complexity theory. These classes are defined inductively as follows (the classes Θ_k^p were introduced later by Wagner [Wag90]):

$$\Theta_0^p = \Sigma_0^p = \Pi_0^p =_{\text{df}} P$$
$$\Theta_k^p =_{\text{df}} L^{\Sigma_{k-1}^p},\ \Sigma_k^p =_{\text{df}} NP^{\Sigma_{k-1}^p},\ \Pi_k^p =_{\text{df}} \text{co}NP^{\Sigma_{k-1}^p} \quad \text{for } k \geq 1.$$

Furthermore, define *PH* as the union of all classes of the polynomial-time hierarchy. It is well-known [Wag90] that $\Sigma_k^p \cup \Pi_k^p \subseteq \Theta_{k+1}^p \subseteq \Sigma_{k+1}^p \cap \Pi_{k+1}^p$ for all $k \geq 0$. An alternative way of looking at these classes is using alternating Turing machines [SM73, Sto77, Wra77]: For $k \geq 0$, Σ_k^p (Π_k^p) is the class of all languages which can be accepted by Σ_k-alternating (Π_k-alternating, respectively) polynomial-time Turing machines. Let *APTIME* be the class of all languages which can be accepted by alternating polynomial-time Turing machines. In [CKS81], "time" and "space" classes were related, such *APTIME* = *PSPACE*.

In 1990, Wagner [Wag90] extended the definition of the Boolean hierarchy: For a function $r : \mathbb{N} \to \mathbb{N}$ define

$$A \in NP(r) \iff_{\text{df}} \text{there exists a set } B \in NP \text{ such that } \chi_B(x, i+1) \leq \chi_B(x, i) \text{ for all } i$$
$$\text{and } \chi_A(x) \equiv \max\{i : 1 \leq i \leq r(|x|) \text{ and } (x, i) \in B\} \mod 2$$

He showed $\Theta_2^p = NP\big(n^{O(1)}\big)$.

In 1977, Gill [Gil77] introduced probabilistic classes. Let *BPP* be the class of all languages which can be accepted by polynomial-time bounded error probabilistic Turing machines, i.e. the class of languages recognized by polynomial-time probabilistic Turing machines whose error probability is bounded above by some positive constant $\varepsilon < \frac{1}{2}$. Furthermore, define *RP* as the class of all languages which can be accepted by polynomial-time one-sided bounded error probabilistic Turing machines, i.e. the class of languages recognized by polynomial-time probabilistic Turing machines which have zero error probability for instances not in the language and error probability bounded by some positive constant $\varepsilon < \frac{1}{2}$ for instances in the language.

The complexity class *EXPTIME* (*NEXPTIME*) is defined as the class of all languages which can be accepted by exponential-time deterministic (nondeterministic, respectively) Turing machines. Exponential-time hierarchies can be defined in different ways. Here we adopt the definition which employs alternating Turing machines, the so called exponential-time alternation hierarchy. For $k \geq 0$, let Σ_k^{exp} (Π_k^{exp}) be the class of all languages which can be accepted by Σ_k-alternating (Π_k-alternating, respectively) exponential-time Turing machines. Furthermore define *EXPH* as the union of all classes of the exponential-time alternation hierarchy. Obviously, *EXPTIME* $= \Sigma_0^{\text{exp}} = \Pi_0^{\text{exp}}$ and *NEXPTIME* $= \Sigma_1^{\text{exp}}$.

Finally, let *IP* (*MIP*) be the class of all languages which can be accepted by interactive proof systems (multi-prover interactive proof systems, respectively) with the probabilistic Turing machine of the protocol being also polynomial-time bounded error. It has been shown [Sha90, BFL90] that *IP* = *PSPACE* and *MIP* = *NEXPTIME*. However, these results do not remain valid in every relativized world [FS88, FRS88], i.e. there exist oracles A and B such that $IP^A \neq PSPACE^A$ and $MIP^B \neq NEXPTIME^B$.

The diagram of Figure 2.2 summarizes the known relationships between the complexity classes presented above. Other important properties about these classes can be found in [WW86, BDG95, BDG90, Pap94].

2.3.2 Classes of Type $\sigma_1 \dots \sigma_k$

Next, we define classes which are the starting point for building up hierarchies here considered. For $k \geq 1$ and $\sigma_1, \dots, \sigma_k \in \{0, 1, 2\}$ define

$$\begin{aligned}
L^{\sigma_1 \dots \sigma_k} &=_{\mathrm{df}} \{L(M) : M \text{ is a logarithmic-space deterministic} \\ &\qquad \text{Turing machine of type } \sigma_1 \dots \sigma_k\} \\
P^{\sigma_1 \dots \sigma_k} &=_{\mathrm{df}} \{L(M) : M \text{ is a polynomial-time deterministic} \\ &\qquad \text{Turing machine of type } \sigma_1 \dots \sigma_k\} \\
NP^{\sigma_1 \dots \sigma_k} &=_{\mathrm{df}} \{L(M) : M \text{ is a polynomial-time nondeterministic} \\ &\qquad \text{Turing machine of type } \sigma_1 \dots \sigma_k\} \\
PSPACE^{\sigma_1 \dots \sigma_k} &=_{\mathrm{df}} \{L(M) : M \text{ is a polynomial-space deterministic} \\ &\qquad \text{Turing machine of type } \sigma_1 \dots \sigma_k\}
\end{aligned}$$

which are called *classes of type* $\sigma_1 \dots \sigma_k$. Such a class consists only of languages $L \subseteq \{0,1\}^{(\sigma_1)} \times \dots \times \{0,1\}^{(\sigma_k)}$, where we define $\{0,1\}^{(0)} =_{\mathrm{df}} \{0,1\}^*$ and $\{0,1\}^{(\sigma)} =_{\mathrm{df}} \mathcal{P}(\{0,1\}^*)$ for $\sigma = 1, 2$ (remember that formally inputs of type 1 and 2 are the same objects, namely sets of words).

The classes of type $\sigma_1 \dots \sigma_k$ are the starting point of our research about operators. Applying operators to these classes we can define and characterize other complexity classes. This will be treated in the following chapters.

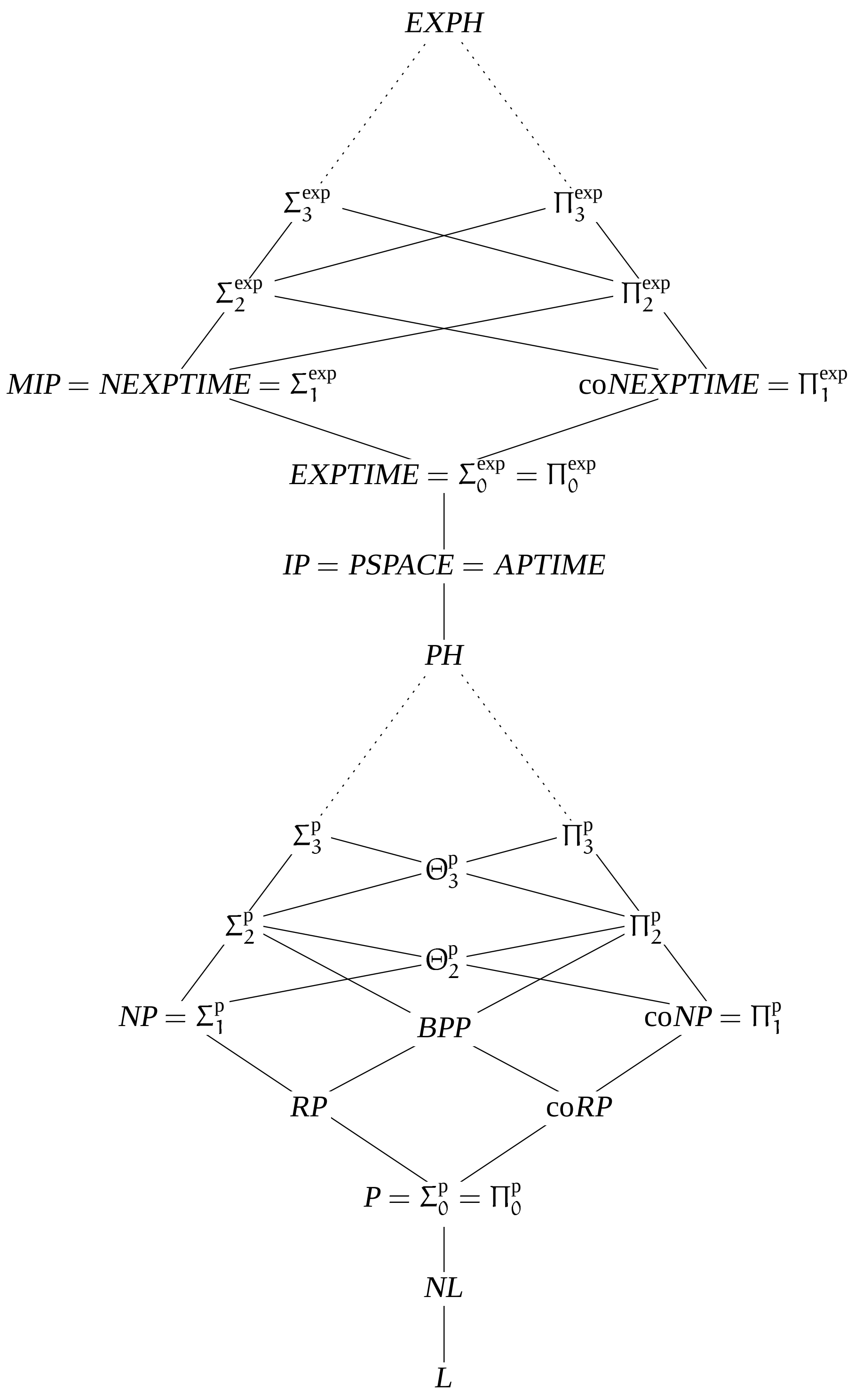


Figure 2.2: *Relationships between well-known complexity classes.*

CHAPTER 3

The Analytic Polynomial-Time Hierarchy

"There are more things in heaven and earth,
than are dreamt of in your philosophy."

Shakespeare

In the present chapter, we investigate a hierarchy defined by existential and universal quantifiers varying over words and oracles of type 1 and 2 on the base of the class *P*. This hierarchy of classes is called the *analytic polynomial-time hierarchy*. It is shown that each class of this hierarchy coincides with one of the classes Σ_k^p and Π_k^p $(k \geq 0)$ of the (arithmetic) polynomial-time hierarchy, *PSPACE*, or one of the classes Σ_k^{exp} and Π_k^{exp} $(k \geq 1)$ of the exponential-time alternation hierarchy and vice versa.

An outline of this chapter follows: We first give some more notations and define the existential and universal quantifiers and the analytic polynomial-time hierarchy (§3.1). In order to prove our main result (§3.4), the investigation on the power of the classes of the analytic polynomial-time hierarchy is divided in two parts:

(a) Using equivalence rules we show that every class of this hierarchy can be represented in a certain normal form (§3.2).

(b) It is shown that each class in this normal form coincides with a well-known complexity class (§3.3).

These results make possible to establish an algorithm which allows to find out the corresponding well-known class in an easy way (§3.4). Finally, we make some comments about the results (§3.5).

3.1 The Operators and the Hierarchy

Next, we define the existential and universal quantifiers (§3.1.1) and the analytic polynomial-time hierarchy (§3.1.2). But first, let us give some more notations to help us in the proofs of the results. For every set $U \subseteq \{0,1\}^*$ and every $m \in \mathbb{N}$ we define the word $\langle U, m\rangle =_{\text{df}} \chi_U(1)\,\chi_U(11)\,\chi_U(111)\ldots\chi_U(1^m)$. For $u \in \{0,1\}^*$ and $U \subseteq \{0,1\}^*$, define the set $u\backslash U =_{\text{df}} \{w : uw \in U\}$. Finally, for $a \in \{0,1\}$ and $u \in \{0,1\}^*$, define the encodings $\widetilde{ua} =_{\text{df}} \widetilde{u}1a$ $(\widetilde{\varepsilon} =_{\text{df}} \varepsilon)$ and $\widehat{ua} =_{\text{df}} \widehat{u}aa$ $(\widehat{\varepsilon} =_{\text{df}} \varepsilon)$.

3.1.1 The Existential and Universal Operators

We will examine a polynomial-time hierarchy defined by word and set quantifiers, namely the existential and universal. The classes $P^{\sigma_1\cdots\sigma_k}$ are the starting point for building up this hierarchy. Next, we define inductively new classes and in parallel the existential and universal quantifiers. Let $k \geq 1$ and $\sigma_1, \ldots, \sigma_k, \sigma \in \{0,1,2\}$. If $\mathcal{K}$ is a class of type $\sigma_1 \ldots \sigma_k \sigma$ then

For $\sigma = 0$: $\exists^p\mathcal{K}$ and $\forall^p\mathcal{K}$ are classes of type $\sigma_1 \ldots \sigma_k$ which are defined as follows

$L \in \exists^p\mathcal{K} \iff_{\mathrm{df}}$ there exist an $L' \in \mathcal{K}$ and a polynomial p, such that

$$(X_1, \ldots, X_k) \in L \longleftrightarrow \exists x\Big(|x| \leq p\big(\sum_{\substack{\sigma_i=0\\ i\leq k}} |X_i|\big) \wedge (X_1, \ldots, X_k, x) \in L'\Big)$$

$L \in \forall^p\mathcal{K} \iff_{\mathrm{df}}$ there exist an $L' \in \mathcal{K}$ and a polynomial p, such that

$$(X_1, \ldots, X_k) \in L \longleftrightarrow \forall x\Big(|x| \leq p\big(\sum_{\substack{\sigma_i=0\\ i\leq k}} |X_i|\big) \rightarrow (X_1, \ldots, X_k, x) \in L'\Big)$$

(Using simple encoding arguments it is easy to see that one can use equivalently "$=$" instead of "$\leq$" in these definitions.)

For $\sigma = 1, 2$: $\exists^\sigma\mathcal{K}$ and $\forall^\sigma\mathcal{K}$ are classes of type $\sigma_1 \ldots \sigma_k$ which are defined as follows

$L \in \exists^\sigma\mathcal{K} \iff_{\mathrm{df}}$ there exists an $L' \in \mathcal{K}$, such that

$$(X_1, \ldots, X_k) \in L \longleftrightarrow \exists X\,((X_1, \ldots, X_k, X) \in L')$$

$L \in \forall^\sigma\mathcal{K} \iff_{\mathrm{df}}$ there exists an $L' \in \mathcal{K}$, such that

$$(X_1, \ldots, X_k) \in L \longleftrightarrow \forall X\,((X_1, \ldots, X_k, X) \in L')$$

To make clear which type of input is used, for $\tau \in \{p, 1, 2\}$ we also write $\exists^\tau X$ instead of $\exists X$, and $\forall^\tau X$ instead of $\forall X$.

Now, some abbreviations and definitions are presented. The set of existential and universal quantifiers is denoted by $\Gamma_p =_{\mathrm{df}} \left\{\exists^p, \exists^1, \exists^2, \forall^p, \forall^1, \forall^2\right\}$. For $k \geq 0$, $Q_1, \ldots, Q_k \in \{\exists, \forall\}$ and $\tau_1, \ldots, \tau_k \in \{p, 1, 2\}$, let $\tau(Q_1^{\tau_1} \ldots Q_k^{\tau_k}) =_{\mathrm{df}} \sigma_1 \ldots \sigma_k$ be the type of the operator (or quantifier) string $Q_1^{\tau_1} \ldots Q_k^{\tau_k}$, where $\sigma_i = 0$ if $\tau_i = p$ and $\sigma_i = \tau_i$ otherwise ($i = 1, \ldots, k$). For $Q = Q_1^{\tau_1} \ldots Q_k^{\tau_k}$ and $X = (X_1, \ldots, X_k)$ we write QX instead of $Q_1^{\tau_1}X_1 \ldots Q_k^{\tau_k}X_k$. Furthermore, we define $\overline{\exists} =_{\mathrm{df}} \forall$, $\overline{\forall} =_{\mathrm{df}} \exists$ and $\overline{Q} =_{\mathrm{df}} \overline{Q_1}^{\tau_1} \ldots \overline{Q_k}^{\tau_k}$.

Proposition 3.1. *Let* $\mu \in \{0,1,2\}^*$ *and* $Q \in \Gamma_p^*$. *Then* $\mathrm{co}QP^{\mu\tau(Q)} = \overline{Q}P^{\mu\tau(Q)}$.

Proof. Let $L \in QP^{\mu\tau(Q)}$. There exists an $L' \in P^{\mu\tau(Q)}$ such that

$$X \in L \iff QY\,((X,Y) \in L')$$

where the lengths of the word inputs in Y are bounded by suitable polynomials depending on the length of the word inputs in X. Negating the both sides of the equivalence, we get

$$\begin{aligned} X \in \overline{L} &\iff \neg QY\,((X,Y) \in L') \\ &\iff \overline{Q}Y\,(\neg\,(X,Y) \in L') \\ &\iff \overline{Q}Y\,\big((X,Y) \in \overline{L'}\big) \end{aligned}$$

Since $P^{\mu\tau(Q)}$ is closed under complement, we get the desired result. ❑

3.1.2 The Analytic Polynomial-Time Hierarchy

We are particularly interested in the classes of type 0, i.e. in "ordinary" classes of languages. In this case, the superscripts to P are omitted, i.e. for quantifier string $Q \in \Gamma_p^*$ we define $QP =_{df} QP^{0\tau(Q)}$. For $k \geq 0$ and $Q_1, \ldots, Q_k \in \{\exists, \forall\}$, it is well-known that each of the classes $Q_1^p \ldots Q_k^p P$ coincides with a class of the (arithmetic) polynomial-time hierarchy [SM73, Sto77, Wra77] and vice versa. To our knowledge, Orponen [Orp83] began in 1983 the study of the existential and universal quantifiers of type 2. He related a hierarchy defined by these quantifiers on the base of the class *PH* to the classes of the exponential-time alternation hierarchy.

Theorem 3.2. [Orp83] *For every $k \geq 1$ let $Q_k = \exists$ if k is odd and $Q_k = \forall$ otherwise. Then, $\exists^2\forall^2\exists^2 \ldots Q_k^2 PH = \Sigma_k^{\text{exp}}$.*

Next, the analytic polynomial-time hierarchy is defined. For quantifier strings $Q \in \Gamma_p^*$, the classes QP are called the classes of the *analytic polynomial-time hierarchy*. The class *APH* is defined as the union of all classes of the analytic polynomial-time hierarchy. Thus, this hierarchy extends the (arithmetic) polynomial-time hierarchy and, as we will see (Theorem 3.14), our results tighten up Orponen's result on quantifiers of type 2.

3.2 Equivalence Rules and a Normal Form

The purpose of this section is to show that every class of the analytic polynomial-time hierarchy can be represented in a certain normal form (§3.2.2), namely an alternating sequence of $\exists$-$\forall$-quantifiers on the base of P with the set quantifiers appearing left of the word quantifiers. To establish this result, we will apply equivalence rules (§3.2.1).

3.2.1 Inclusion and Equivalence Rules

We will use inclusion and equivalence rules to relate classes of the analytic polynomial-time hierarchy. These rules are used in the following sense: For $R, S \in \Gamma_p^*$, the *inclusion rule* $R \rightarrow_P S$ is valid if the replacement of the quantifier string R by the string S in any context does not diminish the class in question, i.e. $\mathsf{R}\mathsf{Q}\mathcal{P}^{\mu\tau(\mathsf{R})\tau(\mathsf{Q})} \subseteq \mathsf{S}\mathsf{Q}\mathcal{P}^{\mu\tau(\mathsf{S})\tau(\mathsf{Q})}$ for all $\mathsf{Q} \in \Gamma_p^*$ and $\mu \in \{0,1,2\}^*$. We say that the *equivalence rule* $\mathsf{R} \leftrightarrow_{\mathcal{P}} \mathsf{S}$ is valid if the replacement of the quantifier string R by the string S in any context does not change the class in question, i.e. $\mathsf{R}\mathsf{Q}\mathcal{P}^{\mu\tau(\mathsf{R})\tau(\mathsf{Q})} = \mathsf{S}\mathsf{Q}\mathcal{P}^{\mu\tau(\mathsf{S})\tau(\mathsf{Q})}$ for all $\mathsf{Q} \in \Gamma_p^*$ and $\mu \in \{0,1,2\}^*$. Obviously, we have $\mathsf{R} \leftrightarrow_{\mathcal{P}} \mathsf{S}$ if and only if $\mathsf{R} \rightarrow_{\mathcal{P}} \mathsf{S}$ and $\mathsf{S} \rightarrow_{\mathcal{P}} \mathsf{R}$.

For a rule $\mathsf{R} \rightarrow_{\mathcal{P}} \mathsf{S}$, we will also have to prove "its complement" $\overline{\mathsf{R}} \rightarrow_{\mathcal{P}} \overline{\mathsf{S}}$. However, the following proposition shows that only one of them has to be proved.

Proposition 3.3 (Complementation). *Let* $\mathsf{R}, \mathsf{S} \in \Gamma_p^*$. *If* $\mathsf{R} \rightarrow_{\mathcal{P}} \mathsf{S}$ *then* $\overline{\mathsf{R}} \rightarrow_{\mathcal{P}} \overline{\mathsf{S}}$.

Proof. Let $\mathsf{Q} \in \Gamma_p^*$ and $\mu \in \{0,1,2\}^*$. We conclude

$$\begin{aligned} L \in \overline{\mathsf{R}}\mathsf{Q}\mathcal{P}^{\mu\tau(\mathsf{R})\tau(\mathsf{Q})} &\Longrightarrow \overline{L} \in \mathrm{co}\overline{\mathsf{R}}\mathsf{Q}\mathcal{P}^{\mu\tau(\mathsf{R})\tau(\mathsf{Q})} \\ &\Longrightarrow \overline{L} \in \mathsf{R}\overline{\mathsf{Q}}\mathcal{P}^{\mu\tau(\mathsf{R})\tau(\mathsf{Q})} && \text{by Proposition 3.1} \\ &\Longrightarrow \overline{L} \in \mathsf{S}\overline{\mathsf{Q}}\mathcal{P}^{\mu\tau(\mathsf{S})\tau(\mathsf{Q})} && \text{by } \mathsf{R} \rightarrow_{\mathcal{P}} \mathsf{S} \\ &\Longrightarrow \overline{L} \in \mathrm{co}\overline{\mathsf{S}}\mathsf{Q}\mathcal{P}^{\mu\tau(\mathsf{S})\tau(\mathsf{Q})} && \text{by Proposition 3.1} \\ &\Longrightarrow L \in \overline{\mathsf{S}}\mathsf{Q}\mathcal{P}^{\mu\tau(\mathsf{S})\tau(\mathsf{Q})} \end{aligned}$$ ❑

The following rules show relations between the existential (universal, respectively) quantifiers of different types.

Lemma 3.4. *The following inclusion rules are valid:*

(1) $\varepsilon \rightarrow_{\mathcal{P}} \exists^p$ *and* $\varepsilon \rightarrow_{\mathcal{P}} \forall^p$;

(2) $\exists^p \rightarrow_{\mathcal{P}} \exists^1$ *and* $\forall^p \rightarrow_{\mathcal{P}} \forall^1$;

(3) $\exists^1 \rightarrow_{\mathcal{P}} \exists^2$ *and* $\forall^1 \rightarrow_{\mathcal{P}} \forall^2$.

Proof. Let $\mathsf{Q} \in \Gamma_p^*$ and $\mu \in \{0,1,2\}^*$.

(1) This is the classical case of introducing a dummy word quantifier.

(2) We prove the first rule, the other follows by complementation. For a language $L \in \exists^p\mathsf{Q}\mathcal{P}^{\mu 0\tau(\mathsf{Q})}$ there exist an $L_1 \in \mathcal{P}^{\mu 0\tau(\mathsf{Q})}$ and a polynomial p such that

$$\begin{aligned} X \in L &\Longleftrightarrow \exists^p u\, \mathsf{Q}Y\, (|u| = p(|X|) \wedge (X,u,Y) \in L_1) \\ &\Longleftrightarrow \exists^1 U\, \mathsf{Q}Y\, ((X,U,Y) \in L_2)\,, \end{aligned}$$

where $L_2 =_{\mathrm{df}} \{(X,U,Y) : (X, \langle U, p(|X|)\rangle, Y) \in L_1\}$. Let M be a polynomial-time machine of type $\mu 0\tau(\mathsf{Q})$ accepting L_1. Consider a machine M' of type $\mu 1\tau(\mathsf{Q})$ that on input (X,U,Y) computes $p(|X|)$ and then $\langle U, p(|X|)\rangle$ by asking $1, 11, \ldots, 1^{p(|X|)}$ to the oracle U. Then, M' works as M on input $(X, \langle U, p(|X|)\rangle, Y)$. Therefore, $L(M') = L_2$ and $L_2 \in \mathcal{P}^{\mu 1\tau(\mathsf{Q})}$, i.e. $L \in \exists^1\mathsf{Q}\mathcal{P}^{\mu 1\tau(\mathsf{Q})}$.

(3) This is obvious since a polynomial-time machine of type $\mu 1\tau(\mathsf{Q})$ can also be considered to be a machine of type $\mu 2\tau(\mathsf{Q})$. ❑

Next, we show an "equivalence rule" which is valid only in a special context. It says that a set quantifier is exactly as powerful as the corresponding word quantifier when applied to P.

Lemma 3.5. *Let* $\mu \in \{0,1,2\}^*$. *Then*

$$\exists^p P^{\mu 0} = \exists^1 P^{\mu 1} = \exists^2 P^{\mu 2} \quad \textit{and} \quad \forall^p P^{\mu 0} = \forall^1 P^{\mu 1} = \forall^2 P^{\mu 2}$$

Proof. We prove the first statement, the second follows by complementation. The inclusions "$\subseteq$" are valid by Lemma 3.4, thus only $\exists^2 P^{\mu 2} \subseteq \exists^p P^{\mu 0}$ has to be proved. By definition $L \in \exists^2 P^{\mu 2}$ if and only if there exists an $L' \in P^{\mu 2}$, such that $X \in L \iff \exists^2 U\,((X,U) \in L')$. Let M be a polynomial-time machine of type $\mu 2$ accepting L', which on input (X,U) queries $p(|X|)$ times the oracle U, where p is a polynomial. Without loss of generality we assume that M does not make a query twice. Let M' be a machine of type $\mu 0$ working on input (X,u) as M on input (X,U) with the following difference: Instead of the answer of U to the i-th query of M the machine M' uses the i-th bit of u. Now, $\exists^2 U\,((X,U) \in L') \iff \exists^p u\,(|u| = p(|X|) \wedge (X,u) \in L(M'))$ can be seen by the following construction:

"$u \to U$": Let $(X,u) \in L(M')$ for a word $u \in \{0,1\}^{p(|X|)}$ and define the set $U =_{\mathrm{df}} \{x : x \text{ is the } i\text{-th query of } M \wedge u(i) = 1\}$. Then $(X,U) \in L'$.

"$U \to u$": Let $(X,U) \in L'$ for a set $U \subseteq \{0,1\}^*$ and define the word $u \in \{0,1\}^{p(|X|)}$ as follows: $u(i) = 1 \iff_{\mathrm{df}}$ answer to the i-th query of M is "yes". Then $(X,u) \in L(M')$.

Hence, $L(M') \in P^{\mu 0}$ and $L \in \exists^p P^{\mu 0}$. ❑

The next result shows how to melt neighboured existential (universal, respectively) quantifiers.

Lemma 3.6. *For* $\sigma, \tau \in \{p,1,2\}$ *the following equivalence rules are valid*

$$\exists^\sigma \exists^\tau \leftrightarrow_P \exists^\rho \quad \textit{and} \quad \forall^\sigma \forall^\tau \leftrightarrow_P \forall^\rho$$

where $\rho = p$ *if* $\sigma = \tau = p$ *and* $\rho = \min\{\tau(\exists^\sigma) + \tau(\exists^\tau), 2\}$ *otherwise.*

Proof. We prove the first rule, the second follows by complementation. Using Lemma 3.4, it is easy to see that we have to prove only the inclusion rules

(1) $\exists^p \exists^p \to_P \exists^p$;

(2) $\exists^p \exists^1 \to_P \exists^1$ and $\exists^1 \exists^p \to_P \exists^1$;

(3) $\exists^2 \exists^2 \to_P \exists^2$;

(4) $\exists^2 \to_P \exists^1 \exists^1$.

In order to prove these inclusions, let $Q \in \Gamma_p^*$ and $\mu \in \{0,1,2\}^*$.

(1) Let $L \in \exists^p\exists^p Q\boldsymbol{P}^{\mu 00\tau(Q)}$. There exist an $L_1 \in \boldsymbol{P}^{\mu 00\tau(Q)}$ and polynomials p_1, p_2 such that

$$\begin{aligned} X \in L &\iff \exists^p u\, \exists^p v\, QY\, (|u| = p_1(|X|) \wedge |v| = p_2(|X|) \wedge (X, u, v, Y) \in L_1) \\ &\iff \exists^p w\, QY\, (|w| = 2p_1(|X|) + 2p_2(|X|) + 2 \wedge (X, w, Y) \in L_2)\,, \end{aligned}$$

where $L_2 =_{\text{df}} \{(X, \widehat{u}01\widehat{v}, Y) : (X, u, v, Y) \in L_1\}$. Let M be a polynomial-time machine of type $\mu 00\tau(Q)$ accepting L_1, and let M' be a machine of type $\mu 0\tau(Q)$ that on input (X, w, Y) computes u and v from $w = \widehat{u}01\widehat{v}$ (where it rejects if w does not have this form) and then works as M on input (X, u, v, Y). Therefore, $L(M') = L_2$ and $L_2 \in \boldsymbol{P}^{\mu 0\tau(Q)}$, i.e. $L \in \exists^p Q\boldsymbol{P}^{\mu 0\tau(Q)}$.

(2) It suffices to prove the first rule, because of the obvious rule $\exists^p\exists^1 \leftrightarrow_P \exists^1\exists^p$. For a language $L \in \exists^p\exists^1 Q\boldsymbol{P}^{\mu 01\tau(Q)}$ there exist an $L_1 \in \boldsymbol{P}^{\mu 01\tau(Q)}$ and a polynomial p such that

$$\begin{aligned} X \in L &\iff \exists^p v\, \exists^1 U\, QY\, (|v| = p(|X|) \wedge (X, v, U, Y) \in L_1) \\ &\iff \exists^1 W\, QY\, ((X, W, Y) \in L_2)\,, \end{aligned}$$

where $L_2 =_{\text{df}} \left\{(X, W, Y) : \left(X, \langle W, p(|X|)\rangle, 1^{p(|X|)+1}\backslash W, Y\right) \in L_1\right\}$. Let M be a polynomial-time machine of type $\mu 01\tau(Q)$ accepting L_1, and let M' be a machine of type $\mu 1\tau(Q)$ that on input (X, W, Y) computes $p(|X|)$ and then $\langle W, p(|X|)\rangle$ by asking $1, 11, \ldots, 1^{p(|X|)}$ to W. Then, the machine M' works like machine M on input $\left(X, \langle W, p(|X|)\rangle, 1^{p(|X|)+1}\backslash W, Y\right)$ with the difference that instead of asking the query u to oracle $1^{p(|X|)+1}\backslash W$, the query $1^{p(|X|)+1}u$ is asked to W. Therefore, $L(M') = L_2$ and $L_2 \in \boldsymbol{P}^{\mu 1\tau(Q)}$, i.e. $L \in \exists^1 Q\boldsymbol{P}^{\mu 1\tau(Q)}$.

(3) Let $L \in \exists^2\exists^2 Q\boldsymbol{P}^{\mu 22\tau(Q)}$. There exists an $L_1 \in \boldsymbol{P}^{\mu 22\tau(Q)}$ such that

$$\begin{aligned} X \in L &\iff \exists^2 U\, \exists^2 V\, QY\, ((X, U, V, Y) \in L_1) \\ &\iff \exists^2 W\, QY\, ((X, W, Y) \in L_2)\,, \end{aligned}$$

where $L_2 =_{\text{df}} \{(X, W, Y) : (X, 0\backslash W, 1\backslash W, Y) \in L_1\}$. Let M be a polynomial-time machine of type $\mu 22\tau(Q)$ accepting L_1. Consider a machine M' of type $\mu 2\tau(Q)$ working on input (X, W, Y) as M on input $(X, 0\backslash W, 1\backslash W, Y)$ with the difference that instead of asking the query w to oracle $0\backslash W$ (oracle $1\backslash W$), the query $0w$ ($1w$, respectively) to oracle W is asked. Therefore, $L(M') = L_2$ and $L_2 \in \boldsymbol{P}^{\mu 2\tau(Q)}$, i.e. $L \in \exists^2 Q\boldsymbol{P}^{\mu 2\tau(Q)}$.

(4) Let us first prove the validity of the new rule $\exists^2 \rightarrow_P \exists^1\exists^1\forall^p$. For a language $L \in \exists^2 Q\boldsymbol{P}^{\mu 2\tau(Q)}$ there exists an $L_1 \in \boldsymbol{P}^{\mu 2\tau(Q)}$ such that for a suitable polynomial p

$$
\begin{aligned}
X \in L \iff & \exists^2 U\, QY\, ((X,U,Y) \in L_1) \\
\iff & \exists^2 U\, \exists^1 W \Big(\forall^p u \forall^p v \Big(|u|, |v| \le p(|X|) \to (u \in U \leftrightarrow v01\hat{u} \in W) \Big) \\
& \qquad \wedge QY \Big((X,U,Y) \in L(M) \Big) \Big)
\end{aligned}
$$

$\big($take $W = \{v01\hat{u} : u, v \in \{0,1\}^* \wedge u \in U\}$ for example, and let M be a polynomial-time machine of type $\mu 2\tau(Q)$ accepting $L_1\big)$

$$
\begin{aligned}
\iff & \exists^1 U\, \exists^1 W \forall^p u \forall^p v\, QY \Big(|u|, |v| \le p(|X|) \to \Big((u \in U \leftrightarrow v01\hat{u} \in W) \\
& \qquad \wedge (X, W, Y) \in L(M') \Big) \Big)
\end{aligned}
$$

$\big(M'$ works on input (X,W,Y) as M on input (X,U,Y) but instead of asking u to U after queries $u_1, u_2, \dots, u_m$ it asks the query $01\hat{u}_1 01\hat{u}_2 01 \dots 01\hat{u}_m 01\hat{u}$ to W. Note that M' asks W in a type 1 manner, i.e. it is a polynomial-time machine of type $\mu 1\tau(Q)\big)$

$$
\begin{aligned}
\iff & \exists^1 U\, \exists^1 W \forall^p u \forall^p v\, QY \Big(|u|, |v| \le p(|X|) \to \\
& \qquad (X, U, W, u, v, Y) \in L(M_L) \Big)
\end{aligned}
$$

$\big(M_L$ on input (X,U,W,u,v,Y) first asks $u \in U$ and $v01\hat{u} \in W$. If the answers do not coincide, then M_L rejects. Otherwise, M_L simulates M' on input (X,W,Y) but instead of asking w to W it asks $v01\hat{u}w$ to W. Note that M_L asks U and W in a type 1 manner, i.e. it is a polynomial-time machine of type $\mu 1100\tau(Q)\big)$

This shows $L \in \exists^1\exists^1\forall^p\forall^p Q P^{\mu 1100\tau(Q)}$, i.e. $\exists^2 \to_P \exists^1\exists^1\forall^p\forall^p$. Now, applying the rule $\exists^p\exists^p \to_P \exists^p$ (Statement (1)) we get the desire rule $\exists^2 \to_P \exists^1\exists^1\forall^p$.

Next, we prove the rule $\exists^2 \to_P \exists^1\exists^1$. For $k \ge 0$, $Q_1, \dots, Q_k \in \{\exists, \forall\}$ and $\tau_1, \dots, \tau_k \in \{p, 1, 2\}$, consider the class $\exists^2 Q_1^{\tau_1} \dots Q_k^{\tau_k} P^{\mu 2\tau\left(Q_1^{\tau_1} \dots Q_k^{\tau_k}\right)}$.

Case 1. If $Q_1 = \cdots = Q_k = \exists$ we can conclude by Lemmas 3.4 and 3.5, and Statement (3)

$$
\exists^2\exists^{\tau_1} \dots \exists^{\tau_k} P^{\mu 2\tau\left(\exists_1^{\tau_1} \dots \exists_k^{\tau_k}\right)} \subseteq \exists^2 P^{\mu 2} \subseteq \exists^p P^{\mu 0} \subseteq \exists^1\exists^1\exists^{\tau_1} \dots \exists^{\tau_k} P^{\mu 11\tau\left(\exists_1^{\tau_1} \dots \exists_k^{\tau_k}\right)}
$$

Case 2. If there exists an $l \in \{1, \dots, k\}$ such that $Q_1 = \cdots = Q_{l-1} = \exists$ and

$Q_l = \forall$, then we can conclude

$$\begin{aligned}
&\exists^2\exists^{\tau_1}\dots\exists^{\tau_{l-1}}\forall^{\tau_l}Q_{l+1}^{\tau_{l+1}}\dots Q_k^{\tau_k}\mathcal{P}^{\mu 2\tau(Q_1^{\tau_1}\dots Q_k^{\tau_k})} && \\
&\quad\subseteq \exists^2\forall^{\tau_l}Q_{l+1}^{\tau_{l+1}}\dots Q_k^{\tau_k}\mathcal{P}^{\mu 2\tau(Q_l^{\tau_l}\dots Q_k^{\tau_k})} && \text{by Lemma 3.4 and (3)}\\
&\quad\subseteq \exists^1\exists^1\forall^p\forall^{\tau_l}Q_{l+1}^{\tau_{l+1}}\dots Q_k^{\tau_k}\mathcal{P}^{\mu 110\tau(Q_l^{\tau_l}\dots Q_k^{\tau_k})} && \text{by } \exists^2 \rightarrow_{\mathcal{P}} \exists^1\exists^1\forall^p\\
&\quad\subseteq \exists^1\exists^1\forall^{\tau_l}Q_{l+1}^{\tau_{l+1}}\dots Q_k^{\tau_k}\mathcal{P}^{\mu 11\tau(Q_l^{\tau_l}\dots Q_k^{\tau_k})} && \text{by (1), (2)}\\
& && \text{or Lemma 3.4 and (3)}\\
&\quad\subseteq \exists^1\exists^1\exists^{\tau_1}\dots\exists^{\tau_{l-1}}\forall^{\tau_l}Q_{l+1}^{\tau_{l+1}}\dots Q_k^{\tau_k}\mathcal{P}^{\mu 11\tau(Q_1^{\tau_1}\dots Q_k^{\tau_k})} && \text{by Lemma 3.4}
\end{aligned}$$

This completes the proof. ❑

Our last rules show that word quantifiers can be eliminated if a set quantifier and at least one more quantifier follow.

Lemma 3.7. *For $\sigma \in \{1,2\}$ and $\tau \in \{p,1,2\}$ the equivalence rules*

$$\exists^p\forall^\sigma\exists^\tau \leftrightarrow_{\mathcal{P}} \forall^\sigma\exists^\tau \quad \textit{and} \quad \forall^p\exists^\sigma\forall^\tau \leftrightarrow_{\mathcal{P}} \exists^\sigma\forall^\tau$$

are valid.

Proof. We prove the second rule, the first follows by complementation. Let $Q \in \Gamma_p^*$ and $\mu \in \{0,1,2\}^*$.

Let us first prove the validity of the inclusion rule $\forall^p\exists^\sigma \rightarrow_{\mathcal{P}} \exists^\sigma\forall^p$ ($\sigma \in \{1,2\}$). For a language $L \in \forall^p\exists^\sigma Q\mathcal{P}^{\mu 0\sigma\tau(Q)}$ there exist an $L_1 \in \mathcal{P}^{\mu 0\sigma\tau(Q)}$ and a polynomial p such that

$$X \in L \Longleftrightarrow \forall^p u\, \exists^\sigma V\, QY\, (|u| = p(|X|) \rightarrow (X,u,V,Y) \in L_1)$$

Now, we define $L_2 =_{\mathrm{df}} \{(X,W,u,Y) : (X,u,u\backslash W,Y) \in L_1\}$ and we prove

$$X \in L \Longleftrightarrow \exists^\sigma W\, \forall^p u\, QY\, (|u| = p(|X|) \rightarrow (X,W,u,Y) \in L_2)$$

"$\Longrightarrow$": For every $u \in \{0,1\}^{p(|X|)}$ let V_u be a set such that $(X,u,V_u,Y) \in L_1$ and define $W = \bigcup_{|u|=p(|X|)}\{uw : w \in V_u\}$. Then $(X,W,u,Y) \in L_2$ for every $u \in \{0,1\}^{p(|X|)}$.

"$\Longleftarrow$": Let $(X,W,u,Y) \in L_2$ for every $u \in \{0,1\}^{p(|X|)}$. Hence, for each $u \in \{0,1\}^{p(|X|)}$ there exists a set $V_u =_{\mathrm{df}} u\backslash W$ such that $(X,u,V_u,Y) \in L_1$.

Let M be a polynomial-time machine of type $\mu 0\sigma\tau(Q)$ accepting L_1, and let M' be a machine of type $\mu\sigma 0\tau(Q)$ working on input (X,W,u,Y) as M on input $(X,u,u\backslash W,Y)$ with the difference that instead of asking the query w to oracle $u\backslash W$ the query uw is asked to W. Since M asks $u\backslash W$ in a type σ manner, the machine M' does it as well. Therefore, $L(M') = L_2$ and $L_2 \in \mathcal{P}^{\mu\sigma 0\tau(Q)}$, i.e. $L \in \exists^\sigma\forall^p Q\mathcal{P}^{\mu\sigma 0\tau(Q)}$.

Now, using this rule and Lemma 3.6 we can conclude the rule $\forall^p\exists^\sigma\forall^\tau \rightarrow_{\mathcal{P}} \exists^\sigma\forall^\tau$. The rule $\exists^\sigma\forall^\tau \rightarrow_{\mathcal{P}} \forall^p\exists^\sigma\forall^\tau$ is valid by Lemma 3.4. ❑

3.2.2 A Normal Form Theorem

Using the equivalence rules presented in §3.2.1 we can state the fact that every class of the analytic polynomial-time hierarchy can be represented in a certain normal form. For $l \geq 1$, let $Q_l =_{\text{df}} \exists$ if l is odd and $Q_l =_{\text{df}} \forall$ otherwise. For $k, m \geq 0$ and $\tau_1, \tau_2, \tau_3, \ldots, \tau_k \in \{1, 2\}$ we define

$$\mathcal{K}_P(\tau_1\tau_2\tau_3 \ldots \tau_k, m) =_{\text{df}} \exists^{\tau_1}\forall^{\tau_2}\exists^{\tau_3} \ldots Q_k^{\tau_k} Q_{k+1}^{p} Q_{k+2}^{p} \ldots Q_{k+m}^{p} P$$

Theorem 3.8 (Normal Form Theorem). *Every class of the analytic polynomial-time hierarchy coincides with one of the classes P, $\mathcal{K}_P(\tau, m)$ or co$\mathcal{K}_P(\tau, m)$, where $\tau \in \{1, 2\}^*$ and $m \geq 1$.*

Proof. Consider an arbitrary class of the analytic polynomial-time hierarchy. If this class is defined without quantifiers, then it is *P*. Otherwise it coincides with one of the classes $\mathcal{K}_P(\tau, m)$ or co$\mathcal{K}_P(\tau, m)$ with $\tau \in \{1, 2\}^*$ and $m \geq 1$. This can be seen as follows: By Lemma 3.6, we bring the quantifier prefix in a form, where no quantifier substring $\exists^{\tau_1}\exists^{\tau_2}$ or $\forall^{\tau_1}\forall^{\tau_2}$ appears ($\tau_1, \tau_2 \in \{p, 1, 2\}$). By Lemma 3.5, we ensure that the last quantifier is a word quantifier. By Lemma 3.7, we eliminate all word quantifiers which are not followed by a word quantifier. This last step can generate quantifier substrings $\exists^{\tau_1}\exists^{\tau_2}$ or $\forall^{\tau_1}\forall^{\tau_2}$ ($\tau_1 \in \{p, 1, 2\}$ and $\tau_2 \in \{1, 2\}$). However, applying repeatedly the rules of Lemmas 3.6 and 3.7 we get the desired result. ❑

3.3 Characterizing the Classes $\mathcal{K}_P(\cdot)$ and co$\mathcal{K}_P(\cdot)$

In this section, we characterize the classes having the form $\mathcal{K}_P(\tau, m)$ or co$\mathcal{K}_P(\tau, m)$ by well-known complexity classes, where $\tau \in \{1, 2\}^*$ and $m \geq 1$. It is well-known that such a class built without set quantifiers coincides with a class of the (arithmetic) polynomial-time hierarchy and vice versa, i.e. $\mathcal{K}_P(\varepsilon, m) = \Sigma_m^p$ and co$\mathcal{K}_P(\varepsilon, m) = \Pi_m^p$ for $m \geq 0$ (see §3.1.2). Thus, it remains to consider the classes involving set quantifiers. The simplest classes of these types are those containing only one set quantifier of type 1 and one word quantifier, i.e. $\mathcal{K}_P(1, 1) = \exists^1\forall^p P$ and co$\mathcal{K}_P(1, 1) = \forall^1\exists^p P$, which turn out to coincide with *PSPACE*.

Theorem 3.9. $\exists^1\forall^p P = \forall^1\exists^p P =$ *PSPACE.*

Proof. Since *PSPACE* is closed under complementation, only $\exists^1\forall^p P =$ *PSPACE* has to be proved.

"*PSPACE* $\subseteq \exists^1\forall^p P$": For a language $L \in$ *PSPACE* let M be a polynomial-time alternating machine accepting L whose computation trees are binary. The machine M accepts an instance x if there exists an accepting subtree of their computation tree $\beta_M(x)$.

Since a good subtree S of $\beta_M(x)$ includes both successors of an universal configuration and exactly one successor of an existential configuration (see p. 9), we can describe S by the set U_S of all words z corresponding to an existential configuration whose right successor $z1$ belongs to S in the following way: $z \in U_S$ means "$z1$ is in S" and $z \notin U_S$

means "$z0$ is in S" (see Figure 3.1). Note that every set U describes a good subtree S_U of $\beta_M(x)$ in this way. Now we can conclude

$$\begin{aligned}
x \in L &\iff \exists S(S \text{ is an accepting subtree of } \beta_M(x))\\
&\iff \exists U(S_U \text{ is an accepting subtree of } \beta_M(x))\\
&\iff \exists U \forall \pi((\pi \text{ is a path in } S_U \wedge \pi \text{ is a path in } \beta_M(x)) \longrightarrow \pi \text{ accepting})\\
&\iff \exists U\, \forall \pi\Big(\Big(\bigwedge_{i=0}^{|\pi|-1} (\pi(1)\dots\pi(i) \text{ is existential} \rightarrow \chi_U(\pi(1)\dots\pi(i)) = \pi(i+1))\\
&\qquad\qquad \wedge\, \pi \text{ is a path in } \beta_M(x)\Big) \longrightarrow \pi \text{ accepting}\Big)
\end{aligned}$$

For fixed U and π, the condition in parenthesis can be checked in polynomial-time with queries to U which are initial parts of the word π. This is a type 1 querying. Therefore, $L \in \exists^1\forall^p P$.

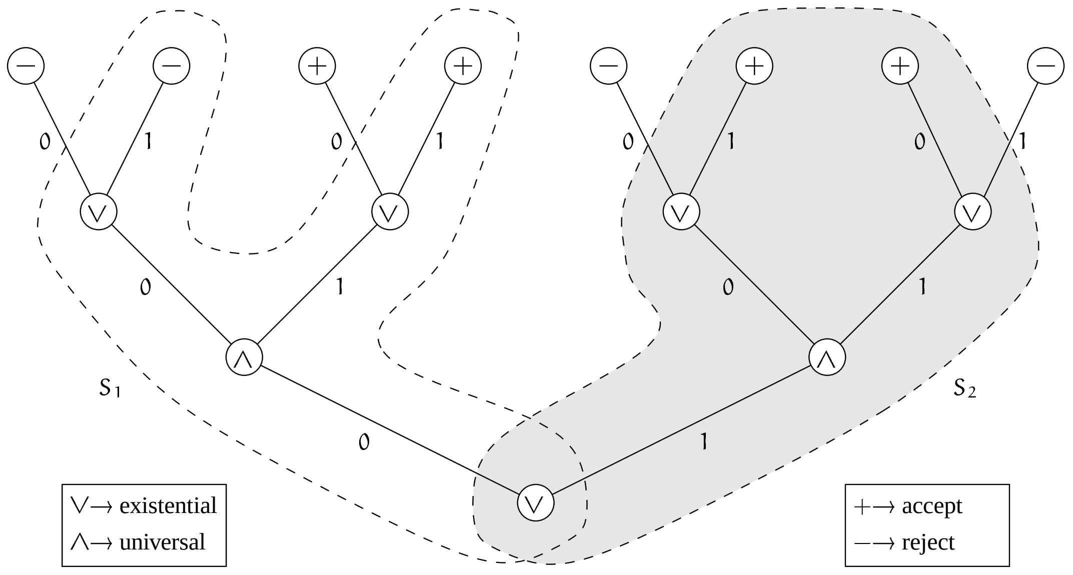


Figure 3.1: *Good subtrees S_1 and S_2 with S_2 being also an accepting subtree, where* $U_{S_1} = \{00, 01\}$ *and* $U_{S_2} = \{\varepsilon, 10\}$.

"$\exists^1\forall^p P \subseteq PSPACE$": Let $L \in \exists^1\forall^p P$. There exist an $L_1 \in P^{010}$ and a polynomial p such that $x \in L \iff \exists^1 U\, \forall^p u\, (|u| = p(|x|) \rightarrow (x, U, u) \in L_1)$. Since deterministic polynomial-space is as powerful as nondeterministic polynomial-space, it suffices to describe a nondeterministic polynomial-space machine accepting L (the nondeterminism will be used to guess the oracle U). Let M' be a machine of type 010 accepting L_1 with time bound q where q is a polynomial. Thus, no query of M' on input (x, U, u) to oracle U is longer than $q(|x|)$. It is important that M' asks for a given input (x, U, u) only queries from one *oracle path* $w_1, w_1w_2, \dots, w_1w_2\dots w_{q(|x|)}$. Let M be a machine that considers step by step all these oracle paths in lexicographical order. For each oracle path $\pi \in \{0, 1\}^{q(|x|)}$:

1. The machine M guesses the answers of the oracle U to the queries on this particular oracle path π. This has to be made in accordance with the guesses for the previous path, i.e. the answers to the queries of the common initial part of π and its predecessor (in lexicographical order) have to be the same. Let π_g be this path with the guessed answers. Note that only this part of U has to be stored which belongs to the current oracle path π.

2. Now M simulates M' for all inputs (x, u) asking only queries from the oracle path π and uses the oracle answers encoded in π_g. In fact, M simulates M' for all inputs (x, u) and stops such a simulation if a query is asked which is not from the oracle path π.

Finally, the machine M accepts if and only if all simulations in item 2, which are not stopped, end accepting. Therefore, $L(M) = L$ and M uses polynomial-space, i.e. $L \in$ *PSPACE.* ❑

The following lemma is a special presentation of the well-known fact that the 3SAT problem is *DLOGTIME*-complete for *NP*. For $a \in \{0, 1\}$ let $a^1 =_{\text{df}} a$ and $a^0 =_{\text{df}} 1 - a$.

Lemma 3.10. *A language* L *is in NP if and only if there exist polynomials* p, q *and functions* $f, g \in$ *DLOGTIME, i.e. each bit of the value of* f *and* g *can be computed in logarithmic time, such that* $f(1^n, i, j) \in \{1, 2, \ldots, p(n)\}$, $g(1^n, i, j) \in \{0, 1\}$ *and*

$$x \in L \iff \exists a_1 a_2 \ldots a_{p(|x|)} \left(\bigwedge_{i=1}^{|x|} (a_i = x(i)) \wedge \bigwedge_{i=1}^{p(|x|)} \bigvee_{j=1}^{3} a_{f(1^{|x|},i,j)}^{g(1^{|x|},i,j)} \right)$$

where $a_1, a_2, \ldots, a_{p(|x|)} \in \{0, 1\}$.

The remainder classes $\mathcal{K}_P(\cdot)$ and co$\mathcal{K}_P(\cdot)$ that we have not already characterized turns out to coincide with classes of the exponential-time alternation hierarchy. The next theorem shows which classes $\mathcal{K}_P(\cdot)$ and co$\mathcal{K}_P(\cdot)$ contain a level of the exponential-time alternation hierarchy.

Theorem 3.11. *For* $k \geq 1$ *the inclusions*

$$\Sigma_k^{\text{exp}} \subseteq \mathcal{K}_P(1^k, 2) \cap \mathcal{K}_P(1^{k-1}2, 1) \quad \textit{and} \quad \Pi_k^{\text{exp}} \subseteq \text{co}\mathcal{K}_P(1^k, 2) \cap \text{co}\mathcal{K}_P(1^{k-1}2, 1)$$

are valid.

Proof. We prove the first inclusion, the second follows by complementation. Consider the case that k is odd. Let $L \in \Sigma_k^{\text{exp}}$, i.e. let L be a language which is accepted by an exponential-time alternating Turing machine starting with an existential state and having at most $k - 1$ alternations on every computation path. Hence, there exist a language $L_1 \in NP$ and a polynomial r such that

$$x \in L \iff \exists u_1 \forall u_2 \exists u_3 \ldots \forall u_{k-1} \left(\left(x10^{2^{|x|}}, u_1, u_2, \ldots, u_{k-1} \right) \in L_1 \right)$$

where the quantifiers vary over words of length $2^{r(|x|)}$.

Let $n =_{df} |x| + 1 + 2^{|x|}$ and $N =_{df} n + (k-1) \cdot 2^{r(|x|)}$. By Lemma 3.10, there exist polynomials p, q and functions $f, g \in$ *DLOGTIME* such that

$$x \in L \Leftrightarrow \exists u_1 \forall u_2 \exists u_3 \dots \forall u_{k-1} \exists a_1 a_2 \dots a_{p(N)} \Bigg(\bigwedge_{i=1}^{|x|} (a_i = x(i)) \wedge (a_{|x|+1} = 1) \wedge \bigwedge_{i=|x|+2}^{n} (a_i = 0) \wedge \bigwedge_{j=1}^{k-1} \bigwedge_{i=1}^{2^{r(|x|)}} \left(a_{n+(j-1) 2^{r(|x|)} + i} = u_j(i)\right) \wedge \bigwedge_{i=1}^{q(N)} \bigvee_{j=1}^{3} a^{g(1^N,i,j)}_{f(1^N,i,j)} \Bigg)$$

where the first $k-1$ quantifiers vary over words of length $2^{r(|x|)}$. Obviously, the functions $f'(x,i,j) =_{df} f(1^N,i,j)$ and $g'(x,i,j) =_{df} g(1^N,i,j)$ are polynomial-time computable (the length of f' is polynomial in $|x|$ bounded and the length of g' is constant, and we need polynomial-time to compute each bit of these functions). Now, the idea is to represent a word w of length $2^{r(|x|)}$ by the oracle $W_w =_{df} \{\text{lex}(i) : w(i) = 1\}$. There exists a polynomial s such that

$$\begin{aligned} x \in L \Leftrightarrow\ & \exists U_1 \forall U_2 \dots \forall U_{k-1}\ \exists U_k \Bigg(\bigwedge_{i=1}^{|x|} (i \in U_k \leftrightarrow x(i) = 1) \wedge ((|x|+1) \in U_k) \\ & \wedge \bigwedge_{i=|x|+2}^{n} (i \notin U_k) \\ & \wedge \bigwedge_{j=1}^{k-1} \bigwedge_{i=1}^{2^{r(|x|)}} \left((n + (j-1) \cdot 2^{r(|x|)} + i) \in U_k \leftrightarrow i \in U_j\right) \\ & \wedge \bigwedge_{i=1}^{2^{s(|x|)}} \bigvee_{j=1}^{3} (f'(x,i,j) \in U_k \leftrightarrow g'(x,i,j) = 1) \Bigg) \\ \Leftrightarrow\ & \exists^1 U_1 \forall^1 U_2 \dots \forall^1 U_{k-1} \exists^1 U_k \forall^p i \Bigg(1 \le i \le N + 2^{s(|x|)} \rightarrow \\ & \exists^p j\, (1 \le j \le 3 \wedge (x, U_1, U_2, \dots, U_{k-1}, U_k, i, j) \in L_2) \Bigg) \end{aligned}$$

where $(x, U_1, U_2, \dots, U_{k-1}, U_k, i, j) \in L_2 \Longleftrightarrow_{df}$

$$\begin{aligned} & (1 \le i \le |x| \rightarrow (i \in U_k \leftrightarrow x(i) = 1)) \wedge ((i = |x|+1) \rightarrow i \in U_k) \\ & \wedge (|x| + 1 < i \le n \rightarrow i \notin U_k) \\ & \wedge (n < i \le n + 2^{r(|x|)} \rightarrow (i \in U_k \leftrightarrow i - n \in U_1)) \\ & \wedge (n + 2^{r(|x|)} < i \le n + 2 \cdot 2^{r(|x|)} \rightarrow (i \in U_k \leftrightarrow i - n - 2^{r(|x|)} \in U_2)) \\ & \quad \vdots \\ & \wedge (n + (k-2) \cdot 2^{r(|x|)} < i \le N \rightarrow (i \in U_k \leftrightarrow i - n - (k-2) \cdot 2^{r(|x|)} \in U_{k-1})) \\ & \wedge (N < i \rightarrow (f'(x, i-N, j) \in U_k \leftrightarrow g'(x, i-N, j) = 1)) \qquad (\star) \end{aligned}$$

It is obvious that L_2 can be accepted by a deterministic polynomial-time oracle machine which, on input $(x, U_1, \dots, U_{k-1}, U_k, i, j)$, queries each of the oracles $U_1, \dots, U_{k-1}$, U_k at most once. Hence, $L_2 \in P^{01^k00}$ and $L \in \mathcal{K}_P(1^k, 2)$.

Alternatively, since the quantifier $\exists^p j$ ranges only over the set $\{1, 2, 3\}$, it can be eliminated:

$$x \in L \Leftrightarrow \exists^1 U_1 \forall^1 U_2 \dots \forall^1 U_{k-1} \exists^2 U_k \forall^p i \Big(1 \le i \le N + 2^{s(|x|)} \to \\ (x, U_1, U_2, \dots, U_{k-1}, U_k, i) \in L_3\Big)$$

where L_3 is defined as L_2 but the last line $(\star)$ has to be modified to

$$\wedge \Big(N < i \to \bigvee_{j=1}^{3} (f'(x, i - N, j) \in U_k \leftrightarrow g'(x, i - N, j) = 1)\Big).$$

Obviously, a deterministic polynomial-time oracle machine can accept L_3 in such a way that on input $(x, U_1, U_2, \dots, U_{k-1}, U_k, i)$ the oracles $U_1, U_2, \dots, U_{k-1}$ are queried at most once and U_k at most three times. Hence, $L_3 \in P^{01^{k-1}20}$ and $L \in \mathcal{K}_P(1^{k-1}2, 1)$.

In the case that k is even, $\Pi_k^{\text{exp}} \subseteq \text{co}\mathcal{K}_P(1^k, 2) \cap \text{co}\mathcal{K}_P(1^{k-1}2, 1)$ can be proved in the same way which yields the desired result by complementation. ❑

The following proposition shows which classes having an alternating sequence of existential and universal quantifiers of type 2 on the base of the class *PSPACE* coincide with a level of the exponential-time alternation hierarchy.

Proposition 3.12. *For $l \ge 1$, let $Q_l =_{\text{df}} \exists$ if l is odd and $Q_l =_{\text{df}} \forall$ otherwise. Then, for $k \ge 1$*

$$\exists^2\forall^2\exists^2 \dots Q_k^2 \mathit{PSPACE}^{02^k} = \Sigma_k^{\text{exp}} \quad \textit{and} \quad \forall^2\exists^2\forall^2 \dots \overline{Q}_k^2 \mathit{PSPACE}^{02^k} = \Pi_k^{\text{exp}}$$

Proof. We prove the first equality, the second follows by complementation. For the inclusion $\exists^2\forall^2\exists^2 \dots Q_k^2 \mathit{PSPACE}^{02^k} \subseteq \Sigma_k^{\text{exp}}$, the idea is to replace the relevant part of an oracle U (only polynomially length-bounded words are asked) by the exponentially length-bounded word w_U with $w_U(i) = 1 \leftrightarrow \text{lex}(i) \in U$. Hence, a language from $\exists^2\forall^2 \dots Q_k^2 \mathit{PSPACE}^{02^k}$ can be accepted by an alternating exponential-time machine with $k - 1$ alternations which starts with an existential state, i.e. $\exists^2\forall^2 \dots Q_k^2 \mathit{PSPACE}^{02^k} \subseteq \Sigma_k^{\text{exp}}$. For the other direction, by Theorem 3.11 and Lemma 3.4 follows $\Sigma_k^{\text{exp}} \subseteq \mathcal{K}_P(2^k, 1)$ which is included in $\exists^2\forall^2\exists^2 \dots Q_k^2 \mathit{PSPACE}^{02^k}$, since the last quantifier, which is a word quantifier, can be easily simulated by a *PSPACE*-computation. ❑

The next theorem shows which classes $\mathcal{K}_P(\cdot)$ and co$\mathcal{K}_P(\cdot)$ are included in a level of the exponential-time alternation hierarchy.

Theorem 3.13. *For $k, m \ge 1$ the inclusions*

$$\mathcal{K}_P(2^k, m) \cup \mathcal{K}_P(2^k1, 1) \subseteq \Sigma_k^{\text{exp}} \quad \textit{and} \quad \text{co}\mathcal{K}_P(2^k, m) \cup \text{co}\mathcal{K}_P(2^k1, 1) \subseteq \Pi_k^{\text{exp}}$$

are valid.

Proof. We prove the first inclusion, the second follows by complementation. Let $Q_k =_{df} \exists$ if k is odd and $Q_k =_{df} \forall$ otherwise.

"$\mathcal{K}_P(2^k, m) \subseteq \Sigma_k^{exp}$": The inclusion $\mathcal{K}_P(2^k, m) \subseteq \exists^2\forall^2 \ldots Q_k^2 PSPACE^{02^k}$ is evident, since the last m quantifiers, which are word quantifiers, can be easily simulated by a *PSPACE*-computation. Now, using Proposition 3.12 we get the desire result.

"$\mathcal{K}_P(2^k 1, 1) \subseteq \Sigma_k^{exp}$": Let k be even. The proof of $\exists^1\forall^p P \subseteq PSPACE$ (Theorem 3.9) remains valid if the machines have additionally k oracles of type 2, i.e. $\exists^1\forall^p P^{02^k 10} \subseteq PSPACE^{02^k}$, and hence $\exists^2\forall^2 \ldots \forall^2\exists^1\forall^p P \subseteq \exists^2\forall^2 \ldots \forall^2 PSPACE^{02^k}$ which is included in Σ_k^{exp} by Proposition 3.12. The case of odd k is treated analogously on the base of $\forall^1\exists^p P \subseteq PSPACE$. ❑

Combining the above results we can state a complete characterization of the classes of the analytic polynomial-time hierarchy having the form $\mathcal{K}_P(\tau, m)$ or co$\mathcal{K}_P(\tau, m)$ for $\tau \in \{1, 2\}^*$ and $m \geq 1$.

Theorem 3.14. *Let* $k \geq 0$, $m \geq 1$ *and* $\tau_1, \ldots, \tau_k \in \{1, 2\}$. *Then*

$$\mathcal{K}_P(\tau_1 \ldots \tau_k, m) = \begin{cases} \Sigma_m^p & \text{if } k = 0, \\ PSPACE & \text{if } k = 1, \tau_k = 1 \text{ and } m = 1, \\ \Sigma_{k-1}^{exp} & \text{if } k \geq 2, \tau_k = 1 \text{ and } m = 1, \\ \Sigma_k^{exp} & \text{if } k \geq 1, \tau_k = 2 \text{ and } m = 1, \\ \Sigma_k^{exp} & \text{if } k \geq 1 \text{ and } m \geq 2. \end{cases}$$

Proof. The first line is obvious [SM73, Sto77, Wra77]. The second line is valid by Theorem 3.9, and the remaining lines by Lemma 3.4 and Theorems 3.11 and 3.13. ❑

3.4 Characterizing by Well-Known Complexity Classes and an Algorithm

We start this section giving a complete characterization of all classes of the analytic polynomial-time hierarchy.

Theorem 3.15. *Each class of the analytic polynomial-time hierarchy APH coincides with one of the classes* Σ_k^p, Π_k^p *(*$k \geq 0$*), PSPACE,* Σ_k^{exp} *or* Π_k^{exp} *(*$k \geq 1$*) and vice versa.*

Proof. By Theorem 3.8, each class of the analytic polynomial-time hierarchy coincides either with *P* or with a class in normal form. Thus, we need only to prove that for $\tau \in \{1, 2\}^*$ and $m \geq 1$, the class $\mathcal{K}_P(\tau, m)$ coincides with one of the mentioned classes, since co$\mathcal{K}_P(\tau, m)$ can be treated by complementation and $P = \Sigma_0^p = \Pi_0^p$. However, this is exactly Theorem 3.14. ❑

Next, an algorithm which solves the problem investigated in this chapter is presented. It summarizes the proofs of Theorems 3.15, 3.8 and 3.14.

Given: A class $\mathsf{Q}\mathit{P}$ of the analytic polynomial-time hierarchy where $\mathsf{Q} \in \Gamma_p^*$.
Question: With which well-known complexity class $\mathsf{Q}\mathit{P}$ coincides?

Algorithm:
If $\mathsf{Q} = \varepsilon$ then $\mathsf{Q}P = P$. Otherwise repeat the following steps until the form $\mathcal{K} = \mathcal{K}_P(\tau_1 \dots \tau_k, m)$ or $\mathcal{K} = \text{co}\mathcal{K}_P(\tau_1 \dots \tau_k, m)$ for some $k \geq 0$, $m \geq 1$ and $\tau_1, \dots, \tau_k \in \{1, 2\}$ is established:

1. Eliminate all substrings $\exists^\sigma\exists^\tau$ and $\forall^\sigma\forall^\tau$ using the equivalence rules $\exists^\sigma\exists^\tau \leftrightarrow_P \exists^\rho$ and $\forall^\sigma\forall^\tau \leftrightarrow_P \forall^\rho$ where $\rho = p$ if $\sigma = \tau = p$ and $\rho = \min\{\tau(\exists^\sigma) + \tau(\exists^\tau), 2\}$ otherwise (Lemma 3.6).

2. Replace the rightmost quantifier O^τ with O^p (Lemma 3.5).

3. For $\sigma \in \{1, 2\}$ and $\tau \in \{p, 1, 2\}$, eliminate all substrings $\exists^p\forall^\sigma\exists^\tau$ and $\forall^p\exists^\sigma\forall^\tau$ using the equivalence rules $\exists^p\forall^\sigma\exists^\tau \leftrightarrow_P \forall^\sigma\exists^\tau$ and $\forall^p\exists^\sigma\forall^\tau \leftrightarrow_P \exists^\sigma\forall^\tau$ (Lemma 3.7).

Now get (Theorem 3.14)

$$\mathcal{K}_P(\tau_1 \dots \tau_k, m) = \begin{cases} \Sigma_m^p & \text{if } k = 0, \\ \mathit{PSPACE} & \text{if } k = 1,\, \tau_k = 1 \text{ and } m = 1, \\ \Sigma_{k-1}^{\exp} & \text{if } k \geq 2,\, \tau_k = 1 \text{ and } m = 1, \\ \Sigma_k^{\exp} & \text{otherwise.} \end{cases}$$

3.5 Conclusions

In §3.4 we gave a complete characterization of all classes of the analytic polynomial-time hierarchy. An algorithm was also presented which allows to find out the corresponding well-known class in an easy way. These results tighten up Orponen's [Orp83] characterization $\Sigma_k^{\exp} = \bigcup_{m\geq 1} \mathcal{K}_P\big(2^k, m\big)$ of k-th level of the exponential-time alternation hierarchy by showing $\Sigma_k^{\exp} = \mathcal{K}_P\big(1^{k-1}2, 1\big) = \mathcal{K}_P\big(1^k, 2\big)$.

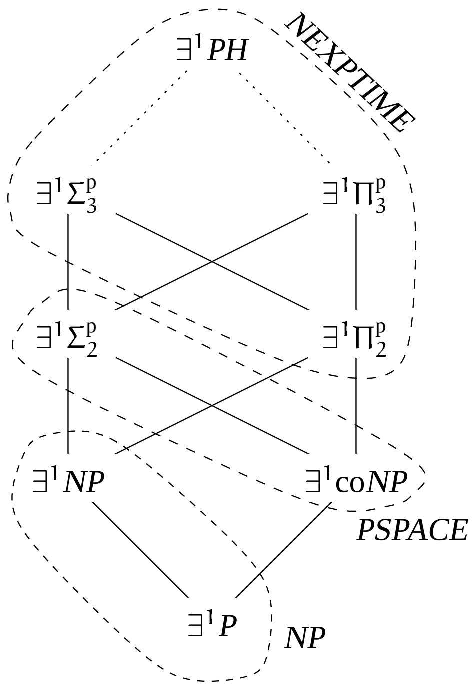
The figure 3.2 give an overview on the results on classes $\exists^1\mathcal{K}$ and $\exists^2\mathcal{K}$ where $\mathcal{K}$ are classes of the polynomial-time hierarchy.

Now, consider the equalities

$$\exists^2 P = NP \qquad \text{and} \qquad \exists^2 \text{co}NP = NEXPTIME$$

It is known that *NP* is properly included in *NEXPTIME*. Couldn't we conclude from this that $P \neq NP$? This is surely not true because the equations really mean $\exists^2 P^{02} = NP$ and $\exists^2\text{co}NP^{02} = NEXPTIME$ from which one can conclude only $P^{02} \neq NP^{02}$. This inequality concerns classes whose languages have besides a word input also a set (oracle) input. But it is known that there exist even single oracles for which *P* and *NP* are unequal [BGS75].

Finally, we emphasize that all the results on the analytic polynomial-time hierarchy obtained here are valid in every relativized world.



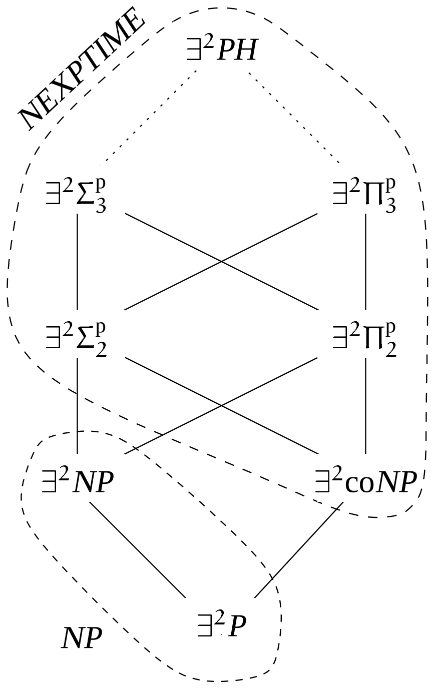


Figure 3.2: $\exists^1$ *and* $\exists^2$ *applied to classes of the polynomial-time hierarchy.*

CHAPTER 4

Bounding Queries in the Analytic Polynomial-Time Hierarchy

"Wenn einer mit Vergnügen in Reih und Glied zu einer Musik marschieren kann, dann verachte ich schon; er hat sein großes Gehirn nur aus Irrtum bekommen, da für ihn das Rückenmark schon völlig genügen würde. Diesen Schandfleck der Zivilisation sollte man so schnell wie möglich zum Verschwinden bringen. Heldentum auf Kommando, sinnlose Gewalttat und die leidige Vaterländerei wie glühend hasse ich sie, wie gemein und verächtlich erscheint mir der Krieg; ich möchte mich lieber in Stücke schlagen lassen, als mich an einem so elenden Tun beteiligen! Töten im Krieg ist nach meiner Auffassung um nichts besser als gewöhnlicher Mord."

A. Einstein

In this chapter, we investigate a hierarchy which refines the analytic polynomial-time hierarchy (§3) by considering restrictions on the number of oracle queries. This hierarchy is called *bounded analytic polynomial-time hierarchy*. We characterize classes this hierarchy by well-known complexity classes. All these characterizations remain valid if the queries are asked in a nonadaptive form, i.e. in "parallel".

An overview this chapter follows: We first present our new computational model comprising restrictions on the number of oracle queries and define some complexity classes (§4.1). Then, we extend the definition of the ∃-∀-quantifiers to include also these restrictions, and define the bounded analytic polynomial-time hierarchy (§4.2). Inclusion and equivalence rules are also helpful (§4.3). It is shown (§4.4) that each class from this hierarchy having a certain normal form coincides with one of the classes *NP*, co*NP*, *PSPACE*, Σ_k^{exp} or Π_k^{exp} ($k \geq 1$). After that, remainder classes are considered and the open cases are presented (§4.5). Finally, we make some comments about the results (§4.6).

4.1 Computational Model and Complexity Classes

We will extend previous definitions to our new context, i.e. including restrictions on the number of oracle queries. Limiting the number of queries that a machine can ask to an oracle (input of type $\sigma \in \{1, 2\}$) during its computation by a function $r : \mathbb{N} \to \mathbb{N}$ depending on the length of the word inputs, i.e. if n is the length of the word inputs then at most $r(n)$ queries can be asked to the oracle, we say that this is an *input of type* $\sigma[r]$.

Let $\Phi =_{\text{df}} \{0, 1, 2\} \cup \{\sigma[r] : \sigma \in \{1, 2\}, r : \mathbb{N} \to \mathbb{N}_+\}$ be the set of input types. For $k \geq 1$ and $\sigma_1, \ldots, \sigma_k \in \Phi$, we say that a Turing machine is of *type* $\sigma_1 \ldots \sigma_k$ if it processes instances of the form $(X_1, \ldots, X_k)$, where X_i is an input of type σ_i for $i = 1, \ldots, k$. Such a machine has $\|\{i : \sigma_i = 0\}\|$ ordinary input tapes and $k - \|\{i : \sigma_i = 0\}\|$ query tapes. Thus, a machine of type $\sigma_1 \ldots \sigma_k$ on input $(X_1, \ldots, X_k)$, whose i-th input is of type $\sigma_i = \tau[r]$ with $\tau \in \{1, 2\}$, can ask at most $r\left(\sum_{j<i, \sigma_j=0} |X_j|\right)$ queries to the oracle X_i. Although we consider new types of set inputs, for the length of the instances (as in §2.2.2) only the word inputs remain relevant. The *length of an instance* $X = (X_1, \ldots, X_k)$, denoted $|X|$, is defined by $|X| =_{\text{df}} \sum_{1 \leq i \leq k, \sigma_i = 0} |X_i|$, i.e. the sum of the length of the word inputs. In what follows we also assume here that X contains at least one word input.

Computation trees and accepting paths are defined as in §2.2.2. Let $k \geq 1$ and σ_1, $\ldots$,$\sigma_k \in \Phi$. For a Turing machine M of type $\sigma_1 \ldots \sigma_k$ on input $(X_1, \ldots, X_k)$ being deterministic or nondeterministic (X_i is an input of type σ_i), its *computation tree*, denoted $\beta_M(X_1, \ldots, X_k)$, is a possibly infinite tree whose nodes are configurations, the root being the initial configuration, and for any node α, its sons are those configurations which are immediate successors of α. Obviously, for M being deterministic, its computation trees are also paths. For M being nondeterministic, without loss of generality we assume that the trees are binary. An *accepting path* of $\beta_M(X_1, \ldots, X_k)$ is a path in $\beta_M(X_1, \ldots, X_k)$ which has the same root node and ends in an accepting configuration.

For $k \geq 1$ and $\sigma_1, \ldots, \sigma_k \in \Phi$, a Turing machine M of type $\sigma_1 \ldots \sigma_k$ is called *polynomial-time* (*polynomial-space*) if there exists a polynomial p such that M halts on every path of every instance $X = (X_1, \ldots, X_k)$ in an accepting or rejecting state using no more than $p(|X|)$ steps (tape cells, respectively). The space bound applies also to the length of oracle queries. For a Turing machine M of type $\sigma_1 \ldots \sigma_k$ define

M is deterministic:

$$L(M) =_{\text{df}} \{(X_1, \ldots, X_k) : \beta_M(X_1, \ldots, X_k) \text{ is an accepting path}\}$$

M is nondeterministic:

$$L(M) =_{\text{df}} \{(X_1, \ldots, X_k) : \beta_M(X_1, \ldots, X_k) \text{ contains an accepting path}\}$$

as the *language accepted by* M.

In limited nondeterminism we consider Turing machines which make a bounded number of nondeterministic steps [KF80, GLM96]. For $k \geq 1$, $\sigma_1, \ldots, \sigma_k \in \Phi$ and a function $f : \mathbb{N} \to \mathbb{N}$, a Turing machine M of type $\sigma_1 \ldots \sigma_k$ is called f-*nondeterministic* if M on every path of every instance $X = (X_1, \ldots, X_k)$ makes at most $f(|X|) =_{\text{df}} f\left(\sum_{\sigma_i=0} |X_i|\right)$

nondeterministic steps. Now, for $k \geq 1$ and $\sigma_1, \ldots, \sigma_k \in \Phi$ define

$$\begin{aligned}
P^{\sigma_1 \ldots \sigma_k} &=_{\text{df}} \{L(M) : M \text{ is a polynomial-time deterministic} \\
&\qquad\qquad \text{Turing machine of type } \sigma_1 \ldots \sigma_k\} \\
(f)\text{-}P^{\sigma_1 \ldots \sigma_k} &=_{\text{df}} \{L(M) : M \text{ is a polynomial-time } f\text{-nondeterministic} \\
&\qquad\qquad \text{Turing machine of type } \sigma_1 \ldots \sigma_k\} \\
PSPACE^{\sigma_1 \ldots \sigma_k} &=_{\text{df}} \{L(M) : M \text{ is a polynomial-space deterministic} \\
&\qquad\qquad \text{Turing machine of type } \sigma_1 \ldots \sigma_k\}
\end{aligned}$$

which are called *classes of type* $\sigma_1 \ldots \sigma_k$. Obviously, $L \in (f)\text{-}P^{\sigma_1 \ldots \sigma_k}$ if and only if there exist an $L' \in P^{\sigma_1 \ldots \sigma_k 0}$ and a polynomial p such that

$$X \in L \iff \exists u\,(|u| = \min\{f(|X|)\,, p(|X|)\} \wedge (X, u) \in L')$$

Finally, we define $(f)\text{-}P =_{\text{df}} (f)\text{-}P^0$.

4.2 Bounding Queries in Set Quantifiers and a New Hierarchy

The analytic polynomial-time hierarchy (§3) was defined using ∃-∀-quantifiers varying over words and oracles of type 1 and 2. We will refine this hierarchy (§4.2.2) considering also ∃-∀-quantifiers varying over oracles of type $1[r]$ and $2[r]$ (§4.2.1).

4.2.1 Bounding Queries in Existential and Universal Set Quantifiers

We will investigate an ∃-∀-hierarchy over P using word quantifiers as well as set quantifiers varying over oracles of type τ for $\tau \in \{1, 2\} \cup \{\sigma[r] : \sigma \in \{1, 2\}, r : \mathbb{N} \to \mathbb{N}_+\}$. The previous definitions of the existential and universal quantifiers (§3.1.1) comprised only quantifiers varying over words and oracles of types 1 and 2. Thus, we will extend these definitions to include also quantifiers varying over oracles of type $1[r]$ and $2[r]$. The classes $P^{\sigma_1 \ldots \sigma_k}$ with $\sigma_1, \ldots, \sigma_k \in \Phi$ are the starting point for building up this new hierarchy. Next, we define inductively new classes and in parallel the quantifiers. The definitions of the word quantifiers and set quantifiers of type 1 and 2 in our new context are similar to the previous definitions, only the type of the class in question can change. However, for convenience they are also included. Let $k \geq 1$ and $\sigma_1, \ldots, \sigma_k, \sigma \in \Phi$. If $\mathcal{K}$ is a class of type $\sigma_1 \ldots \sigma_k \sigma$ then

For $\sigma = 0$: $\exists^p \mathcal{K}$ and $\forall^p \mathcal{K}$ are classes of type $\sigma_1 \ldots \sigma_k$ which are defined as follows

$L \in \exists^p \mathcal{K} \iff_{\text{df}}$ there exist an $L' \in \mathcal{K}$ and a polynomial p, such that

$$(X_1, \ldots, X_k) \in L \longleftrightarrow \exists x \Big(|x| \leq p\big(\sum_{\substack{\sigma_i = 0 \\ i \leq k}} |X_i|\big) \wedge (X_1, \ldots, X_k, x) \in L' \Big)$$

$L \in \forall^p \mathcal{K} \Longleftrightarrow_{df}$ there exist an $L' \in \mathcal{K}$ and a polynomial p, such that

$$(X_1, \ldots, X_k) \in L \longleftrightarrow \forall x \Big(|x| \leq p\big(\sum_{\substack{\sigma_i = 0 \\ i \leq k}} |X_i|\big) \to (X_1, \ldots, X_k, x) \in L' \Big)$$

(Using simple encoding arguments it is easy to see that one can use equivalently "$=$" instead of "$\leq$" in these definitions.)

For $\sigma \neq 0$: $\exists^\sigma \mathcal{K}$ and $\forall^\sigma \mathcal{K}$ are classes of type $\sigma_1 \ldots \sigma_k$ which are defined as follows

$L \in \exists^\sigma \mathcal{K} \Longleftrightarrow_{df}$ there exists an $L' \in \mathcal{K}$, such that

$$(X_1, \ldots, X_k) \in L \longleftrightarrow \exists X \left((X_1, \ldots, X_k, X) \in L' \right)$$

$L \in \forall^\sigma \mathcal{K} \Longleftrightarrow_{df}$ there exists an $L' \in \mathcal{K}$, such that

$$(X_1, \ldots, X_k) \in L \longleftrightarrow \forall X \left((X_1, \ldots, X_k, X) \in L' \right)$$

To make clear which type of input is used, we also write $\exists^\tau X$ instead of $\exists X$, and $\forall^\tau X$ instead of $\forall X$. Furthermore, in what follows we also write $\exists^\sigma[r]$ instead of $\exists^{\sigma[r]}$, and $\forall^\sigma[r]$ instead of $\forall^{\sigma[r]}$.

Now, we translate some abbreviations and definitions from §3.1.1 to our new context. For $Q = \exists$ ($Q = \forall$), the set of existential (universal, respectively) quantifiers is denoted by $\Gamma_{[p,Q]} =_{df} \{Q^\sigma : \sigma \in \Phi\}$ and the set of existential and universal quantifiers by $\Gamma_{[p]} =_{df} \Gamma_{[p,\exists]} \cup \Gamma_{[p,\forall]}$. For $k \geq 0$, $Q_1, \ldots, Q_k \in \{\exists, \forall\}$ and $\tau_1, \ldots, \tau_k \in \Phi$, let $\tau(Q_1^{\tau_1} \ldots Q_k^{\tau_k}) =_{df} \sigma_1 \ldots \sigma_k$ be the type of the operator (or quantifier) string $Q_1^{\tau_1} \ldots Q_k^{\tau_k}$, where $\sigma_i = 0$ if $\tau_i = p$ and $\sigma_i = \tau_i$ otherwise ($i = 1, \ldots, k$). For $Q = Q_1^{\tau_1} \ldots Q_k^{\tau_k}$ and $X = (X_1, \ldots, X_k)$ we write QX instead of $Q_1^{\tau_1} X_1 \ldots Q_k^{\tau_k} X_k$. Furthermore, we define $\overline{Q} =_{df} \overline{Q}_1^{\tau_1} \ldots \overline{Q}_k^{\tau_k}$. The following proposition is evident.

Proposition 4.1. *Let* $\mu \in \Phi^*$ *and* $Q \in \Gamma_{[p]}^*$. *Then* $\mathrm{co}QP^{\mu\tau(Q)} = \overline{Q}P^{\mu\tau(Q)}$.

Proof. See proof of Proposition 3.1. ❑

4.2.2 The Bounded Analytic Polynomial-Time Hierarchy and a Normal Form

As for the analytic polynomial-time hierarchy, we are here particularly interested in the classes of type 0, i.e. in "ordinary" classes of languages. In this case, we will also omit the superscripts to P, i.e. for quantifier string $Q \in \Gamma_{[p]}^*$ we define $QP =_{df} QP^{0\tau(Q)}$. Next, the bounded analytic polynomial-time hierarchy is defined. For quantifier strings $Q \in \Gamma_{[p]}^*$, the classes QP are called the classes of the *bounded analytic polynomial-time hierarchy*. Because of

$$Q_{pref} Q^{\tau[r]} Q_{suf} P \subseteq Q_{pref} Q^\tau Q_{suf} P$$

where $Q_{\text{pref}}, Q_{\text{suf}} \in \Gamma^*_{[p]}$, $Q \in \{\exists, \forall\}$, $\tau \in \{1,2\}$ and $r : \mathbb{N} \to \mathbb{N}_+$, the union of all classes of the bounded analytic polynomial-time hierarchy is also *APH*. On the other hand, for r being so large that is not a real restriction we obtain an equality in above inclusion

$$Q_{\text{pref}} Q^{\tau[r]} Q_{\text{suf}} P = Q_{\text{pref}} Q^{\tau} Q_{\text{suf}} P$$

Hence, for Q^τ is equivalent to take $Q^{\tau[r]}$ with r being so large that it is not a real restriction and vice versa.

For the analytic polynomial-time hierarchy (§3.2.2) we define the normal form classes $\mathcal{K}_P(\tau, m)$ with $\tau \in \{1,2\}^*$ and $m \geq 0$. Obviously, these classes are also classes of the bounded analytic polynomial-time hierarchy. Next, we extend this definition to comprise also quantifiers of type $1[r]$ and $2[r]$. For $l \geq 1$, let $Q_l =_{\text{df}} \exists$ if l is odd and $Q_l =_{\text{df}} \forall$ otherwise. For $k, m \geq 0$ and $\tau_1, \tau_2, \tau_3, \ldots, \tau_k \in \Phi \setminus \{0\}$ we define

$$\mathcal{K}_P(\tau_1\tau_2\tau_3\ldots\tau_k, m) =_{\text{df}} \exists^{\tau_1}\forall^{\tau_2}\exists^{\tau_3}\ldots Q_k^{\tau_k} Q_{k+1}^{p} Q_{k+2}^{p} \ldots Q_{k+m}^{p} P$$

4.3 Inclusion and Equivalence Rules

Inclusion and equivalence rules will be helpful to relate classes of the bounded analytic polynomial-time hierarchy. These rules will be applied in a similar sense as for "$\rightarrow_P$" and "$\leftrightarrow_P$". For $R, S \in \Gamma^*_{[p]}$, the *inclusion rule* $R \rightarrow_{[P]} S$ is valid if the replacement of the quantifier string R by the string S in any context does not diminish the class in question, i.e. $RQP^{\mu\tau(R)\tau(Q)} \subseteq SQP^{\mu\tau(S)\tau(Q)}$ for all $Q \in \Gamma^*_{[p]}$ and $\mu \in \Phi^*$. We say that the *equivalence rule* $R \leftrightarrow_{[P]} S$ is valid if the replacement of the quantifier string R by the string S in any context does not change the class in question, i.e. $RQP^{\mu\tau(R)\tau(Q)} = SQP^{\mu\tau(S)\tau(Q)}$ for all $Q \in \Gamma^*_{[p]}$ and $\mu \in \Phi^*$. Obviously, we have $R \leftrightarrow_{[P]} S$ if and only if $R \rightarrow_{[P]} S$ and $S \rightarrow_{[P]} R$.

For a rule $R \rightarrow_{[P]} S$, we will also have to prove "its complement" $\overline{R} \rightarrow_{[P]} \overline{S}$. However, as for classes of the analytic polynomial-time hierarchy, the following proposition shows that only one of them has to be proved.

Proposition 4.2 (Complementation). *Let* $R, S \in \Gamma^*_{[p]}$. *If* $R \rightarrow_{[P]} S$ *then* $\overline{R} \rightarrow_{[P]} \overline{S}$.

Proof. The proof is the same as for Proposition 3.3. Let $Q \in \Gamma^*_{[p]}$ and $\mu \in \Phi^*$. Then

$$\begin{aligned} L \in \overline{R}QP^{\mu\tau(R)\tau(Q)} &\Longrightarrow \overline{L} \in \text{co}\overline{R}QP^{\mu\tau(R)\tau(Q)} && \\ &\Longrightarrow \overline{L} \in R\overline{Q}P^{\mu\tau(R)\tau(Q)} && \text{by Proposition 4.1} \\ &\Longrightarrow \overline{L} \in S\overline{Q}P^{\mu\tau(S)\tau(Q)} && \text{by } R \rightarrow_{[P]} S \\ &\Longrightarrow \overline{L} \in \text{co}\overline{S}QP^{\mu\tau(S)\tau(Q)} && \text{by Proposition 4.1} \\ &\Longrightarrow L \in \overline{S}QP^{\mu\tau(S)\tau(Q)} && \end{aligned}$$ ❑

In 3.2.1, some rules were proved to relate classes of the analytic polynomial-time hierarchy. In our new context we add as necessary the terms $[r]$. Some of the rules (Lemma 3.4) show relations between the existential (universal, respectively) quantifiers of different types. They can be restated as follows.

Lemma 4.3. *Let* $\tau \in \{1,2\}$ *and* $r : \mathbb{N} \to \mathbb{N}$*. Then*

(1) $\varepsilon \to_{[P]} \exists^p$ *and* $\varepsilon \to_{[P]} \forall^p$*;*

(2) $\varepsilon \leftrightarrow_{[P]} \exists^\tau[0]$ *and* $\varepsilon \leftrightarrow_{[P]} \forall^\tau[0]$*;*

(3) $\exists^p \to_{[P]} \exists^\tau$ *and* $\forall^p \to_{[P]} \forall^\tau$*;*

(4) $\exists^1[r] \to_{[P]} \exists^2[r]$ *and* $\forall^1[r] \to_{[P]} \forall^2[r]$.

Proof. The proof of Statement (2) is evident, since no queries are allowed to the oracle. For the other statements, the proof follows as in Lemma 3.4. ❑

We next restate some rules (Lemma 3.6) showing how to melt neighboured existential (universal, respectively) quantifiers.

Lemma 4.4. *Let* $\tau \in \{1,2\}$ *and* $r, r' : \mathbb{N} \to \mathbb{N}$*. Then*

(1) $\exists^p\exists^p \leftrightarrow_{[P]} \exists^p$ *and* $\forall^p\forall^p \leftrightarrow_{[P]} \forall^p$*;*

(2) $\exists^p\exists^\tau[r] \to_{[P]} \exists^\tau$ *and* $\forall^p\forall^\tau[r] \to_{[P]} \forall^\tau$*;*

(3) $\exists^\tau[r]\,\exists^p \to_{[P]} \exists^\tau$ *and* $\forall^\tau[r]\,\forall^p \to_{[P]} \forall^\tau$*;*

(4) $\exists^2[r]\,\exists^2[r'] \to_{[P]} \exists^2[r+r']$ *and* $\forall^2[r]\,\forall^2[r'] \to_{[P]} \forall^2[r+r']$.

Proof. The proof follows as in Lemma 3.6. Note that the proof for $\tau = 2$ is the same as for $\tau = 1$. ❑

Some rules show how to reduce the type 2 of a set quantifier, for example the rule $\exists^2 \to_P \exists^1\exists^1\forall^p$ stated in the proof of Lemma 3.6. Following the proof of this rule we obtain $\exists^2[r] \to_{[P]} \exists^1[1]\,\exists^1[r+1]\,\forall^p$ in our new context. However, in this proof we can make some refinements and obtain a similar rule requiring less queries in the second set quantifier. This is shown in the following lemma.

Lemma 4.5. *Let* $r : \mathbb{N} \to \mathbb{N}_+$*. Then*

(1) $\exists^2[r] \to_{[P]} \exists^1[1]\,\exists^1[r-1]\,\forall^p$ *and* $\forall^2[r] \to_{[P]} \forall^1[1]\,\forall^1[r-1]\,\exists^p$*;*

(2) $\exists^2[r] \to_{[P]} \exists^1[r-1]\,\exists^1[1]\,\forall^p$ *and* $\forall^2[r] \to_{[P]} \forall^1[r-1]\,\forall^1[1]\,\exists^p$.

Proof. We prove the first rule of every statement, since the other follow by complementation. Because of obvious rule $\exists^1[1]\,\exists^1[r-1] \to_{[P]} \exists^1[r-1]\,\exists^1[1]$, it suffices to prove $\exists^2[r] \to_{[P]} \exists^1[1]\,\exists^1[r-1]\,\forall^p$. For $r = 1$ is evident. For $r \geq 2$ let $\mu \in \Phi^*$ and $Q \in \Gamma^*_{[p]}$, and let $L \in \exists^2[r]\,Q P^{\mu 2[r]\tau(Q)}$. There exists an $L_1 \in P^{\mu 2[r]\tau(Q)}$ such that for a suitable polynomial p

$$
\begin{aligned}
X \in L \Longleftrightarrow & \exists^2 U\, QY\,((X,U,Y) \in L_1) \\
\Longleftrightarrow\ & \exists^2 U\, \exists^1 W \Big(\forall^p u\, \forall^p v \Big(|u|, |v| \le p(|X|) \to (u \in U \leftrightarrow v01\widehat{u} \in W)\Big) \\
& \qquad \wedge QY\Big((X,U,Y) \in L(M)\Big)\Big)
\end{aligned}
$$

$\big($take $W = \{v01\widehat{u} : u, v \in \{0,1\}^* \wedge u \in U\}$ for example, and let M be a polynomial-time machine of type $\mu 2[r]\,\tau(Q)$ accepting $L_1\big)$

$$
\begin{aligned}
\Longleftrightarrow\ & \exists^1 U\, \exists^1 W\, \forall^p a\, \forall^p u\, \forall^p v\, QY\Big(|u|, |v| \le p(|X|) \to \\
& \qquad \Big((a = 0 \to (u \in U \leftrightarrow v01\widehat{u} \in W)) \wedge \\
& \qquad (a = 1 \to (X,U,W,Y) \in L(M'))\Big)\Big)
\end{aligned}
$$

$\big(M'$ works on input (X,U,W,Y) as M on input (X,U,Y) but it only asks the first query to U like M does. Instead of asking u to U after queries $u_1,\ldots,u_m$ $(m \ge 1)$ it asks $01\widehat{u}_1 01\widehat{u}_2 01 \ldots 01\widehat{u}_m 01\widehat{u}$ to W. Note that for every (X,a,u,v,Y), oracle U is asked exactly once and oracle W is asked at most $r-1$ times in a type 1 manner.$\big)$

This shows $L \in \exists^1[1]\,\exists^1[r-1]\,\forall^p\forall^p\forall^p Q P^{\mu 1[1]1[r-1]000\tau(Q)}$ and using the rules of Lemma 4.4, $L \in \exists^1[1]\,\exists^1[r-1]\,\forall^p Q P^{\mu 1[1]1[r-1]0\tau(Q)}$. ❏

Rules stated in the proof of Lemma 3.7 show how to shift existential and universal word quantifiers. They can be restated as follows.

Lemma 4.6. *Let* $\tau \in \{1,2\}$ *and* $r : \mathbb{N} \to \mathbb{N}$. *Then*

$$\exists^p \forall^\tau[r] \to_{[P]} \forall^\tau[r]\,\exists^p \quad \textit{and} \quad \forall^p \exists^\tau[r] \to_{[P]} \exists^\tau[r]\,\forall^p$$

Proof. The proof follows as in Lemma 3.7. ❏

Finally, we show an "equivalence rule" which is valid only in a special context. It says that set quantifiers applied to P are not more powerful than word quantifiers.

Lemma 4.7. *Let* $\mu \in \Phi^*$. *Then*

$$\exists^p P^{\mu 0} = \exists^1 P^{\mu 1} = \exists^2 P^{\mu 2} \quad \textit{and} \quad \forall^p P^{\mu 0} = \forall^1 P^{\mu 1} = \forall^2 P^{\mu 2}$$

Proof. The proof follows as in Lemma 3.5. ❏

4.4 Characterizing the Classes $\mathcal{K}_P(\cdot)$ and co$\mathcal{K}_P(\cdot)$

We will characterize all the classes of the bounded analytic polynomial-time hierarchy which have the form $\mathcal{K}_P(\tau_1 \ldots \tau_k, m)$ or co$\mathcal{K}_P(\tau_1 \ldots \tau_k, m)$, where $k \ge 0$, $m \ge 1$ and $\tau_1, \ldots, \tau_k \in \{1,2\} \cup \{\sigma[r] : \sigma \in \{1,2\}, r : \mathbb{N} \to \mathbb{N}_+\}$. It is well-known that such a class built without set quantifiers coincides with a class of the (arithmetic) polynomial-time hierarchy and vice versa, i.e. $\mathcal{K}_P(\varepsilon, m) = \Sigma^p_m$ and co$\mathcal{K}_P(\varepsilon, m) = \Pi^p_m$ for $m \geq 0$ [SM73, Sto77, Wra77]. Thus, the interesting cases are those involving set quantifiers. It will turn out that each of these classes in normal form involving set quantifiers coincides with one of the classes *NP*, co*NP*, *PSPACE*, Σ^{exp}_k or Π^{exp}_k ($k \geq 1$) and vice versa. In what follows we will state and prove only the results for $\mathcal{K}_P(\cdot)$. The results for co$\mathcal{K}_P(\cdot)$ are immediate consequences.

The Table 4.1 includes the corresponding results for the classes $\mathcal{K}_P(\tau_1 \ldots \tau_k, m)$, where $k, m \geq 1$ and $\tau_1, \ldots, \tau_k \in \{\sigma[r] : \sigma \in \{1,2\}, r : \mathbb{N} \to \mathbb{N}_+\}$, i.e. for the cases where there is a restriction to the number of queries for every set quantifier.

$m = 1$	$r_k = 1$	$k = 1$	$\mathcal{K}_P(\sigma_1[1], 1) =$ co*NP*	4.10, 4.11
		$k = 2$	$\mathcal{K}_P(\sigma_1[r_1]\,\sigma_2[1], 1) =$ *NP*	4.13
		$k = 3$	$\mathcal{K}_P(\sigma_1[1]\,\sigma_2[r_2]\,\sigma_3[1], 1) =$ *PSPACE*	4.13
			$\mathcal{K}_P(\sigma_1[r_1]\,\sigma_2[r_2]\,\sigma_3[1], 1) =$ *NEXPTIME* $(r_1 \geq 2)$	4.13
		$k \geq 4$	$\mathcal{K}_P(\sigma_1[r_1] \ldots \sigma_{k-1}[r_{k-1}]\,\sigma_k[1], 1) = \Sigma^{exp}_{k-3}$ $(r_{k-2} = 1)$	4.13
			$\mathcal{K}_P(\sigma_1[r_1] \ldots \sigma_{k-1}[r_{k-1}]\,\sigma_k[1], 1) = \Sigma^{exp}_{k-2}$ $(r_{k-2} \geq 2)$	4.13
	$r_k = 2$	$k = 1$	$\mathcal{K}_P(\sigma_1[2], 1) =$ *PSPACE*	4.10, 4.11
		$k \geq 2$	$\mathcal{K}_P(\sigma_1[r_1] \ldots \sigma_{k-1}[r_{k-1}]\,\sigma_k[2], 1) = \Sigma^{exp}_{k-1}$	4.12
	$r_k \geq 3$	$k = 1$	$\mathcal{K}_P(1[r_1], 1) =$ *PSPACE*	4.10
			$\mathcal{K}_P(2[r_1], 1) =$ *NEXPTIME*	4.11
		$k \geq 2$	$\mathcal{K}_P(\sigma_1[r_1] \ldots \sigma_{k-1}[r_{k-1}]\,1[r_k], 1) = \Sigma^{exp}_{k-1}$	4.12
			$\mathcal{K}_P(\sigma_1[r_1] \ldots \sigma_{k-1}[r_{k-1}]\,2[r_k], 1) = \Sigma^{exp}_{k}$	4.12
$m \geq 2$	$r_k \geq 1$	$k \geq 1$	$\mathcal{K}_P(\sigma_1[r_1] \ldots \sigma_k[r_k], m) = \Sigma^{exp}_k$	4.9

Table 4.1: *Characterization of the classes* $\mathcal{K}_P(\sigma_1[r_1] \ldots \sigma_k[r_k], m)$ *with* $\sigma_i \in \{1,2\}$ *and* $r_i : \mathbb{N} \to \mathbb{N}_+$. *The last column refers to the Theorem(s) which states the corresponding result.*

Obviously, we have in all cases stated in the Table 4.1

$$\mathcal{K}_P(\tau_1 \ldots \tau_{i-1}\sigma_i[r_i]\,\tau_{i+1} \ldots \tau_k, m) = \mathcal{K}_P(\tau_1 \ldots \tau_{i-1}\sigma_i[3]\,\tau_{i+1} \ldots \tau_k, m)$$

for $\sigma_i = 1, 2$, $r_i \geq 3$ and $\tau_1, \ldots, \tau_{i-1}, \tau_{i+1} \ldots, \tau_k \in \{\sigma[r] : \sigma \in \{1,2\}, r : \mathbb{N} \to \mathbb{N}_+\}$. Choosing r_i so large that it is not a real restriction we have also

$$\mathcal{K}_P(\tau_1 \ldots \tau_{i-1}\sigma_i\tau_{i+1} \ldots \tau_k, m) = \mathcal{K}_P(\tau_1 \ldots \tau_{i-1}\sigma_i[3]\,\tau_{i+1} \ldots \tau_k, m)$$

Iterating this step we obtain results for all $\mathcal{K}_P(\tau_1 \ldots \tau_k, m)$, where $k, m \geq 1$ and τ_1, . . . ,$\tau_k \in \{1,2\} \cup \{\sigma[r] : \sigma \in \{1,2\}, r : \mathbb{N} \to \mathbb{N}_+\}$.

Hence, proving the results of the Table 4.1 we obtain the desire characterizations (§4.4.1). In addition it is shown that all these characterizations remain valid if the oracle machines are allowed to make only parallel queries (§4.4.2), i.e. they have to form a list of all queries before any of them is queried to the oracle.

4.4.1 Characterizing the Classes $\mathcal{K}_P(\sigma_1[r_1] \ldots \sigma_k[r_k], m)$

Next, the classes having the form $\mathcal{K}_P(\sigma_1[r_1] \ldots \sigma_k[r_k], m)$ are characterized by well-known complexity classes for $k, m \geq 1$, $\sigma_1, \ldots, \sigma_k \in \{1,2\}$ and $r_1, \ldots, r_k : \mathbb{N} \to \mathbb{N}_+$

(the results for co$\mathcal{K}_P(\sigma_1[r_1]\dots\sigma_k[r_k]\,,m)$ are immediate consequences). We start with translating a result from §3.3 in the language of the bounded analytic polynomial-time hierarchy. Theorem 3.11 states which classes of the analytic polynomial-time hierarchy contain a level of the exponential-time alternation hierarchy. In the proof of this result some remarks on the number of queries to the oracles were made. This can be restated as follows:

Theorem 4.8. *For* $k \geq 1$ *the inclusions*

$$\Sigma_k^{\mathrm{exp}} \subseteq \mathcal{K}_P\big((1[1])^k, 2\big) \cap \mathcal{K}_P\big((1[1])^{k-1}2[3]\,, 1\big)$$

are valid.

Proof. The proof follows as in Theorem 3.11. ❑

Using previous theorem and a results obtained for the classes of the analytic polynomial-time hierarchy, we obtain that each class containing at least one set quantifier and at least two word quantifiers coincides with one class of the exponential-time alternation hierarchy.

Theorem 4.9. *Let* $k \geq 1$, $m \geq 2$, $\sigma_1, \dots, \sigma_k \in \{1,2\}$ *and* $r_1, \dots, r_k : \mathbb{N} \to \mathbb{N}_+$. *Then*

$$\mathcal{K}_P(\sigma_1[r_1]\dots\sigma_k[r_k]\,, m) = \Sigma_k^{\mathrm{exp}}$$

Proof. Direct from Theorems 4.8 and 3.13. ❑

Thus, remains to consider the classes $\mathcal{K}_P(\sigma_1[r_1]\dots\sigma_k[r_k]\,,1)$ with $k \geq 1$. The simplest classes of these types are those containing only one set quantifier of type 1 which turn out to coincide with co*NP* or *PSPACE*.

Theorem 4.10. *Let* $r : \mathbb{N} \to \mathbb{N}$. *Then*

(1) $\mathcal{K}_P(1[1]\,,1) = $ co*NP;*

(2) $\mathcal{K}_P(1[r]\,,1) = $ *PSPACE for* $r \geq 2$.

Proof. Statement (1): The proof of co$NP = \forall^p P \subseteq \exists^1[1]\,\forall^p P = \mathcal{K}_P(1[1]\,,1)$ is evident (Lemma 4.3). Next we prove $\exists^1[1]\,\forall^p P \subseteq \forall^p P$. Let $L \in \exists^1[1]\,\forall^p P$. By definition there exist an $L_1 \in P^{01[1]0}$ and a polynomial p such that

$$x \in L \Longleftrightarrow \exists^1 U\, \forall^p v\, (|v| = p(|x|) \to (x, U, v) \in L_1)$$

where M is a polynomial-time machine of type $01\,[1]\,0$ accepting L_1 is such a way that the oracle U is queried only once. Let f be a function computable in polynomial-time such that the machine M on input (x, U, v) queries $f(x, v)$ to oracle U. Thus, there exist an $L_2 \in P^{000}$ and a polynomial q such that

$$\begin{aligned} x \in L &\Longleftrightarrow \exists^1 U\, \forall^p v\, (|v| = p(|x|) \to (x, v, c_U(f(x,v))) \in L_2) \\ &\Longleftrightarrow \forall^p u\, \forall^p v\, \forall^p w \Big(|u| \leq q(|x|) \wedge |v| = |w| = p(|x|) \to \\ &\qquad \big((f(x,v) = u \to (x, v, 1) \in L_2) \vee (f(x,w) = u \to (x, w, 0) \in L_2)\big)\Big) \end{aligned}$$

Therefore, $L \in \forall^p\forall^p\forall^p P$. Now, using Lemma 4.4 we get the desired result.

Statement (2): The inclusion $\exists^1[2]\,\forall^p P \subseteq \exists^1\forall^p P$ is evident and $\exists^1\forall^p P = \mathit{PSPACE}$ follows by Theorem 3.9. Thus, only $\mathit{PSPACE} \subseteq \exists^1[2]\,\forall^p P$ has to be proved. We use Cai and Furst's characterization of *PSPACE* [CF91]. Let A_5 be the group of even permutations on $[5] =_{\text{df}} \{0,\dots,4\}$, and let $\circ$ be the multiplication of this group. Cai and Furst proved that for every language $L \in \mathit{PSPACE}$ there exist a function $f : \{0,1\}^* \times \{0,1\}^* \to A_5$ computable in polynomial-time and a polynomial p such that

$$x \in L \iff f(x, 1^{p(|x|)}) \circ f(x, 1^{p(|x|)-1}0) \circ \dots \circ f(x, 0^{p(|x|)})(1) = 1$$

Consequently,

$$\begin{aligned} x \in L \iff \exists g \Bigg((g : \{0,1\}^* \to [5]) \wedge \forall u \Bigg(|u| = p(|x|) \to \\ \Big(\Big(u = 0^{p(|x|)} \to f\,(x,u)\,(1) = g\,(u)\Big) \\ \wedge \Big(u \notin \big\{0^{p(|x|)}, 1^{p(|x|)}\big\} \to f\,(x,u)\,(g(\overleftrightarrow{u})) = g\,(u)\Big) \\ \wedge \Big(u = 1^{p(|x|)} \to f\,(x,u)\,(g(\overleftrightarrow{u})) = 1\Big)\Big)\Bigg)\Bigg) \\ \iff \exists^2 U\, \forall^p u\, \forall^p i\, \forall^p j \Bigg(|u| = p(|x|) \wedge 0 \le i,j \le 4 \to \\ \Big(\Big(u = 0^{p(|x|)} \to (f\,(x,u)\,(1) = i \leftrightarrow u0^i \in U)\Big) \\ \wedge \Big(u \notin \big\{0^{p(|x|)}, 1^{p(|x|)}\big\} \wedge \overleftrightarrow{u}0^i \in U \to (f\,(x,u)\,(i) = j \leftrightarrow u0^j \in U)\Big) \\ \wedge \Big(u = 1^{p(|x|)} \wedge \overleftrightarrow{u}0^i \in U \to f\,(x,u)\,(i) = 1\Big)\Big)\Bigg) \end{aligned}$$

However, for each (x, U, u, i, j) such that $u \notin \big\{0^{p(|x|)}, 1^{p(|x|)}\big\}$ the two queries $\overleftrightarrow{u}0^i$ and $u0^j$ are asked to U which is not a type 1 querying. To overcome this difficulty we encode the words from $\big(\{0,1\}^{p(|x|)} \setminus \big\{1^{p(|x|)}\big\}\big) \times \{0\}^{\le 4}$ by an injective function α which has the property that, for every $u \notin \big\{0^{p(|x|)}, 1^{p(|x|)}\big\}$ and $0 \le i,j \le 4$, either $\alpha(\overleftrightarrow{u}, 0^i)$ is an initial word of $\alpha(u, 0^j)$ or vice versa (take for example the function α of Figure 4.1 with $n = p(|x|)$ and $l = 4$). Now instead of querying $\overleftrightarrow{u}0^i$ and $u0^j$ to U the queries $\alpha(\overleftrightarrow{u}, 0^i)$ and $\alpha(u, 0^j)$ are made to the oracle $V =_{\text{df}} \big\{\alpha(u, 0^i) : 0 \le i \le 4 \wedge u0^i \in U\big\}$ in a type 1 manner. Therefore,

$$x \in L \iff \exists^1 V\, \forall^p u\, \forall^p i\, \forall^p j \Big((|u| = p(|x|) \wedge 0 \le i,j \le 4) \to (x, V, u, i, j) \in L_1 \Big)$$

where $(x, V, u, i, j) \in L_1 \iff_{\text{df}}$

$$\begin{aligned} &\Big(u = 0^{p(|x|)} \to (f\,(x,u)\,(1) = i \leftrightarrow \alpha(u, 0^i) \in V)\Big) \\ \wedge &\Big(u \notin \big\{0^{p(|x|)}, 1^{p(|x|)}\big\} \wedge \alpha(\overleftrightarrow{u}, 0^i) \in V \to (f\,(x,u)\,(i) = j \leftrightarrow \alpha(u, 0^j) \in V)\Big) \\ \wedge &\Big(u = 1^{p(|x|)} \wedge \alpha(\overleftrightarrow{u}, 0^i) \in V \to f\,(x,u)\,(i) = 1\Big) \end{aligned}$$

Obviously, L_1 can be accepted by a deterministic polynomial-time oracle Turing machine which, on input (x, V, u, i, j), queries the oracle V in type 1 manner and at most two times. Hence, $L_1 \in P^{01[2]000}$ and $L \in \exists^1[2]\,\forall^p\forall^p\forall^p P$. Now, using Lemma 4.4 we conclude $L \in \exists^1[2]\,\forall^p P$. ❑

u		$\alpha(u, 0^i)$
$\mathbf{000}0$	$\longrightarrow$	$0^4\mathbf{0}0^4\mathbf{0}0^4\mathbf{0}0^i$
$\mathbf{00}01$	$\longrightarrow$	$0^4\mathbf{0}0^4\mathbf{0}0^i$
$\mathbf{001}0$	$\longrightarrow$	$0^4\mathbf{0}0^4\mathbf{0}0^4\mathbf{1}0^i$
$\mathbf{0}011$	$\longrightarrow$	$0^4\mathbf{0}0^i$
$\mathbf{010}0$	$\longrightarrow$	$0^4\mathbf{0}0^4\mathbf{1}0^4\mathbf{0}0^i$
$\mathbf{01}01$	$\longrightarrow$	$0^4\mathbf{0}0^4\mathbf{1}0^i$
$\mathbf{011}0$	$\longrightarrow$	$0^4\mathbf{0}0^4\mathbf{1}0^4\mathbf{1}0^i$
0111	$\longrightarrow$	0^i
$\mathbf{100}0$	$\longrightarrow$	$0^4\mathbf{1}0^4\mathbf{0}0^4\mathbf{0}0^i$
$\mathbf{10}01$	$\longrightarrow$	$0^4\mathbf{1}0^4\mathbf{0}0^i$
$\mathbf{101}0$	$\longrightarrow$	$0^4\mathbf{1}0^4\mathbf{0}0^4\mathbf{1}0^i$
$\mathbf{1}011$	$\longrightarrow$	$0^4\mathbf{1}0^i$
$\mathbf{110}0$	$\longrightarrow$	$0^4\mathbf{1}0^4\mathbf{1}0^4\mathbf{0}0^i$
$\mathbf{11}01$	$\longrightarrow$	$0^4\mathbf{1}0^4\mathbf{1}0^i$
$\mathbf{111}0$	$\longrightarrow$	$0^4\mathbf{1}0^4\mathbf{1}0^4\mathbf{1}0^i$

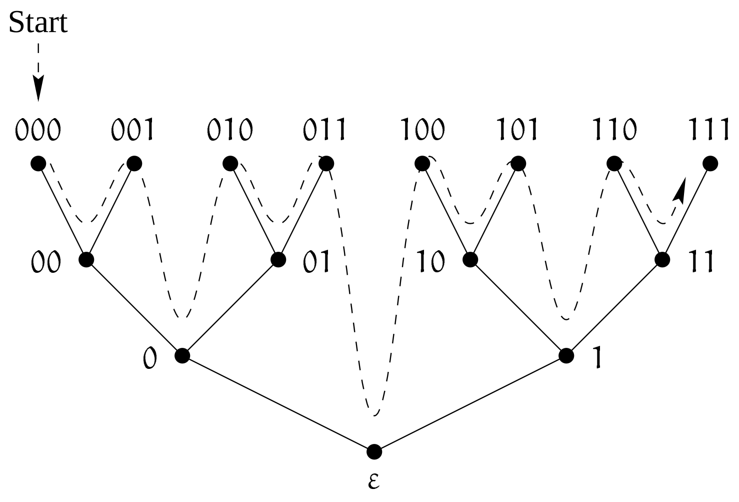


Figure 4.1: *Encoding function* $\alpha : (\{0,1\}^n \setminus \{1^n\}) \times \{0\}^{\leq l} \to \{0,1\}^*$ *defined by* $\alpha(a_1 \ldots a_m 0 1^{n-m-1}, 0^i) =_{\text{df}} 0^l a_1 0^l a_2 \ldots 0^l a_m 0^i$, *where* $0 \leq m < n$ *and* $a_1, \ldots, a_m \in \{0,1\}$. *The table shows an example for* $n = l = 4$ *with the* a_i*'s being represented in boldface. The tree gives the corresponding encodings words removing the substrings* 0^4 *and the* 0^i *whose order is represented by the doted line. The substrings* 0^l *were introduced in the function* α *due to the parameter* 0^i.

The next result shows that for $\mathcal{K}_P(\sigma[r]\,,1)$ a single type 2 quantifier is probably more powerful than a type 1 quantifier when more than two queries are allowed.

Theorem 4.11. *Let* $r : \mathbb{N} \to \mathbb{N}$. *Then*

(1) $\mathcal{K}_P(2[1]\,,1) =$ co*NP;*

(2) $\mathcal{K}_P(2[2]\,,1) =$ *PSPACE;*

(3) $\mathcal{K}_P(2[r]\,,1) =$ *NEXPTIME for* $r \geq 3$.

Proof. The first statement follows by Theorem 4.10 and the last statement follows by Theorems 4.8 and 3.13. Next we prove the second statement. By Theorem 4.10 and Lemma 4.3 follows *PSPACE* $\subseteq \mathcal{K}_P(1[2]\,,1) \subseteq \mathcal{K}_P(2[2]\,,1)$. Thus, it remains to prove $\mathcal{K}_P(2[2]\,,1) \subseteq$ *PSPACE*. For $a \in \{0,1\}$ let $a^1 =_{\text{df}} a$ and $a^0 =_{\text{df}} 1 - a$, and for a set U let $U^1 =_{\text{df}} U$ and $U^0 =_{\text{df}} \overline{U}$. For $L \in \exists^2[2]\,\forall^p P$ there exist polynomial-time computable functions f, g_1, g_2 and polynomial p such that f is a 0-1-function ($g_i \approx$ the i-th query to oracle) and

$$\begin{aligned} x \in L &\iff \exists U \,\forall u \Big(|u| = p(|x|) \to \bigvee_{\substack{a,b \in \{0,1\} \\ f(x,u,a,b)=1}} (g_1(x,u) \in U^a \wedge g_2(x,u,a) \in U^b) \Big) \\ &\iff \text{the boolean formula } \bigwedge_{|u|=p(|x|)} \bigvee_{\substack{a,b \in \{0,1\} \\ f(x,u,a,b)=1}} (z^a_{g_1(x,u)} \wedge z^b_{g_2(x,u,a)}) \text{ with the} \\ &\qquad \text{variables } z_0, \dots, z_{2^{q(|x|)}} \text{ is satisfiable (q suitable polynomial)} \end{aligned}$$

A simple computation shows that the formula $\bigvee_{\substack{a,b \in \{0,1\} \\ f(x,u,a,b)=1}} (z^a_{g_1(x,u)} \wedge z^b_{g_2(x,u,a)})$ is equivalent to a conjunction of clauses with at most two literals each, though the formula has three variables. (An easier way to see that is to use Lemma 4.14 which shows that we can without loss of generality assume that the queries are made in parallel. In this case the formula has two variables at all and hence its conjunctive normal form has only clauses with at most two literals.)

Hence, L can be m-reduced to the 2-SAT problem using polynomial space. Because 2-SAT $\in$ *NL* (see [Pap94, p. 185]) we obtain $L \in$ *PSPACE.* ❑

Thus, we have characterized the classes of the bounded analytic polynomial-time hierarchy having the form $\mathcal{K}_P(\sigma[r]\,, 1)$. To reach our goal in this section it remains to consider the classes containing at least two set quantifiers followed by one word quantifier. This will be done in two steps. First we consider the classes where the last set quantifier is not restricted to one query.

Theorem 4.12. *Let* $k \geq 1$, $\sigma_1, \dots, \sigma_k, \sigma \in \{1,2\}$ *and* $r_1, \dots, r_k, r : \mathbb{N} \to \mathbb{N}_+$. *Then*

(1) $\mathcal{K}_P(\sigma_1[r_1] \dots \sigma_k[r_k]\, \sigma[2]\,, 1) = \Sigma^{\text{exp}}_k$*;*

(2) $\mathcal{K}_P(\sigma_1[r_1] \dots \sigma_k[r_k]\, 1[r]\,, 1) = \Sigma^{\text{exp}}_k$ *for* $r \geq 3$*;*

(3) $\mathcal{K}_P(\sigma_1[r_1] \dots \sigma_k[r_k]\, 2[r]\,, 1) = \Sigma^{\text{exp}}_{k+1}$ *for* $r \geq 3$.

Proof. The last statement follows by Theorems 4.8 and 3.13. For the second statement, we conclude

$$\begin{aligned} \mathcal{K}_P(\sigma_1[r_1] \dots \sigma_k[r_k]\, 1[r]\,, 1) \subseteq \mathcal{K}_P\big(2^k 1, 1\big) \subseteq \Sigma^{\text{exp}}_k & \qquad \text{by Theorem 3.13} \\ \subseteq \mathcal{K}_P(\sigma_1[r_1] \dots \sigma_k[r_k]\, 1[2]\,, 1) & \qquad \text{by Statement (1)} \\ \subseteq \mathcal{K}_P(\sigma_1[r_1] \dots \sigma_k[r_k]\, 1[r]\,, 1) & \end{aligned}$$

Now, we prove the first statement. It is enough to show $\mathcal{K}_P\big(2^k 2[2]\,, 1\big) \subseteq \Sigma^{\text{exp}}_k$ and $\Sigma^{\text{exp}}_k \subseteq \mathcal{K}_P\big((1[1])^k 1[2]\,, 1\big)$.

"$\mathcal{K}_P\big(2^k 2[2]\,, 1\big) \subseteq \Sigma^{\text{exp}}_k$": Let k be even. Note that the proof of $\mathcal{K}_P(2[2]\,, 1) \subseteq$ *PSPACE* (Theorem 4.11) remains valid if the machines have additionally k oracles of type 2, i.e. $\exists^2[2]\,\forall^p P^{0 2^k 2[2] 0} \subseteq \mathit{PSPACE}^{0 2^k}$. Hence $\exists^2\forall^2 \dots \forall^2\exists^2[2]\,\forall^p P \subseteq \exists^2\forall^2 \dots \forall^2 \mathit{PSPACE}^{0 2^k}$

which is included in $\Sigma_k^{\exp}$ by Proposition 3.12. The case of odd k is treated analogously on the base of co$\mathcal{K}_P(2[2]\,,1) \subseteq$ *PSPACE*.

"$\Sigma_k^{\exp} \subseteq \mathcal{K}_P\big((1[1])^k 1[2]\,,1\big)$": By Theorem 4.8 we obtain $\Sigma_k^{\exp} \subseteq \mathcal{K}_P\big((1[1])^k, 2\big)$. Hence, it is easy to see that we have to prove only $\exists^p\forall^p P^{0(1[1])^k00} \subseteq \exists^1[2]\,\forall^p P^{0(1[1])^k1[2]0}$. For a language $L \in \exists^p\forall^p P^{0(1[1])^k00}$ there exist an $L_1 \in P^{0(1[1])^k00}$ and polynomials q and r such that

$$X \in L \Longleftrightarrow \exists^p u\,\forall^p w\,(|u| = q(|X|) \wedge (|w| = r(|X|) \rightarrow (X,u,w) \in L_1))$$

For $a \in \{0,1,2\}$, $u \in \{0,1\}^{q(|X|)}$ and $w \in \{0,1\}^{r(|X|)}$ define the function f as follows

$$f(X,uw)(a) =_{\mathrm{df}} \begin{cases} 1 & \text{if } a = 1, \\ 2 & \text{if } a = 0 \text{ and } w = 0^{r(|X|)} \text{ and } (X,u,w) \in L_1, \\ 2 & \text{if } a = 2 \text{ and } w \neq 1^{r(|X|)} \text{ and } (X,u,w) \in L_1, \\ 1 & \text{if } a = 2 \text{ and } w = 1^{r(|X|)} \text{ and } (X,u,w) \in L_1, \\ 0 & \text{otherwise.} \end{cases}$$

Obviously, f is computable in polynomial-time and

$$X \in L \Longleftrightarrow f(X, 1^{p(|X|)}) \circ f(X, 1^{p(|X|)-1}0) \circ \ldots \circ f(X, 0^{p(|X|)})(0) = 1$$

where $p(|X|) =_{\mathrm{df}} q(|X|) + r(|X|)$ and $\circ$ is the traditional composition of functions. Now, the inclusion $\exists^p\forall^p P^{0(1[1])^k00} \subseteq \exists^1[2]\,\forall^p P^{0(1[1])^k1[2]0}$ follows as the inclusion *PSPACE* $\subseteq \exists^1[2]\,\forall^p P$ proved in Theorem 4.10.

$$\begin{aligned} X \in L \Longleftrightarrow \exists g\Bigg(&(g:\{0,1\}^* \rightarrow \{0,1,2\}) \wedge \forall u\Bigg(|u| = p(|X|) \rightarrow \\ &\qquad \Big(\Big(u = 0^{p(|X|)} \rightarrow f\,(X,u)\,(0) = g\,(u)\Big) \\ &\qquad \wedge \Big(u \notin \big\{0^{p(|X|)}, 1^{p(|X|)}\big\} \rightarrow f\,(X,u)\,(g(\overleftrightarrow{u})) = g\,(u)\Big) \\ &\qquad \wedge \Big(u = 1^{p(|X|)} \rightarrow f\,(X,u)\,(g(\overleftrightarrow{u})) = 1\Big)\Big)\Bigg)\Bigg) \\ \Longleftrightarrow \exists^2 U\,\forall^p u\,\forall^p i\,\forall^p j\Bigg(&|u| = p(|X|) \wedge 0 \leq i,j \leq 2 \rightarrow \\ &\Big(\Big(u = 0^{p(|X|)} \rightarrow (f\,(X,u)\,(0) = i \leftrightarrow u0^i \in U)\Big) \\ &\wedge \Big(u \notin \big\{0^{p(|X|)}, 1^{p(|X|)}\big\} \wedge \overleftrightarrow{u}0^i \in U \rightarrow (f\,(X,u)\,(i) = j \leftrightarrow u0^j \in U)\Big) \\ &\wedge \Big(u = 1^{p(|X|)} \wedge \overleftrightarrow{u}0^i \in U \rightarrow f\,(X,u)\,(i) = 1\Big)\Big)\Bigg) \end{aligned}$$

However, for each (X,U,u,i,j) such that $u \notin \{0^{p(|X|)}, 1^{p(|X|)}\}$ the two queries $\overleftrightarrow{u}0^i$ and $u0^j$ are asked to U which is not a type 1 querying. To overcome this difficulty we encode the words from $(\{0,1\}^{p(|X|)} \setminus \{1^{p(|X|)}\}) \times \{0\}^{\leq 2}$ by an injective function α which has the property that, for every $u \notin \{0^{p(|X|)}, 1^{p(|X|)}\}$ and $0 \leq i,j \leq 2$, either $\alpha(\overleftrightarrow{u}, 0^i)$ is an initial word of $\alpha(u, 0^j)$ or vice versa (take for example the function α of Figure 4.1 with $n = p(|X|)$ and $l = 2$). Now instead of querying $\overleftrightarrow{u}0^i$ and $u0^j$ to U the queries $\alpha(\overleftrightarrow{u}, 0^i)$ and $\alpha(u, 0^j)$ are made to the oracle $V =_{\text{df}} \left\{\alpha(u, 0^i) : 0 \leq i \leq 2 \wedge u0^i \in U\right\}$ in a type 1 manner. Therefore,

$$X \in L \iff \exists^1 V \,\forall^p u \,\forall^p i \,\forall^p j \Big((|u| = p(|X|) \wedge 0 \leq i,j \leq 2) \to (X, V, u, i, j) \in L_1 \Big)$$

where $(X, V, u, i, j) \in L_1 \iff_{\text{df}}$

$$\begin{aligned}
&\Big(u = 0^{p(|X|)} \to (f(X,u)(0) = i \leftrightarrow \alpha(u, 0^i) \in V)\Big)\\
\wedge&\Big(u \notin \left\{0^{p(|X|)}, 1^{p(|X|)}\right\} \wedge \alpha(\overleftrightarrow{u}, 0^i) \in V \to (f(X,u)(i) = j \leftrightarrow \alpha(u, 0^j) \in V)\Big)\\
\wedge&\Big(u = 1^{p(|X|)} \wedge \alpha(\overleftrightarrow{u}, 0^i) \in V \to f(X,u)(i) = 1\Big)
\end{aligned}$$

Obviously, L_1 can be accepted by a deterministic polynomial-time oracle Turing machine which, on input (X, V, u, i, j), queries the oracle V in type 1 manner and at most two times. Hence, $L_1 \in \mathcal{P}^{0(1[1])^k 1[2]000}$ and $L \in \exists^1[2]\,\forall^p\forall^p\forall^p \mathcal{P}^{0(1[1])^k 1[2]000}$. Now, using Lemma 4.4 we conclude $L \in \exists^1[2]\,\forall^p \mathcal{P}^{0(1[1])^k 1[2]0}$. ❑

Now we consider the case at least two set quantifiers where the last one is restricted to one query.

Theorem 4.13. *Let* $\sigma_1, \ldots, \sigma_k, \sigma, \tau \in \{1,2\}$ *and* $r_1, \ldots, r_k, r : \mathbb{N} \to \mathbb{N}_+$. *Then*

(1) $\mathcal{K}_{\mathcal{P}}(\sigma[r]\,\tau[1]\,, 1) =$ *NP;*

(2) $\mathcal{K}_{\mathcal{P}}(\sigma_1[1]\,\sigma[r]\,\tau[1]\,, 1) =$ *PSPACE;*

(3) $\mathcal{K}_{\mathcal{P}}(\sigma_1[r_1] \ldots \sigma_{k-1}[r_{k-1}]\,\sigma_k[1]\,\sigma[r]\,\tau[1]\,, 1) = \Sigma^{\text{exp}}_{k-1}$ *for* $k \geq 2$*;*

(4) $\mathcal{K}_{\mathcal{P}}(\sigma_1[r_1] \ldots \sigma_k[r_k]\,\sigma[r]\,\tau[1]\,, 1) = \Sigma^{\text{exp}}_{k}$ *for* $k \geq 1$ *and* $r_k \geq 2$.

Proof. For $l \geq 1$, let $Q_l =_{\text{df}} \exists$ if l is odd and $Q_l =_{\text{df}} \forall$ otherwise. In the proof of $\exists^\tau[1]\,\forall^p \mathcal{P} \subseteq \forall^p \mathcal{P}$ (Theorems 4.10 and 4.11) we need to simulate only two times the original machine for each word uvw. Therefore, if the machines have additional oracles, the number of queries to each one of these additional oracles is doubled and the type 1 property cannot be guaranteed. Thus, for $k \geq 0$ (if k is even, take the base $\forall^\tau[1]\,\exists^p \mathcal{P} \subseteq \exists^p \mathcal{P}$)

$$\begin{aligned}
\exists^{\sigma_1}[r_1] \ldots Q^{\sigma_k}_k[r_k]\, Q^{\sigma}_{k+1}[r]\, Q^{\tau}_{k+2}[1]\, Q^{p}_{k+3}\mathcal{P} &\subseteq \exists^2[2r_1] \ldots Q^2_k[2r_k]\, Q^2_{k+1}[2r]\, Q^p_{k+3}\mathcal{P}\\
&\subseteq \exists^2[2r_1] \ldots Q^2_k[2r_k]\, Q^p_{k+1}\mathcal{P}
\end{aligned} \tag{4.1}$$

since $Q_{k+1} = Q_{k+3}$ and by Lemmas 4.4 and 4.7 we can melt these last 2 quantifiers to one word quantifier. Hence we conclude

Statement (1):

$$\mathcal{K}_P(\sigma[r]\,\tau[1]\,,1) \subseteq \mathcal{K}_P(\varepsilon,1) \subseteq NP \subseteq \mathcal{K}_P(\sigma[r]\,\tau[1]\,,1) \qquad \text{by Equation (4.1)}$$

Statement (2):

$$\begin{aligned}\mathcal{K}_P(\sigma_1[1]\,\sigma[r]\,\tau[1]\,,1) &\subseteq \mathcal{K}_P(2[2]\,,1) && \text{by Equation (4.1)}\\ &\subseteq PSPACE \subseteq \mathcal{K}_P(1[2]\,,1) && \text{by Theorems 4.11 and 4.10}\\ &\subseteq \mathcal{K}_P\big((1[1])^3,1\big) && \text{by Lemmas 4.5 and 4.3}\\ &\subseteq \mathcal{K}_P(\sigma_1[1]\,\sigma[r]\,\tau[1]\,,1)\end{aligned}$$

Statement (3):

$$\begin{aligned}&\mathcal{K}_P(\sigma_1[r_1]\dots\sigma_{k-1}[r_{k-1}]\,\sigma_k[1]\,\sigma[r]\,\tau[1]\,,1)\\ &\quad\subseteq \mathcal{K}_P(2[2r_1]\dots2[2r_{k-1}]\,2[2]\,,1) && \text{by Equation (4.1)}\\ &\quad\subseteq \Sigma^{\exp}_{k-1} \subseteq \mathcal{K}_P\big((1[1])^{k-1}1[2]\,,1\big) && \text{by Theorem 4.12}\\ &\quad\subseteq \mathcal{K}_P\big((1[1])^{k+2},1\big) && \text{by Lemmas 4.5 and 4.3}\\ &\quad\subseteq \mathcal{K}_P(\sigma_1[r_1]\dots\sigma_{k-1}[r_{k-1}]\,\sigma_k[1]\,\sigma[r]\,\tau[1]\,,1)\end{aligned}$$

Statement (4):

$$\begin{aligned}&\mathcal{K}_P(\sigma_1[r_1]\dots\sigma_k[r_k]\,\sigma[r]\,\tau[1]\,,1)\\ &\quad\subseteq \mathcal{K}_P(2[2r_1]\dots2[2r_k]\,,1) && \text{by Equation (4.1)}\\ &\quad\subseteq \Sigma^{\exp}_{k} \subseteq \mathcal{K}_P\big((1[1])^{k-1}2[3]\,,1\big) && \text{by Theorems 3.13 and 4.8}\\ &\quad\subseteq \mathcal{K}_P\big((1[1])^{k-1}1[2]\,1[1]\,1[1]\,,1\big) && \text{by Lemmas 4.5 and 4.3}\\ &\quad\subseteq \mathcal{K}_P(\sigma_1[r_1]\dots\sigma_k[r_k]\,\sigma[r]\,\tau[1]\,,1)\end{aligned}$$

❑

4.4.2 Parallel Queries

Now, we consider oracle machines that during their computations ask the queries in a parallel manner, i.e. they compute a list of all queries before asking one of them. We will represent the parallel restriction over an oracle bounded by set quantifier $Q^\sigma[r]$ by $Q^\sigma[\|r]$, where $Q \in \{\exists,\forall\}$, $\sigma \in \{1,2\}$ and $r : \mathbb{N} \to \mathbb{N}$. The following lemma shows equivalences between classes involving and not involving parallel queries.

Lemma 4.14. *Let* $\mu \in \Phi^*$. *Then, for* $k \in \mathbb{N}$ *and* $\sigma \in \{1,2\}$

$$\exists^\sigma[\|k]\,\forall^p \boldsymbol{P}^{\mu\sigma[\|k]0} = \exists^\sigma[k]\,\forall^p \boldsymbol{P}^{\mu\sigma[k]0}$$

Proof. The direction "$\subseteq$" is evident. For the other direction let $k \geq 2$ (for $k \in \{0,1\}$ it is obvious) and $L \in \exists^\sigma[k]\,\forall^p \boldsymbol{P}^{\mu\sigma[k]0}$. There exist an $L' \in \boldsymbol{P}^{\mu\sigma[k]0}$ and a polynomial p such that

$$X \in L \iff \exists^\sigma U\,\forall^p v\,(|v| = p(|X|) \to (X,U,v) \in L')$$

Let M be a polynomial-time machine of type $\mu\sigma[k]\,0$ accepting L' and, let $f, g_1, \ldots, g_k$ be functions computable in polynomial-time such that for M on input (X, U, v) and for $i = 1, \ldots, k$, $g_i(X, v, u_1, \ldots, u_{i-1})$ is the i-th query of M to U assumed the answers to the $i-1$ first queries were $u_1, u_2, \ldots, u_{i-1}$, and $f(X, v, u_1, \ldots, u_k)$ is the result of M on input (X, U, v) if the answers of U to the k queries are $u_1, \ldots, u_k$. Hence,

$$\begin{aligned}
X \in L \Leftrightarrow \exists U\,\forall v\Big(\,|v| = p(|X|) \to \bigvee_{|u|=k} \Big(&(g_1(X,v) \in U \leftrightarrow u(1) = 1) \wedge \cdots \wedge \\
&\wedge (g_k(X, v, u(1), \ldots, u(k-1)) \in U \leftrightarrow u(k) = 1) \\
&\wedge f(X, v, u(1), \ldots, u(k)) = 1\Big)\Big) \\
\Leftrightarrow \exists U\,\forall v\Big(\,|v| = p(|X|) \to \bigwedge_{|u|=k} \Big(&(g_1(X,v) \in U \leftrightarrow u(1) = 0) \vee \cdots \vee \\
&\vee (g_k(X, v, u(1), \ldots, u(k-1)) \in U \leftrightarrow u(k) = 0) \\
&\vee f(X, v, u(1), \ldots, u(k)) = 1\Big)\Big) \\
\Leftrightarrow \exists U\,\forall v\,\forall u\Big((|v| = p(|X|) \wedge |u| = k) \to (X, U, v, u) \in L_1\Big)&
\end{aligned}$$

where $(X, U, v, u) \in L_1 \iff_{\text{df}} |u| = k \wedge$

$$\Big((g_1(X,v) \in U \leftrightarrow u(1) = 0) \vee \cdots \vee (g_k(X, v, u(1), \ldots, u(k-1)) \in U \leftrightarrow u(k) = 0)$$
$$\vee f(X, v, u(1), \ldots, u(k)) = 1\Big)$$

Obviously, a deterministic polynomial-time machine of type $\mu\sigma[k]\,00$ can accept L_1 asking the k queries to the oracle bounded by the last set quantifier in parallel. Hence, $L_1 \in P^{\mu\sigma[\|k]00}$ and $L \in \exists^\sigma[\|k]\,\forall^p\forall^p P^{\mu\sigma[\|k]00}$. Then $L \in \exists^\sigma[\|k]\,\forall^p P^{\mu\sigma[\|k]0}$ follows by Lemma 4.4. ❑

All the characterizations obtained in §4.4.1 remain valid for the counterpart classes $\mathcal{K}_P(\cdot)$ having the parallel queries restriction. This can be seen as follows: For the Theorem 4.9 it is evident, since only one query for each oracle is enough. For the other results with exception Theorem 4.13, only the number of queries to the oracle bounded by the last set quantifier is relevant (for each one of the remainder oracles only one query is enough). Hence and by Lemma 4.14 we have the desired characterization. For Theorem 4.13, only in the proof of Statement (4) can arise problems. However, observe that the proof of Theorem 3.11 yields $\Sigma_k^{\exp} \subseteq \mathcal{K}_P\big((1\,[1])^{k-1}2[\|3]\,,1\big)$ and that the rules of Lemma 4.5 remain valid if we consider the parallel restriction, i.e. $\exists^2[\|r] \to \exists^1[1]\,\exists^1[\|(r-1)]\,\forall^p$ and $\forall^2[\|r] \to \forall^1[1]\,\forall^1[\|(r-1)]\,\exists^p$.

4.5 Remainder Complexity Classes

In §3.2.2 was proved a Normal Form Theorem (Theorem 3.8) for the classes of the analytic polynomial-time hierarchy. It was shown that each class of this hierarchy coincides with a class in the form $\mathcal{K}_{P}(\cdot)$ or co$\mathcal{K}_{P}(\cdot)$. Unfortunately, the equivalence rules used in the proof of this theorem do not preserve the number of oracle queries and cannot be applied to a chain of quantifiers of a class of the bounded analytic polynomial-time hierarchy. Otherwise, using the results of §4.4 we would obtain a complete characterization for the classes of the bounded analytic polynomial-time hierarchy. Thus, in this section we consider classes of the bounded analytic polynomial-time hierarchy do not having necessarily the form $\mathcal{K}_{P}(\cdot)$ or co$\mathcal{K}_{P}(\cdot)$.

By equivalence rule $\exists^p\exists^p \leftrightarrow \exists^p$ ($\forall^p\forall^p \leftrightarrow \forall^p$) of Lemma 4.4, we can melt adjacent existential (universal, respectively) word quantifiers. Hence and by obvious rule $\exists^p\exists^\sigma[r] \leftrightarrow \exists^\sigma[r]\,\exists^p$ ($\forall^p\forall^\sigma[r] \leftrightarrow \forall^\sigma[r]\,\forall^p$), in what follows we will assume that in every consecutive sequence of existential (universal, respectively) quantifiers of a quantifier string there exists at most one word quantifier. Furthermore, we will state and prove only the results for $\mathrm{Q}P$ with $\mathrm{Q} \in \Gamma^*_{[p]}$, since the results for co$\mathrm{Q}P$ are immediate consequences. Finally, let us state again the fact (§4.2.2) that for Q^τ it is equivalent to take $\mathrm{Q}^{\tau[r]}$ with r being so large that it is not a real restriction and vice versa, where $\mathrm{Q} \in \{\exists, \forall\}$, $\tau \in \{1, 2\}$ and $r : \mathbb{N} \to \mathbb{N}_+$.

For $S \in \Gamma^*_{[p]}$ and $T \in \Gamma^+_{[p,Q]}$, where $\mathrm{Q} \in \{\exists, \forall\}$ and the quantifier Q^p, Q^1 or Q^2 appears in the chain T, we gave a complete characterization of the classes STP by well-known complexity classes (§4.5.1). Then, other classes are considered and the open cases are stated (§4.5.2). We show in addition that all the results obtained remain valid if the oracle machines are allowed to make only parallel queries (§4.5.3).

4.5.1 Characterizing the Classes STP

Next, we characterize the classes of the bounded analytic polynomial-time hierarchy having the form STP by well-known complexity classes for $S \in \Gamma^*_{[p]}$ and $T \in \Gamma^+_{[p,Q]}$, where $\mathrm{Q} \in \{\exists, \forall\}$ and the quantifier Q^p, Q^1 or Q^2 appears in the chain T. Note that the classes examined in §4.4 are special cases of STP with T being a word quantifier, i.e. $T = \mathrm{Q}^p$. Since each one of the classes of the bounded analytic polynomial-time hierarchy not containing set quantifiers coincides with one class of the (arithmetic) polynomial-time hierarchy [SM73, Sto77, Wra77], it remains to consider the classes STP containing at least one set quantifier. It will turn out that each one of the these classes involving set quantifiers coincides with one of the classes Σ^p_k, Π^p_k ($k \geq 1$), *PSPACE*, Σ^{exp}_k or Π^{exp}_k ($k \geq 1$) and vice versa. Our first result says that it is enough to consider the classes $S\mathrm{Q}^pP$.

Proposition 4.15. *Let* $\mathrm{Q} \in \{\exists, \forall\}$, $S \in \Gamma^*_{[p]}$ *and* $T \in \Gamma^+_{[p,Q]}$. *If the quantifier* Q^p, Q^1 *or* Q^2 *appears in the chain* T, *then* $STP = S\mathrm{Q}^pP$.

Proof. By Lemmas 4.3, 4.4 and 4.7 we conclude $STP \subseteq S\mathrm{Q}^2P \subseteq S\mathrm{Q}^pP \subseteq STP$. ❑

Next, a desired form for a quantifier string is defined. Let $\sigma, \tau \in \{1, 2\}$ and $r, r' : \mathbb{N} \to \mathbb{N}$. For a chain $S \in \Gamma^*_{[p]}$, let $\mathrm{Norm}(S)$ be an alternate sequence of $\exists$-$\forall$-quantifiers with the set quantifiers appearing left to the word quantifiers, which is obtained applying the following rules over S, where the melt rules have precedence over the shift rules, i.e. a shift rule is applied if and only if no meld rule can be applied:

(1) Melt rules (Lemmas 4.3 and 4.4):

$$\begin{array}{rcl}
\exists^p\exists^p \rightarrow_{[P]} \exists^p & \text{and} & \forall^p\forall^p \rightarrow_{[P]} \forall^p \\
\exists^p\exists^\sigma[r] \rightarrow_{[P]} \exists^\sigma & \text{and} & \forall^p\forall^\sigma[r] \rightarrow_{[P]} \forall^\sigma \\
\exists^\sigma[r]\,\exists^p \rightarrow_{[P]} \exists^\sigma & \text{and} & \forall^\sigma[r]\,\forall^p \rightarrow_{[P]} \forall^\sigma \\
\exists^\sigma[r]\,\exists^\tau[r'] \rightarrow_{[P]} \exists^2[r+r'] & \text{and} & \forall^\sigma[r]\,\forall^\tau[r'] \rightarrow_{[P]} \forall^2[r+r']
\end{array}$$

(2) Shift rules (Lemma 4.6): $\exists^p\forall^\sigma[r] \rightarrow_{[P]} \forall^\sigma[r]\,\exists^p$ and $\forall^p\exists^\sigma[r] \rightarrow_{[P]} \exists^\sigma[r]\,\forall^p$

Obviously, for $R, S, T \in \Gamma^*_{[p]}$, we have $RST\mathcal{P} \subseteq R\,\mathrm{Norm}(S)\,T\mathcal{P}$ and $\mathrm{Norm}(S)\,\mathcal{P}$ is a class in the form $\mathcal{K}_{\mathcal{P}}(\cdot)$ or $\mathrm{co}\mathcal{K}_{\mathcal{P}}(\cdot)$. Finally, let $\mathrm{FirstSet}(S)$ be $\exists$ ($\forall$) if the first set quantifier appearing in S is an existential (universal, respectively) quantifier ($\mathrm{FirstSet}(S) =_{\mathrm{df}} \varepsilon$ if S does not contain set quantifiers).

The following theorem shows which classes ending in an $\exists$-$\forall$-alternate sequence of word quantifiers coincide with a class of the exponential-time alternation hierarchy.

Theorem 4.16. *Let $S \in \Gamma^+_{[p]}$ with $\mathrm{FirstSet}(S) = \exists$, and $Q \in \{\exists, \forall\}$ be the last quantifier in S. If k is the number of set quantifiers in $\mathrm{Norm}(S)$, then $S\overline{Q}^pQ^p\mathcal{P} = \Sigma^{\mathrm{exp}}_k$.*

Proof. We conclude $S\overline{Q}^pQ^p\mathcal{P} \subseteq \mathrm{Norm}(S)\,\overline{Q}^pQ^p\mathcal{P} \subseteq \mathrm{Norm}\big(S\overline{Q}^pQ^p\big)\,\mathcal{P}$ which is included in the class $\mathcal{K}_{\mathcal{P}}\big((1[1])^k, 2\big) = \Sigma^{\mathrm{exp}}_k$ by Theorem 4.9, and $\mathcal{K}_{\mathcal{P}}\big((1[1])^k, 2\big) \subseteq S\overline{Q}^pQ^p\mathcal{P}$ is obvious. ❑

Before the remainder characterizations are presented, we state two results to help us in their proofs. The next proposition shows a result which was already observed in the proof of Theorem 4.13. For $S \in \Gamma^*_{[p]}$, let $(S)_2$ be the chain S but: If $Q^\sigma[r]$ appears in S then in $(S)_2$ we have $Q^2[2r]$ for $\sigma \in \{1, 2\}$, $Q \in \{\exists, \forall\}$ and $r : \mathbb{N} \rightarrow \mathbb{N}$, i.e. all set quantifiers from S have type 2 in $(S)_2$ and if a set quantifier varies over oracles whose number of queries is bounded by function r, then this quantifier in $(S)_2$ varies over oracles whose number of queries is bounded by function $2r$.

Proposition 4.17. *Let $\sigma \in \{1, 2\}$, $Q \in \{\exists, \forall\}$ and $S \in \Gamma^*_{[p]}$. Then $SQ^\sigma[1]\,\overline{Q}^p\mathcal{P} \subseteq (S)_2\,\overline{Q}^p\mathcal{P}$.*

Proof. In the proof of $\exists^\sigma[1]\,\forall^p\mathcal{P} \subseteq \forall^p\mathcal{P}$ (Theorems 4.10 and 4.11) we need to simulate only two times the original machine for each word uvw (if $Q = \forall$, take the base $\forall^\sigma[1]\,\exists^p\mathcal{P} \subseteq \exists^p\mathcal{P}$). Therefore, if the machines have additional oracles, the number of queries to each one of these additional oracles is doubled and the type 1 property cannot be guaranteed. Thus, $SQ^\sigma[1]\,\overline{Q}^p\mathcal{P} \subseteq (S)_2\,\overline{Q}^p\mathcal{P}$. ❑

The following result states an "equivalence rule" which shows how to eliminate word quantifiers.

Proposition 4.18. *Let $k, l \in \mathbb{N}$ and $Y \in \{\exists^p, \forall^p\}^l$. Then, for $Q \in \{\exists, \forall\}$*

$$YQ^2[2]\,\overline{Q}^p\mathcal{P}^{02^k0^l2[2]0} = Q^2[2]\,\overline{Q}^p\mathcal{P}^{02^k2[2]0} = \mathit{PSPACE}^{02^k}$$

Proof. The proof of $\exists^2[2]\,\forall^p P = \mathit{PSPACE}$ (Theorem 4.11) remains valid if the machines have additionally k inputs of type 2 and l inputs of type 0 (if $Q = \forall$, take the base $\forall^2[2]\,\exists^p P = \mathit{PSPACE}$), i.e. $\exists^2[2]\,\forall^p P^{02^k0^l2[2]0} = \mathit{PSPACE}^{02^k0^l}$. Furthermore, $Y\,\mathit{PSPACE}^{02^k0^l} \subseteq \mathit{PSPACE}^{02^k}$, since the Y word quantifiers can be easily simulate by a *PSPACE*-simulation. Hence,

$$\begin{aligned} YQ^2[2]\,\overline{Q}^p P^{02^k0^l2[2]0} &\subseteq Y\,\mathit{PSPACE}^{02^k0^l} \\ &\subseteq \mathit{PSPACE}^{02^k} \subseteq Q^2[2]\,\overline{Q}^p P^{02^k2[2]0} \\ &\subseteq YQ^2[2]\,\overline{Q}^p P^{02^k0^l2[2]0} \end{aligned}$$ ❑

Characterizing the classes $S\overline{Q}^p P$

By Proposition 4.15 and Theorem 4.16 it remains to consider the classes $S\overline{Q}^p P$, where

- $S \in \Gamma^+_{[p]}$;
- $Q \in \{\exists, \forall\}$ is the last quantifier in S;
- in the last consecutive sequence of Q-quantifiers in S there exists at least one set quantifier, i.e. after the last set quantifier in S does not appear the substring $\overline{Q}^p Q^p$ and;
- in each consecutive sequence of existential (universal, respectively) quantifiers in the chain S there exists at most one word quantifier.

We will divide the study of the $S\overline{Q}^p P$ classes depending on chain S. Let R and T be quantifier strings such that $RT = S$ with $R = \varepsilon$ or the last quantifier of R being a set $\overline{Q}$-quantifier. Hence, T contains at least one set Q-quantifier. The Table 4.2 shows where the characterizations of the $RT\overline{Q}^p P$ classes, i.e. $S\overline{Q}^p P$ classes, by well-known complexity classes can be found.

Theorem(s)	Chain R	Chain T
4.19, 4.20, 4.21	$R = \varepsilon$	$T \in \left(\Gamma_{[p,Q]} \cup \left\{\overline{Q}^p\right\}\right)^+$
4.22	$R \in \Gamma^+_{[p]}$	$T \in \left(\Gamma_{[p,Q]} \cup \left\{\overline{Q}^p\right\}\right)^+$ such that in T appears a set quantifier that differs from $Q^1[1]$ and $Q^2[1]$
4.23, 4.24, 4.25	$R \in \Gamma^+_{[p]}$	$T \in \left\{Q^1[1]\,, Q^2[1]\,, \exists^p, \forall^p\right\}^+$

Table 4.2: *Characterizations of the classes $RT\overline{Q}^p P$ with $R = \varepsilon$ or the last quantifier of R being a set $\overline{Q}$-quantifier.*

A function that enable us to specify quantifier strings containing desired properties will be helpful to us: For $k, l \geq 0$ and $Q_1, \ldots, Q_k, T_1, \ldots, T_l \in \Gamma_{[p]}$, we define the function $\mathcal{F}(Q_1, \ldots, Q_k\,|T_1, \ldots, T_l)$ as the set of all possible quantifier strings in $\Gamma^*_{[p]}$, such that only the quantifiers $Q_1, \dots, Q_k, T_1, \dots, T_l$ can appear and each one of the quantifiers $Q_1, \dots, Q_k$ appears once (if $Q_i = T_j$, then the quantifier Q_i must appear at least once).

The simplest classes $S\overline{Q}^p P$ are those with all set quantifiers being $\exists$-quantifiers ($\forall$-quantifiers, respectively). The next result shows that these classes containing a quantifier of type 1 which is not restricted to one query, turn out to coincide with *PSPACE* if only one set appears in S and with *NEXPTIME* or co*NEXPTIME* otherwise.

Theorem 4.19. *Let* $\sigma \in \{1, 2\}$ *and* $r, r' : \mathbb{N} \to \mathbb{N}_+$ *such that* $r \geq 2$. *Then*

(1) $S\forall^p P = \mathit{PSPACE}$ *for* $S \in \mathcal{F}\big(\exists^1[r] \,|\exists^p, \forall^p\big)$;

(2) $S\forall^p P = \mathit{NEXPTIME}$ *for* $S \in \mathcal{F}\big(\exists^1[r]\,, \exists^\sigma[r'] \,\big|\Gamma_{[p,\exists]}, \forall^p\big)$.

Proof. By Theorems 4.10 and 4.11 follows

$$\begin{aligned} S\forall^p P &\subseteq \mathrm{Norm}(S\forall^p)\, P \\ &\subseteq \begin{cases} \exists^1\forall^p P = \exists^1[2]\,\forall^p P = \mathit{PSPACE} & \text{if } S \in \mathcal{F}\big(\exists^1[r] \,|\exists^p, \forall^p\big), \\ \exists^2\forall^p P = \exists^2[3]\,\forall^p P = \mathit{NEXPTIME} & \text{if } S \in \mathcal{F}\big(\exists^1[r]\,, \exists^\sigma[r'] \,\big|\Gamma_{[p,\exists]}, \forall^p\big), \end{cases} \end{aligned}$$

which is included in $S\forall^p P$ (by Lemma 4.5 follows $\exists^2[3]\,\forall^p P \subseteq \exists^1[2]\,\exists^\sigma[1]\,\forall^p P$). ❑

The following theorem shows that a class $S\overline{Q}^p P$, whose set quantifiers are existential (universal) and one of them is of type 2 not being restricted to one query, turn out to coincide with *PSPACE* if $\exists^2[2]$ ($\forall^2[2]$) is the unique set quantifier appearing in S and with *NEXPTIME* (co*NEXPTIME*, respectively) otherwise.

Theorem 4.20. *Let* $\sigma \in \{1, 2\}$ *and* $r, r' : \mathbb{N} \to \mathbb{N}_+$ *such that* $r \geq 3$. *Then*

(1) $S\forall^p P = \mathit{PSPACE}$ *for* $S \in \mathcal{F}\big(\exists^2[2] \,|\exists^p, \forall^p\big)$;

(2) $S\forall^p P = \mathit{NEXPTIME}$ *for* $S \in \mathcal{F}\big(\exists^2[r] \,\big|\Gamma_{[p,\exists]}, \forall^p\big) \cup \mathcal{F}\big(\exists^\sigma[r']\,, \exists^2[2] \,\big|\Gamma_{[p,\exists]}, \forall^p\big)$.

Proof. We conclude for

Statement (1): By obvious rule $\exists^2[2]\,\exists^p \leftrightarrow_{[P]} \exists^p\exists^2[2]$, there exists an $\exists$-$\forall$-alternate sequence Y of word quantifiers such that $S\forall^p \to_{[P]} Y\exists^2[2]\,\forall^p$ (remember that after the set quantifier in S does not appear the substring $\forall^p\exists^p$). Thus, by Proposition 4.18 follows $S\forall^p P \subseteq Y\exists^2[2]\,\forall^p P \subseteq \mathit{PSPACE} \subseteq \exists^2[2]\,\forall^p P \subseteq S\forall^p P$.

Statement (2):

$$\begin{aligned} S\forall^p P &\subseteq \mathrm{Norm}(S\forall^p)\, P \subseteq \exists^2\forall^p P \\ &\subseteq \exists^2[3]\,\forall^p P = \mathit{NEXPTIME} && \text{by Theorem 4.11} \\ &\subseteq S\forall^p P \end{aligned}$$

where for the last inclusion we use Lemma 4.5 if $S \in \mathcal{F}\big(\exists^\sigma[r']\,, \exists^2[2] \,\big|\Gamma_{[p,\exists]}, \forall^p\big)$. ❑

Next, we point out how heavily (and nicely) the results for the classes $S\overline{Q}^p P$, whose set quantifiers are existential (universal, respectively) restricted to one query, can depend on the number of set quantifiers.

Theorem 4.21. *Let* $\sigma, \tau \in \{1,2\}$ *and* $U = \bigcup_{\rho \in \{1,2\}} \mathcal{F}(\exists^p, \exists^\rho[1] \,|\, \exists^p, \forall^p)$. *Then*

(1) $S\forall^p P = \Sigma_k^p$ *for* $S \in \mathcal{F}(\exists^\sigma[1] \,|\, \exists^p, \forall^p)$ *such that the first word quantifier in* $S\forall^p$ *is* $\exists^p$*;*

(2) $S\forall^p P = \Pi_k^p$ *for* $S \in \mathcal{F}(\exists^\sigma[1] \,|\, \exists^p, \forall^p)$ *such that the first word quantifier in* $S\forall^p$ *is* $\forall^p$*;*

(3) $S\forall^p P = PSPACE$ *for* $S \in \mathcal{F}(\exists^\sigma[1], \exists^\tau[1] \,|\, \exists^p, \forall^p)$ *such that the substring* $\forall^p T$ *does not appear after the first set quantifier in* S *for all* $T \in U$*;*

(4) $S\forall^p P = NEXPTIME$ *for* $S \in \mathcal{F}(\exists^\sigma[1], \exists^\tau[1] \,|\, \exists^p, \forall^p)$ *such that the substring* $\forall^p T$ *appears after the first set quantifier in* S *for some* $T \in U$*;*

(5) $S\forall^p P = NEXPTIME$ *for* $S \in \mathcal{F}\big(\exists^\sigma[1], \exists^\sigma[1], \exists^\tau[1] \,\big|\, \exists^1[1], \exists^2[1], \exists^p, \forall^p\big)$*;*

where $k-1$ *is the number of* $\exists$-$\forall$*-alternations of the word quantifiers in* $S\forall^p$.

Proof. We conclude for

Statements (1) and (2): We prove only Statement (1), since the other follows in the same way. By obvious rule $\exists^\sigma[1]\,\exists^p \leftrightarrow_{[P]} \exists^p\exists^\sigma[1]$, there exists an $\exists$-$\forall$-alternate sequence Y of word quantifiers such that $S\forall^p \rightarrow_{[P]} Y\exists^\sigma[1]\,\forall^p$. Thus, by Proposition 4.17 we conclude $S\forall^p P \subseteq Y\exists^\sigma[1]\,\forall^p P \subseteq Y\forall^p P \subseteq \Sigma_k^p \subseteq S\forall^p P$.

Statement (3): By assumption if an $\forall^p$ appears after the first set quantifier in S, then no $\exists^p$ follows this $\forall^p$. Therefore, by obvious rule $\exists^\sigma[1]\,\exists^p \leftrightarrow_{[P]} \exists^p\exists^\sigma[1]$ and rules of Section 4.3, there exists an $\exists$-$\forall$-alternate sequence Y of word quantifiers such that $S\forall^p \rightarrow_{[P]} Y\exists^2[2]\,\forall^p$. Hence,

$$\begin{aligned} S\forall^p P &\subseteq Y\exists^2[2]\,\forall^p P \subseteq PSPACE \subseteq \exists^2[2]\,\forall^p P && \text{by Proposition 4.18} \\ &\subseteq \exists^1[1]\,\exists^1[1]\,\forall^p P && \text{by Lemma 4.5} \\ &\subseteq S\forall^p P \end{aligned}$$

Statement (4):

$$\begin{aligned} S\forall^p P &\subseteq \mathrm{Norm}(S\forall^p)\,P \subseteq \exists^2\forall^p P \\ &\subseteq NEXPTIME = \exists^1[1]\,\forall^p\exists^p P && \text{by Theorems 4.11 and 4.9} \\ &\subseteq \exists^1[1]\,\forall^p\exists^p\forall^p P && \text{by Lemma 4.3} \\ &\subseteq S\forall^p P && \text{by assumption} \end{aligned}$$

Statement (5):

$$\begin{aligned} S\forall^p P &\subseteq \mathrm{Norm}(S\forall^p)\,P \subseteq \exists^2\forall^p P \\ &\subseteq \exists^2[3]\,\forall^p P = NEXPTIME && \text{by Theorem 4.11} \\ &\subseteq \exists^1[1]\,\exists^1[1]\,\exists^1[1]\,\forall^p P && \text{by Lemma 4.5} \\ &\subseteq S\forall^p P \end{aligned}$$

❏

Thus, we have characterized all classes $S\overline{Q}^p P$ by well-known complexity classes, whose set quantifiers appearing in S are existential (universal, respectively). Next we consider the classes containing existential and universal set quantifiers. For $S \in \Gamma^*_{[p]}$, let $\mathrm{Last}(S)$ be the last quantifier of the chain S ($\mathrm{Last}(\varepsilon) =_{\mathrm{df}} \varepsilon$). Observe that if $\mathrm{Last}(S)$ is a set quantifier in $\Gamma_{[p,\overline{Q}]}$ for $Q \in \{\exists, \forall\}$, then $\mathrm{Norm}(S)$ contains at most one word quantifier which is Q^p (if exists).

If $S\overline{Q}^p P$ is a class containing existential and universal set quantifiers, where in the last $\exists$-$\forall$-alternate sequence of set quantifiers there exists one which is not restricted to one query, then $S\overline{Q}^p P$ coincides with one class of the exponential-time alternation hierarchy. This is shown in the following theorem.

Theorem 4.22. *Let* $Q \in \{\exists, \forall\}$ *and* $R \in \Gamma^+_{[p]}$ *with* $\mathrm{FirstSet}(R) = \exists$ *and* $\mathrm{Last}(R)$ *being a set quantifier in* $\Gamma_{[p,\overline{Q}]}$. *Furthermore, let* $\sigma, \tau \in \{1, 2\}$ *and* $r_1, r_2, r_3 : \mathbb{N} \to \mathbb{N}_+$ *such that* $r_2 \geq 2$ *and* $r_3 \geq 3$. *Then*

(1) $RS\overline{Q}^p P = \Sigma^{\exp}_{k-1}$ *for* $S \in \mathcal{F}\big(Q^1[r_2] \,|\, \exists^p, \forall^p\big) \cup \mathcal{F}\big(Q^2[2] \,|\, \exists^p, \forall^p\big)$;

(2) $RS\overline{Q}^p P = \Sigma^{\exp}_{k}$ *for* $S \in \mathcal{F}\big(Q^2[r_3] \,\big|\, \Gamma_{[p,Q]}, \overline{Q}^p\big) \cup \mathcal{F}\big(Q^\sigma[r_1]\,, Q^\tau[r_2] \,\big|\, \Gamma_{[p,Q]}, \overline{Q}^p\big)$;

where k *is the number of set quantifiers in* $\mathrm{Norm}(RS)$.

Proof. For $l \geq 1$, let $Q_l =_{\mathrm{df}} \exists$ if l is odd and $Q_l =_{\mathrm{df}} \forall$ otherwise, and let $Z =_{\mathrm{df}} Q^p$ if $\mathrm{Norm}(R)$ contains a word quantifier and $Z =_{\mathrm{df}} \varepsilon$ otherwise. Thus,

$$RS\overline{Q}^p P \subseteq \mathrm{Norm}(R)\, S\overline{Q}^p P \subseteq \exists^2\forall^2\exists^2 \ldots Q^2_{k-1} Z S\overline{Q}^p P$$

Let $\mathcal{C} =_{\mathrm{df}} \exists^2\forall^2\exists^2 \ldots Q^2_{k-1} Z S\overline{Q}^p P$. We conclude for

Statement (1): For $S \in \mathcal{F}\big(Q^1[r_2] \,|\, \exists^p, \forall^p\big)$ follows

$$\begin{aligned} \mathcal{C} &\subseteq \exists^2\forall^2\exists^2 \ldots Q^2_{k-1}\, \mathrm{Norm}\big(ZS\overline{Q}^p\big)\, P \subseteq \mathcal{K}_P\big(2^{k-1}1, 1\big) \\ &\subseteq \mathcal{K}_P\big((1[1])^{k-1}1[2]\,, 1\big) = \Sigma^{\exp}_{k-1} && \text{by Theorem 4.12} \\ &\subseteq RS\overline{Q}^p P \end{aligned}$$

Now, consider the case $S \in \mathcal{F}\big(Q^2[2] \,|\, \exists^p, \forall^p\big)$. By obvious rule $Q^2[2]\, Q^p \leftrightarrow_{[P]} Q^p Q^2[2]$, there is an $\exists$-$\forall$-alternate sequence Y of word quantifiers such that $ZS\overline{Q}^p \to_{[P]} YQ^2[2]\,\overline{Q}^p$. Hence,

$$\begin{aligned} \mathcal{C} &\subseteq \exists^2\forall^2\exists^2 \ldots Q^2_{k-1} Y Q^2[2]\, \overline{Q}^p P \\ &\subseteq \exists^2\forall^2\exists^2 \ldots Q^2_{k-1} \mathit{PSPACE}^{0 2^{k-1}} && \text{by Proposition 4.18} \\ &\subseteq \Sigma^{\exp}_{k-1} && \text{by Proposition 3.12} \\ &\subseteq \mathcal{K}_P\big((1[1])^{k-1}2[2]\,, 1\big) && \text{by Theorem 4.12} \\ &\subseteq RS\overline{Q}^p P \end{aligned}$$

Statement (2):

$$\begin{aligned} \mathcal{C} &\subseteq \exists^2\forall^2\exists^2\dots Q^2_{k-1}\,\mathrm{Norm}\big(ZS\overline{Q}^p\big)\,P \subseteq \mathcal{K}_P\big(2^k, 1\big) \\ &\subseteq \mathcal{K}_P\big((1\,[1])^{k-1}2[3]\,,1\big) = \Sigma^{\exp}_k && \text{by Theorem 4.12} \\ &\subseteq RS\overline{Q}^pP \end{aligned}$$

where for the last inclusion we use Lemma 4.5 if $S \in \mathcal{F}\big(Q^\sigma[r_1]\,, Q^\tau[r_2]\,\big|\Gamma_{[p,Q]}, \overline{Q}^p\big)$. ❑

Hence, it remains to consider the classes $S\overline{Q}^pP$ containing existential and universal set quantifiers, where in the last ∃-∀-alternate sequence of set quantifiers all are restricted to one query. Not all cases are considered in the next theorem, the case $S \in \mathcal{F}\big(Q^\sigma[1]\,\big|\overline{Q}^p\big)$ will be examined in Theorems 4.24 and 4.25.

Theorem 4.23. *Let* $Q \in \{\exists, \forall\}$ *and* $R \in \Gamma^+_{[p]}$ *with* $\mathrm{FirstSet}(R) = \exists$ *and* $\mathrm{Last}(R)$ *being a set quantifier in* $\Gamma_{[p,\overline{Q}]}$. *Furthermore, let* $\sigma, \tau \in \{1, 2\}$ *and* $U = \bigcup_{\rho\in\{1,2\}} \mathcal{F}(Q^p, Q^\rho[1]\,|\exists^p, \forall^p\,)$. *Then*

(1) $RS\overline{Q}^pP = \Sigma^{\exp}_{k-1}$ *for* $S \in \mathcal{F}(Q^p, Q^\sigma[1]\,|\exists^p, \forall^p\,)$

(2) $RS\overline{Q}^pP = \Sigma^{\exp}_{k-1}$ *for* $S \in \mathcal{F}(Q^\sigma[1]\,, Q^\tau[1]\,|\exists^p, \forall^p\,)$ *such that the substring* $\overline{Q}^pT$ *does not appear after the first set quantifier in* S *for all* $T \in U$

(3) $RS\overline{Q}^pP = \Sigma^{\exp}_{k}$ *for* $S \in \mathcal{F}(Q^\sigma[1]\,, Q^\tau[1]\,|\exists^p, \forall^p\,)$ *such that the substring* $\overline{Q}^pT$ *appears after the first set quantifier in* S *for some* $T \in U$

(4) $RS\overline{Q}^pP = \Sigma^{\exp}_{k}$ *for* $S \in \mathcal{F}\big(Q^\sigma[1]\,, Q^\sigma[1]\,, Q^\tau[1]\,\big|Q^1[1]\,, Q^2[1]\,, \exists^p, \forall^p\big)$

where k *is the number of set quantifiers in* $\mathrm{Norm}(RS)$.

Proof. For $l \geq 1$, let $Q_l =_{\mathrm{df}} \exists$ if l is odd and $Q_l =_{\mathrm{df}} \forall$ otherwise, and let $Z =_{\mathrm{df}} Q^p$ if $\mathrm{Norm}(R)$ contains a word quantifier and $Z =_{\mathrm{df}} \varepsilon$ otherwise. We conclude for

Statement (1): By obvious rule $Q^\sigma[1]\,Q^p \leftrightarrow_{[P]} Q^pQ^\sigma[1]$, there exists an ∃-∀-alternate sequence Y of word quantifiers such that $ZS\overline{Q}^p \rightarrow_{[P]} YQ^\sigma[1]\,\overline{Q}^p$. Hence,

$$\begin{aligned} RS\overline{Q}^pP &\subseteq \mathrm{Norm}(R)\,S\overline{Q}^pP \subseteq \exists^2\forall^2\exists^2\dots Q^2_{k-1}ZS\overline{Q}^pP \\ &\subseteq \exists^2\forall^2\exists^2\dots Q^2_{k-1}YQ^\sigma[1]\,\overline{Q}^pP \subseteq \exists^2\forall^2\exists^2\dots Q^2_{k-1}Y\overline{Q}^pP && \text{by Proposition 4.17} \\ &\subseteq \mathcal{K}_P\big((1\,[1])^{k-1}, 2\big) = \Sigma^{\exp}_{k-1} && \text{by Theorem 4.9} \\ &\subseteq RS\overline{Q}^pP \end{aligned}$$

Statement (2): By assumption if a $\overline{Q}^p$ appears after the first set quantifier in S, then no Q^p follows this $\overline{Q}^p$. Therefore, by obvious rule $Q^\sigma[1]\,Q^p \leftrightarrow_{[P]} Q^pQ^\sigma[1]$ and rules of Section 4.3, there exists an ∃-∀-alternate sequence Y of word quantifiers such that

$ZS\overline{Q}^p \to_{[P]} YQ^2[2]\,\overline{Q}^p$. Hence,

$$\begin{aligned}
RS\overline{Q}^pP &\subseteq \mathrm{Norm}(R)\,S\overline{Q}^pP \subseteq \exists^2\forall^2\exists^2\dots Q^2_{k-1}ZS\overline{Q}^pP \\
&\subseteq \exists^2\forall^2\exists^2\dots Q^2_{k-1}YQ^2[2]\,\overline{Q}^pP \\
&\subseteq \exists^2\forall^2\exists^2\dots Q^2_{k-1}PSPACE^{02^{k-1}} = \Sigma^{\exp}_{k-1} && \text{by Propositions 4.18 and 3.12} \\
&\subseteq \mathcal{K}_P\big((1[1])^{k-1}1[2]\,,1\big) && \text{by Theorem 4.12} \\
&\subseteq \exists^1[1]\,\forall^1[1]\,\exists^1[1]\dots Q^1_{k-1}[1]\;Q^1[1]\;Q^1[1]\;\overline{Q}^pP && \text{by Lemma 4.5} \\
&\subseteq RS\overline{Q}^pP
\end{aligned}$$

Statement (3):

$$\begin{aligned}
RS\overline{Q}^pP &\subseteq \mathrm{Norm}\big(RS\overline{Q}^p\big)\,P \subseteq \exists^2\forall^2\exists^2\dots Q^2_k\overline{Q}^pP \\
&\subseteq \Sigma^{\exp}_k \subseteq \mathcal{K}_P\big((1[1])^k,2\big) && \text{by Theorems 4.12 and 4.9} \\
&\subseteq \exists^1[1]\,\forall^1[1]\,\exists^1[1]\dots Q^1_{k-1}[1]\;Q^1[1]\;\overline{Q}^pQ^p\overline{Q}^pP && \text{by Lemma 4.3} \\
&\subseteq RS\overline{Q}^pP && \text{by assumption}
\end{aligned}$$

Statement (4):

$$\begin{aligned}
RS\overline{Q}^pP &\subseteq \mathrm{Norm}\big(RS\overline{Q}^p\big)\,P \subseteq \exists^2\forall^2\exists^2\dots Q^2_k\overline{Q}^pP \\
&\subseteq \mathcal{K}_P\big((1[1])^{k-1}2[3]\,,1\big) = \Sigma^{\exp}_k && \text{by Theorem 4.12} \\
&\subseteq \exists^1[1]\,\forall^1[1]\,\exists^1[1]\dots Q^1_{k-1}[1]\;Q^1[1]\;Q^1[1]\;Q^1[1]\;\overline{Q}^pP && \text{by Lemma 4.5} \\
&\subseteq RS\overline{Q}^pP
\end{aligned}$$ ❑

Thus, only the case $RS\overline{Q}^pP$ with $S \in \mathcal{F}\big(Q^\sigma[1]\,\big|\overline{Q}^p\big)$ and $\mathrm{Last}(R)$ being a set quantifier in $\Gamma_{[p,\overline{Q}]}$ remains to be considered. By Lemmas 4.4 and 4.6 follows $RS\overline{Q}^pP = RQ^\sigma[1]\,\overline{Q}^pP$. The classes $RQ^\sigma[1]\,\overline{Q}^pP$ will be study in two steps: When $\mathrm{Norm}(R)$ does not contain a word quantifier (Theorem 4.24) and when $\mathrm{Norm}(R)$ contains a word quantifier (Theorem 4.25). Note that in these theorems the quantifier string R is subdivided in three parts, namely RST.

Theorem 4.24. *Let* $Q \in \{\exists,\forall\}$, $R \in \Gamma^*_{[p]}$ *with* $R = \varepsilon$ *or* $\mathrm{Last}(R)$ *being a set quantifier in* $\Gamma_{[p,\overline{Q}]}$, $S \in \big(\Gamma_{[p,Q]} \cup \{\overline{Q}^p\}\big)^*$ *with* $S = \varepsilon$ *or* $\mathrm{Last}(S)$ *being a set quantifier, and* $T \in \big(\Gamma_{[p,\overline{Q}]} \cup \{Q^p\}\big)^+$ *with* $\mathrm{Last}(T)$ *being a set quantifier. Furthermore, let* $\mathrm{FirstSet}(RST) = \exists$, $\sigma,\tau,\rho \in \{1,2\}$ *and* $r : \mathbb{N} \to \mathbb{N}$ *such that* $r \geq 2$. *If* $\mathrm{Norm}(RST)$ *does not contain a word quantifier then*

(1) $RSTQ^\sigma[1]\,\overline{Q}^pP = NP$ *for* $R = S = \varepsilon$*;*

(2) $RSTQ^\sigma[1]\,\overline{Q}^pP = PSPACE$ *for* $S \in \mathcal{F}(Q^\tau[1]\,|\exists^p,\forall^p)$ *and* $R = \varepsilon$*;*

(3) $RSTQ^\sigma[1]\,\overline{Q}^pP = \Sigma^{\exp}_{k-1}$ *for* $S \in \mathcal{F}(Q^\tau[1]\,|\exists^p,\forall^p)$ *and* $R \neq \varepsilon$*;*

(4) $\mathsf{RSTQ}^{\sigma}[1]\,\overline{\mathsf{Q}}^{\mathsf{p}}P = \Sigma_{\mathsf{k}}^{\exp}$ *for*

$$\mathsf{S} \in \mathcal{F}\big(\mathsf{Q}^{\tau}[1]\,,\, \mathsf{Q}^{\rho}[1]\,\big|\Gamma_{[\mathsf{p},\mathsf{Q}]}, \overline{\mathsf{Q}}^{\mathsf{p}}\big) \cup \mathcal{F}\big(\mathsf{Q}^{\tau}[\mathsf{r}]\,\big|\Gamma_{[\mathsf{p},\mathsf{Q}]}, \overline{\mathsf{Q}}^{\mathsf{p}}\big)\,;$$

where k *is the number of set quantifiers in* $\mathrm{Norm}(\mathsf{RS})$.

Proof. For $\mathsf{l} \geq 1$, let $\mathsf{Q}_{\mathsf{l}} =_{\mathrm{df}} \exists$ if l is odd and $\mathsf{Q}_{\mathsf{l}} =_{\mathrm{df}} \forall$ otherwise.

Statement (1): We conclude ($\mathsf{Q} = \forall$)

$$\begin{aligned}
\mathsf{RSTQ}^{\sigma}[1]\,\overline{\mathsf{Q}}^{\mathsf{p}}P &\subseteq \mathrm{Norm}(\mathsf{T})\,\mathsf{Q}^{\sigma}[1]\,\overline{\mathsf{Q}}^{\mathsf{p}}P \subseteq \exists^2\forall^{\sigma}[1]\,\exists^{\mathsf{p}}P \\
&\subseteq \exists^2\exists^{\mathsf{p}}P && \text{by Proposition 4.17} \\
&\subseteq \exists^{\mathsf{p}}P = NP && \text{by Lemmas 4.4 and 4.7} \\
&\subseteq \mathsf{RSTQ}^{\sigma}[1]\,\overline{\mathsf{Q}}^{\mathsf{p}}P
\end{aligned}$$

Statements (2) and (3): Let $\mathsf{Z} =_{\mathrm{df}} \mathsf{Q}^{\mathsf{p}}$ if $\mathrm{Norm}(\mathsf{R})$ contains a word quantifier and $\mathsf{Z} =_{\mathrm{df}} \varepsilon$ otherwise. Since $\mathrm{Norm}(\mathsf{RST})$ does not contain a word quantifier, there is only one possibility for Q^{p} quantifiers to appear in T which is left to the $\overline{\mathsf{Q}}$-quantifiers, i.e. at begin of T. Let T' be the chain T removing the Q^{p} quantifiers. Since $\mathrm{Last}(\mathsf{S})$ is a set quantifier and by obvious rule $\mathsf{Q}^2[2]\,\mathsf{Q}^{\mathsf{p}} \leftrightarrow_{[P]} \mathsf{Q}^{\mathsf{p}}\mathsf{Q}^2[2]$, there exists an $\exists$-$\forall$-alternate sequence Y of word quantifiers such that $\mathsf{Z}\,(\mathsf{S})_2\,\mathsf{Q}^{\mathsf{p}} \leftrightarrow_{[P]} \mathsf{Y}\mathsf{Q}^2[2]$. Thus,

$$\begin{aligned}
\mathsf{RSTQ}^{\sigma}[1]\,\overline{\mathsf{Q}}^{\mathsf{p}}P &\subseteq \mathrm{Norm}(\mathsf{R})\,S\mathsf{Q}^{\mathsf{p}}\,\mathrm{Norm}(\mathsf{T}')\,\mathsf{Q}^{\sigma}[1]\,\overline{\mathsf{Q}}^{\mathsf{p}}P \\
&\subseteq \exists^2\forall^2\exists^2\ldots\mathsf{Q}^2_{\mathsf{k}-1}\mathsf{Z}S\mathsf{Q}^{\mathsf{p}}\overline{\mathsf{Q}}^2\mathsf{Q}^{\sigma}[1]\,\overline{\mathsf{Q}}^{\mathsf{p}}P \\
&\subseteq \exists^2\forall^2\exists^2\ldots\mathsf{Q}^2_{\mathsf{k}-1}\mathsf{Z}\,(\mathsf{S})_2\,\mathsf{Q}^{\mathsf{p}}\overline{\mathsf{Q}}^2\overline{\mathsf{Q}}^{\mathsf{p}}P && \text{by Proposition 4.17} \\
&\subseteq \exists^2\forall^2\exists^2\ldots\mathsf{Q}^2_{\mathsf{k}-1}\mathsf{Z}\,(\mathsf{S})_2\,\mathsf{Q}^{\mathsf{p}}\overline{\mathsf{Q}}^{\mathsf{p}}P && \text{by Lemmas 4.4 and 4.7} \\
&\subseteq \exists^2\forall^2\exists^2\ldots\mathsf{Q}^2_{\mathsf{k}-1}\mathsf{Y}\mathsf{Q}^2[2]\,\overline{\mathsf{Q}}^{\mathsf{p}}P && \text{by } \mathsf{Z}\,(S)_2\,\mathsf{Q}^{\mathsf{p}} \leftrightarrow_{[P]} \mathsf{Y}\mathsf{Q}^2[2] \\
&\subseteq \exists^2\forall^2\exists^2\ldots\mathsf{Q}^2_{\mathsf{k}-1}PSPACE^{0 2^{\mathsf{k}-1}} && \text{by Proposition 4.18}
\end{aligned}$$

Let $\mathcal{C} =_{\mathrm{df}} \exists^2\forall^2\exists^2\ldots\mathsf{Q}^2_{\mathsf{k}-1}PSPACE^{0 2^{\mathsf{k}-1}}$. For $\mathsf{R} = \varepsilon$ (i.e. $\mathsf{k} = 1$), $\mathcal{C} = PSPACE$ which is included in $\mathsf{Q}^1[2]\,\overline{\mathsf{Q}}^{\mathsf{p}}P$ by Theorem 4.10, and therefore it is included in $\mathsf{RSTQ}^{\sigma}[1]\,\overline{\mathsf{Q}}^{\mathsf{p}}P$ by Lemma 4.5. Now, for $\mathsf{R} \neq \varepsilon$ (i.e. $\mathsf{k} \geq 2$) we conclude ($\mathsf{Q}_{\mathsf{k}} = \mathsf{Q}$)

$$\begin{aligned}
\mathcal{C} &\subseteq \Sigma^{\exp}_{\mathsf{k}-1} && \text{by Proposition 3.12} \\
&\subseteq \mathcal{K}_P\big((1[1])^{\mathsf{k}-1}1[2]\,,1\big) && \text{by Theorem 4.12} \\
&\subseteq \exists^1[1]\,\forall^1[1]\,\exists^1[1]\ldots\mathsf{Q}^1_{\mathsf{k}-1}[1]\,\mathsf{Q}^1_{\mathsf{k}}[1]\,\mathsf{Q}^{\sigma}[1]\,\overline{\mathsf{Q}}^{\mathsf{p}}P && \text{by Lemma 4.5} \\
&\subseteq \mathsf{RSTQ}^{\sigma}[1]\,\overline{\mathsf{Q}}^{\mathsf{p}}P
\end{aligned}$$

Statement (4): We conclude ($Q_{k+1} = \overline{Q}$)

$$
\begin{aligned}
&\mathsf{RSTQ}^{\sigma}[1]\,\overline{Q}^{p}P \\
&\quad\subseteq \operatorname{Norm}(\mathsf{RST})\,Q^{\sigma}[1]\,\overline{Q}^{p}P \\
&\quad\subseteq \exists^2\forall^2\exists^2\ldots Q^2_{k+1}Q^{\sigma}[1]\,\overline{Q}^{p}P \\
&\quad\subseteq \exists^2\forall^2\exists^2\ldots Q^2_{k+1}\overline{Q}^{p}P && \text{by Proposition 4.17} \\
&\quad\subseteq \exists^2\forall^2\exists^2\ldots Q^2_{k}\overline{Q}^{p}P && \text{by Lemmas 4.4 and 4.7} \\
&\quad\subseteq \mathcal{K}_P\big((1[1])^{k-1}2[3]\,,1\big) = \Sigma^{\mathrm{exp}}_k && \text{by Theorem 4.12} \\
&\quad\subseteq \left\{\begin{array}{l} \exists^1[1]\,\forall^1[1]\ldots Q^1_{k-1}[1]\,Q^1_k[1]\,Q^1_k[1]\,Q^{\sigma}[1]\,\overline{Q}^{p}P \\ \exists^1[1]\,\forall^1[1]\ldots Q^1_{k-1}[1]\,Q^1_k[2]\,Q^{\sigma}[1]\,\overline{Q}^{p}P \end{array}\right. && \text{by Lemma 4.5} \\
&\quad\subseteq \mathsf{RSTQ}^{\sigma}[1]\,\overline{Q}^{p}P && ❑
\end{aligned}
$$

In the previous theorem, there exists only one possibility for $\operatorname{Norm}(\mathsf{RST})$ to contain a word quantifier: There exists an Q^p quantifier appearing right to an $\overline{Q}$-quantifier in T. With this observation, we can state the following result which completes the characterization of the classes $S\overline{Q}^{p}P$.

Theorem 4.25. *Let* $Q \in \{\exists, \forall\}$, $R \in \Gamma^*_{[p]}$ *with* $R = \varepsilon$ *or* $\operatorname{Last}(R)$ *being a set quantifier in* $\Gamma_{[p,\overline{Q}]}$, $S \in \left(\Gamma_{[p,Q]} \cup \left\{\overline{Q}^p\right\}\right)^*$ *with* $S = \varepsilon$ *or* $\operatorname{Last}(S)$ *being a set quantifier, and* $T \in \mathcal{F}\left(Q^p \,\middle|\, \Gamma_{[p,\overline{Q}]}, Q^p\right)$ *with* $\operatorname{Last}(T)$ *being a set quantifier. Furthermore, let* $\operatorname{FirstSet}(\mathsf{RST}) = \exists$ *and if* $S = \varepsilon$ *then let* $R = \varepsilon$. *If* $\operatorname{Norm}(\mathsf{RST})$ *contains a word quantifier then*

(1) $\mathsf{RSTQ}^{\sigma}[1]\,\overline{Q}^{p}P = \Sigma^p_l$ *if* $R = S = \varepsilon$, *no* Q^p *quantifier appears right to a set quantifier in* T *and the first word quantifier in* T *is* $\exists^p$

(2) $\mathsf{RSTQ}^{\sigma}[1]\,\overline{Q}^{p}P = \Pi^p_l$ *if* $R = S = \varepsilon$, *no* Q^p *quantifier appears right to a set quantifier in* T *and the first word quantifier in* T *is* $\forall^p$

(3) $\mathsf{RSTQ}^{\sigma}[1]\,\overline{Q}^{p}P = \Sigma^{\mathrm{exp}}_k$ *if* $S \neq \varepsilon$ *and no* Q^p *quantifier appears right to a set quantifier in* T

(4) $\mathsf{RSTQ}^{\sigma}[1]\,\overline{Q}^{p}P = \Sigma^{\mathrm{exp}}_{k+1}$ *if there is an* Q^p *quantifier appearing right to a set quantifier in* T

where $l - 1$ *is the number of* $\exists$-$\forall$-*alternations of the word quantifiers in* $T\overline{Q}^p$ *and* k *is the number of set quantifiers in* $\operatorname{Norm}(\mathsf{RS})$.

Proof. We prove Statements (1), (3) and (4), since Statement (2) follows as Statement (1). For $n \geq 1$, let $Q_n =_{\mathrm{df}} \exists$ if n is odd and $Q_n =_{\mathrm{df}} \forall$ otherwise. By assumption, there exists an Q^p quantifier appearing right to an $\overline{Q}$-quantifier in T ($\operatorname{Norm}(\mathsf{RST})$ contains a word quantifier).

Statements (1) and (3): Let $Z =_{\mathrm{df}} \overline{Q}^p$ if $\operatorname{Norm}(\mathsf{RS})$ contains a word quantifier and $Z =_{\mathrm{df}} \varepsilon$ otherwise, and let Y be an $\exists$-$\forall$-alternate sequence of word quantifiers with $l - 2$ alternations, where the first word quantifier of Y and T coincide. Thus, by assumption follows

$T \to_{[P]} Y\overline{Q}^2$. Then,

$$\begin{aligned}
RSTQ^\sigma[1]\,\overline{Q}^pP &\subseteq \mathrm{Norm}(RS)\,Y\overline{Q}^2Q^\sigma[1]\,\overline{Q}^pP \\
&\subseteq \exists^2\forall^2\exists^2\dots Q_k^2ZY\overline{Q}^2Q^\sigma[1]\,\overline{Q}^pP \\
&\subseteq \exists^2\forall^2\exists^2\dots Q_k^2ZY\overline{Q}^2\overline{Q}^pP && \text{by Proposition 4.17} \\
&\subseteq \exists^2\forall^2\exists^2\dots Q_k^2ZY\overline{Q}^pP && \text{by Lemmas 4.4 and 4.7}
\end{aligned}$$

Let $\mathcal{C} =_{\mathrm{df}} \exists^2\forall^2\exists^2\dots Q_k^2ZY\overline{Q}^pP$. For $S = \varepsilon$ we have $R = Z = \varepsilon$ and $\mathcal{C} \subseteq Y\overline{Q}^pP \subseteq \exists^p\forall^p\exists^p\dots Q_l^pP \subseteq \Sigma_l^p$ which is included in $RSTQ^\sigma[1]\,\overline{Q}^pP$. Now, for $S \neq \varepsilon$ (by assumption, in T must appear the substring $\overline{Q}^pQ^p$ and $Q_{k+1} = \overline{Q}$)

$$\begin{aligned}
\mathcal{C} &\subseteq \mathcal{K}_P\big((1[1])^k, 2\big) = \Sigma_k^{\exp} && \text{by Theorem 4.9} \\
&\subseteq \exists^1[1]\,\forall^1[1]\,\exists^1[1]\dots Q_k^1[1]\,\overline{Q}^pQ^pQ^\sigma[1]\,\overline{Q}^pP \\
&\subseteq RSTQ^\sigma[1]\,\overline{Q}^pP
\end{aligned}$$

Statement (4): We conclude ($Q_k = Q$)

$$\begin{aligned}
RSTQ^\sigma[1]\,\overline{Q}^pP &\subseteq \mathrm{Norm}(RST)\,Q^\sigma[1]\,\overline{Q}^pP \\
&\subseteq \exists^2\forall^2\exists^2\dots Q_{k+1}^2Q^pQ^\sigma[1]\,\overline{Q}^pP \\
&\subseteq \exists^2\forall^2\exists^2\dots Q_{k+1}^2Q^p\overline{Q}^pP && \text{by Proposition 4.17} \\
&\subseteq \mathcal{K}_P\big((1[1])^{k+1}, 2\big) = \Sigma_{k+1}^{\exp} && \text{by Theorem 4.9} \\
&\subseteq \exists^1[1]\,\forall^1[1]\,\exists^1[1]\dots Q_{k+1}^1[1]\,Q^pQ^\sigma[1]\,\overline{Q}^pP \\
&\subseteq RSTQ^\sigma[1]\,\overline{Q}^pP && \text{by assumption} \qquad ❑
\end{aligned}$$

4.5.2 Open Cases

From the results obtained in §4.5.1 it remains to characterize the classes of the bounded analytic polynomial-time hierarchy having the form $ST P$ for $Q \in \{\exists, \forall\}$, $S \in \Gamma_{[p]}^*$ and $T \in \Gamma_{[p,Q]}^+$, where the quantifiers Q^p, Q^1 and Q^2 do not appear in the chain T. Next, we examine some of these remainder classes. We start showing that existential (universal) set quantifiers with restriction on oracle queries applied to P are as powerful as the class (r)-P (co(r)-P, respectively) for some $r : \mathbb{N} \to \mathbb{N}$.

Lemma 4.26. *Let* $k \in \mathbb{N}$, $\sigma_1, \dots, \sigma_k \in \{1, 2\}$, $r_1, \dots, r_k : \mathbb{N} \to \mathbb{N}$ *and* $r = r_1 + \dots + r_k$. *Then* $\exists^{\sigma_1}[r_1]\dots\exists^{\sigma_k}[r_k]\,P = (r)$-$P$.

Proof. By Lemmas 4.3 and 4.4 follows $\exists^{\sigma_1}[r_1]\dots\exists^{\sigma_k}[r_k]\,P \subseteq \exists^2[r]\,P$. Thus, it suffices to prove (r)-$P \subseteq \exists^{\sigma_1}[r_1]\dots\exists^{\sigma_k}[r_k]\,P$ and $\exists^2[r]\,P \subseteq (r)$-$P$.

"(r)-$P \subseteq \exists^{\sigma_1}[r_1]\dots\exists^{\sigma_k}[r_k]\,P$": Let $L \in (r)$-P. There exist an $L_1 \in P^{00}$ and a polynomial p such that

$$\begin{aligned}
x \in L &\iff \exists^p u\,(|u| = \min\{r(|x|), p(|x|)\} \wedge (x, u) \in L_1) \\
&\iff \exists^1 U_1 \dots \exists^1 U_k\,((x, U_1, \dots, U_k) \in L_2),
\end{aligned}$$

where $L_2 =_{\mathrm{df}} \{(x, U_1, \dots, U_k) : (x, \langle U_1, f_1(|x|)\rangle \dots \langle U_k, f_k(|x|)\rangle) \in L_1\}$ and $f_1, \dots, f_k : \mathbb{N} \to \mathbb{N}$ are functions satisfying following properties[1]: $f_i(|x|) \leq r_i(|x|)$ for $i = 1, \dots, k$ and $\sum_{i=1}^{k} f_i(|x|) = \min\{r(|x|), p(|x|)\}$. Obviously, the language $L_2 \in P^{01[r_1]\dots 1[r_k]}$, i.e. $L \in \exists^1[r_1] \dots \exists^1[r_k]\, P$.

"$\exists^2[r]\, P \subseteq (r)\text{-}P$": The proof follows as in Lemma 3.5. By definition $L \in \exists^2[r]\, P$ if and only if there exists an $L' \in P^{02[r]}$, such that $x \in L \iff \exists^2 U\, ((x, U) \in L')$. Next, let M be a machine of type $02[r]$ accepting L' with time bound p where p is a polynomial. Without loss of generality we assume that M does not make a query twice. Let M' be a machine of type 0 that on input x guesses a word u of length $\min\{r(|x|), p(|x|)\}$ and then works as M on input (x, U) with the following difference: Instead of the answer of U to the i-th query of M the machine M' uses the i-th bit of u. Hence, M' is a polynomial-time r-nondeterministic machine and $\exists^2 U\, ((x, U) \in L') \iff x \in L(M')$ can be seen as in Lemma 3.5. Therefore, $L \in (r)\text{-}P$. ❑

The previous lemma remains valid if a quantifier string $S \in \Gamma_{[p]}^{*}$ is applied to these classes, i.e. we have $S\exists^{\sigma_1}[r_1] \dots \exists^{\sigma_k}[r_k]\, P = S\,(r)\text{-}P^{0\tau(S)}$. Hence, the following result is evident.

Corollary 4.27. *Let* $S \in \Gamma_{[p]}^{*}$, $k \in \mathbb{N}$, $\sigma_1, \dots, \sigma_k \in \{1, 2\}$, $r_1, \dots, r_k : \mathbb{N} \to \mathbb{N}$ *and* $r = r_1 + \dots + r_k$. *Then* $S\exists^{\sigma_1}[r_1] \dots \exists^{\sigma_k}[r_k]\, P = S\exists^1[r]\, P$.

Thus, only the following cases are still open for $Q \in \{\exists, \forall\}$, $S \in \Gamma_{[p]}^{*}$ and $r : \mathbb{N} \to \mathbb{N}_+$ (Lemma 4.26 and Corollary 4.27): $SQ^1[r]\, P$ with $\mathrm{Last}(S) \in \Gamma_{[p,\overline{Q}]}$ and r being a real restriction. Next we consider some of these remainder cases.

Proposition 4.28. *Let* $S \in \Gamma_{[p,\exists]}^{*}$, $\sigma \in \{1, 2\}$ *and* $r : \mathbb{N} \to \mathbb{N}_+$. *If the quantifier* $\exists^p$, $\exists^1$ *or* $\exists^2$ *appears in the chain* S *then* $S\forall^{\sigma}[1]\, \exists^1[r]\, P = NP$.

Proof. We conclude

$$\begin{aligned} S\forall^{\sigma}[1]\, \exists^1[r]\, P &\subseteq \exists^2\forall^{\sigma}[1]\, \exists^p P && \text{by Lemmas 4.3, 4.4 and 4.7} \\ &\subseteq \exists^p P = NP && \text{by Proposition 4.17 and Lemmas 4.4 and 4.7} \\ &\subseteq S\forall^{\sigma}[1]\, \exists^1[r]\, P && \text{by Lemma 4.3} \end{aligned}$$ ❑

Proposition 4.29. *Let* $S \in \Gamma_{[p]}^{*}$ *with* $\mathrm{FirstSet}(S) = \exists$ *and* $r : \mathbb{N} \to \mathbb{N}_+$. *If there exist at least two word quantifiers in* $\mathrm{Norm}(S)$, *then* $S\exists^1[r]\, P = S\forall^1[r]\, P = \Sigma_k^{\mathrm{exp}}$, *where* k *is the number of set quantifiers in* $\mathrm{Norm}(S)$.

[1]Take for example:

$$f_i(|x|) =_{\mathrm{df}} \begin{cases} r_i(|x|) & \text{if } \sum_{j=1}^{i} f_j(|x|) \leq p(|x|), \\ p(|x|) - \sum_{j=1}^{i-1} f_j(|x|) & \text{if } \sum_{j=1}^{i-1} f_j(|x|) < p(|x|) < \sum_{j=1}^{i} f_j(|x|), \\ 0 & \text{otherwise.} \end{cases}$$

Proof. We conclude for $Q \in \{\exists, \forall\}$

$$\begin{aligned} SQ^1[r]\,P &\subseteq SQ^pP \subseteq \mathrm{Norm}(S)\,Q^pP && \text{by Lemma 4.7} \\ &\subseteq \mathcal{K}_P\big((1[1])^k, 2\big) = \Sigma_k^{\exp} && \text{by Theorem 4.9} \\ &\subseteq SQ^1[r]\,P && \square \end{aligned}$$

4.5.3 Parallel Queries

In §4.4.2 we show that all the characterizations obtained for the classes $\mathcal{K}_P(\cdot)$ and co$\mathcal{K}_P(\cdot)$ (§4.4.1) remain valid for the counterpart classes having the parallel queries restriction. Next, we show that also the characterizations of §4.5.1 and §4.5.2 preserve this restriction.

Observe that from proof of Lemma 4.26 and Corollary 4.27 we have already shown

$$S\exists^{\sigma_1}[r_1]\ldots\exists^{\sigma_k}[r_k]\,P = S\exists^{\sigma_1}[\|r_1]\ldots\exists^{\sigma_k}[\|r_k]\,P = S\exists^1[\|r]\,P = S\,(r)\text{-}P^{0\tau(S)}$$

Thus, Lemma 4.26 and Corollary 4.27 are also valid for these classes involving parallel queries. If we follow the proofs of the other results obtained in §4.5.1 and §4.5.2, either only one query for each oracle is enough or we have a characterization by a class in the form $\mathcal{K}_P(\cdot)$ and optionally we use the rules of Lemma 4.5. However, as we already observed (§4.4.2) all the characterizations of the classes $\mathcal{K}_P(\cdot)$ remain valid for the counterpart classes having the parallel queries restriction, and the rules of Lemma 4.5 preserve this restriction.

Therefore, all the characterizations of the classes of the bounded analytic polynomial-time hierarchy presented in this chapter remain valid under the parallel queries restriction.

4.6 Conclusions

We characterize classes of the bounded analytic polynomial-time hierarchy (§4.4 and §4.5). However, the following cases are still open for $Q \in \{\exists, \forall\}$, $S \in \Gamma^*_{[p]}$ and $r : \mathbb{N} \to \mathbb{N}_+$: $SQ^1[r]\,P$ with $\mathrm{Last}(S) \in \Gamma_{[p,\overline{Q}]}$, r being a real restriction and $SQ^1[r]\,P$ not satisfying the Propositions 4.28 and 4.29. In addition (§4.4.2 and §4.5.3), we show that these characterizations remain valid if the queries are asked in a nonadaptive form, i.e. in “parallel”. In special, all the characterizations for the classes of the analytic polynomial-time hierarchy (§3) also remain valid under the parallel queries restriction.

Finally, in the Figure 4.2 we point out how heavily (and nicely) the results can depend on the number of queries allowed.

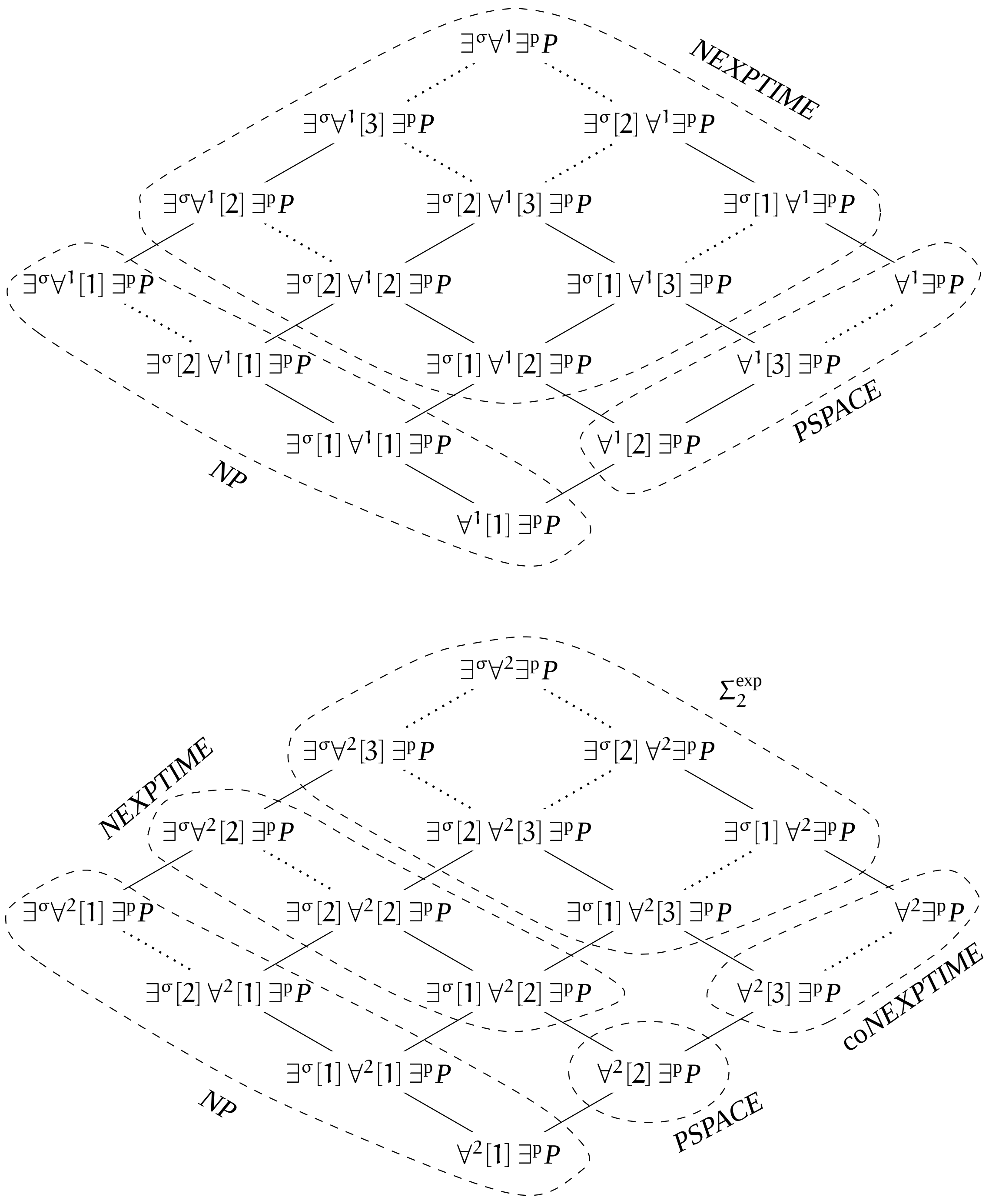

$\exists^\sigma\forall^1\exists^p P$
NEXPTIME
$\exists^\sigma\forall^1[3]\,\exists^p P$
$\exists^\sigma[2]\,\forall^1\exists^p P$
$\exists^\sigma\forall^1[2]\,\exists^p P$
$\exists^\sigma[2]\,\forall^1[3]\,\exists^p P$
$\exists^\sigma[1]\,\forall^1\exists^p P$
$\exists^\sigma\forall^1[1]\,\exists^p P$
$\exists^\sigma[2]\,\forall^1[2]\,\exists^p P$
$\exists^\sigma[1]\,\forall^1[3]\,\exists^p P$
$\forall^1\exists^p P$
$\exists^\sigma[2]\,\forall^1[1]\,\exists^p P$
$\exists^\sigma[1]\,\forall^1[2]\,\exists^p P$
$\forall^1[3]\,\exists^p P$
PSPACE
$\exists^\sigma[1]\,\forall^1[1]\,\exists^p P$
$\forall^1[2]\,\exists^p P$
NP
$\forall^1[1]\,\exists^p P$
$\exists^\sigma\forall^2\exists^p P$
Σ_2^{exp}
$\exists^\sigma\forall^2[3]\,\exists^p P$
$\exists^\sigma[2]\,\forall^2\exists^p P$
NEXPTIME
$\exists^\sigma\forall^2[2]\,\exists^p P$
$\exists^\sigma[2]\,\forall^2[3]\,\exists^p P$
$\exists^\sigma[1]\,\forall^2\exists^p P$
$\exists^\sigma\forall^2[1]\,\exists^p P$
$\exists^\sigma[2]\,\forall^2[2]\,\exists^p P$
$\exists^\sigma[1]\,\forall^2[3]\,\exists^p P$
$\forall^2\exists^p P$
$\exists^\sigma[2]\,\forall^2[1]\,\exists^p P$
$\exists^\sigma[1]\,\forall^2[2]\,\exists^p P$
$\forall^2[3]\,\exists^p P$
coNEXPTIME
$\exists^\sigma[1]\,\forall^2[1]\,\exists^p P$
$\forall^2[2]\,\exists^p P$
NP
PSPACE
$\forall^2[1]\,\exists^p P$

Figure 4.2: *Classes $\exists^\sigma[r]\,\forall^\tau[s]\,\exists^p P$ with $\sigma, \tau \in \{1, 2\}$ and $r, s : \mathbb{N} \to \mathbb{N}$, such that $r \geq 0$ and $s \geq 1$. In the left direction we increase r and in the right direction we increase s (for short we write $\forall^\tau[s]\,\exists^p P$ instead of $\exists^\sigma[0]\,\forall^\tau[s]\,\exists^p P$).*

CHAPTER 5

The Analytic Logarithmic-Space Hierarchy

"Puedo escribir los versos más tristes esta noche.
Pensar que no la tengo. Sentir que la he perdido.
Oír la noche inmensa, más inmensa sin ella.
Y el verso cae al alma como al pasto el rocío.
Que importa que el amor no pudiera guardarla.
La noche está estrellada y ella no está conmigo."

Pablo Neruda

Hierarchies defined over *P* using quantifiers have been intensively investigated whereas such hierarchies defined over subclasses of *P* remain unclear. In the present chapter, we are interested in the question of whether analytic polynomial-time hierarchy like results (§3) could also be established for a hierarchy over *L*, i.e. an ∃-∀-hierarchy defined over *L* using logarithmically length bounded word quantifiers as well as set quantifiers of type 1 and 2. This hierarchy is called *analytic logarithmic-space hierarchy*.

This chapter is organized as follows: We start defining the existential and universal quantifiers and the analytic logarithmic-space hierarchy (§5.1). Then, using equivalence rules we show that every class of this hierarchy can be represented in a certain normal form (§5.2). It turns out that the last quantifier of a class in this normal form is either a word quantifier or a set quantifier of type 2. Thus, we divide our study in two parts depending on this last quantifier: Whether it is a word quantifier (§5.3) or a set quantifier of type 2 (§5.4). It is shown that each class in this normal form, whose last quantifier is a word quantifier, coincides with *L* or one of the classes Σ_k^p and Π_k^p ($k \geq 1$) of the (arithmetic) polynomial-time hierarchy and vice versa (§5.3). Finally, some remarks about the results are made (§5.5).

5.1 The Operators and the Hierarchy

We will examine a logarithmic-space hierarchy built up by word and set quantifiers, which intuitively can be interpreted as the analytic polynomial-time hierarchy (§3) defined over *L* instead of *P*. Next, we define the existential and universal quantifiers (§5.1.1) and the analytic logarithmic-space hierarchy (§5.1.2).

5.1.1 The Existential and Universal Operators

We will investigate an $\exists$-$\forall$-hierarchy over L using word quantifiers as well as set quantifiers of type 1 and 2. The definitions of the quantifiers are as in §3.1.1, but the word quantifiers vary over words whose lengths are logarithmically bounded (instead of polynomially bounded) in the length of the input. However, to fix our notation we include these definitions. As usual, we define inductively new classes and in parallel the existential and universal quantifiers. Let $k \geq 1$ and $\sigma_1, \ldots, \sigma_k, \sigma \in \{0, 1, 2\}$. If $\mathcal{K}$ is a class of type $\sigma_1 \ldots \sigma_k \sigma$ then

For $\sigma = 0$: $\exists^{\log}\mathcal{K}$ and $\forall^{\log}\mathcal{K}$ are classes of type $\sigma_1 \ldots \sigma_k$ which are defined as follows

$L \in \exists^{\log}\mathcal{K} \Longleftrightarrow_{\mathrm{df}}$ there exist an $L' \in \mathcal{K}$ and a constant $c \in \mathbb{N}$, such that

$$(X_1, \ldots, X_k) \in L \longleftrightarrow \exists x\Big(|x| \leq c \cdot \log\big(\sum_{\substack{\sigma_i=0\\ i \leq k}} |X_i|\big) \wedge (X_1, \ldots, X_k, x) \in L'\Big)$$

$L \in \forall^{\log}\mathcal{K} \Longleftrightarrow_{\mathrm{df}}$ there exist an $L' \in \mathcal{K}$ and a constant $c \in \mathbb{N}$, such that

$$(X_1, \ldots, X_k) \in L \longleftrightarrow \forall x\Big(|x| \leq c \cdot \log\big(\sum_{\substack{\sigma_i=0\\ i \leq k}} |X_i|\big) \to (X_1, \ldots, X_k, x) \in L'\Big)$$

(Using simple encoding arguments it is easy to see that one can use equivalently "$=$" instead of "$\leq$" in these definitions.)

For $\sigma = 1, 2$: $\exists^{\sigma}\mathcal{K}$ and $\forall^{\sigma}\mathcal{K}$ are classes of type $\sigma_1 \ldots \sigma_k$ which are defined as follows

$L \in \exists^{\sigma}\mathcal{K} \Longleftrightarrow_{\mathrm{df}}$ there exists an $L' \in \mathcal{K}$, such that

$$(X_1, \ldots, X_k) \in L \longleftrightarrow \exists X\,((X_1, \ldots, X_k, X) \in L')$$

$L \in \forall^{\sigma}\mathcal{K} \Longleftrightarrow_{\mathrm{df}}$ there exists an $L' \in \mathcal{K}$, such that

$$(X_1, \ldots, X_k) \in L \longleftrightarrow \forall X\,((X_1, \ldots, X_k, X) \in L')$$

To make clear which type of input is used, we also write $\exists^{\log}x$ instead of $\exists x$, and $\forall^{\log}x$ instead of $\forall x$.

Following our notation, the set of existential and universal quantifiers is denoted by $\Gamma_{\log} =_{\mathrm{df}} \{\exists^{\log}, \exists^1, \exists^2, \forall^{\log}, \forall^1, \forall^2\}$. In our new context, we adapt the definition of the quantifier string function to comprise $\exists^{\log}$ and $\forall^{\log}$: For $k \geq 0$, $Q_1, \ldots, Q_k \in \{\exists, \forall\}$ and $\tau_1, \ldots, \tau_k \in \{\log, 1, 2\}$, let $\tau(Q_1^{\tau_1} \ldots Q_k^{\tau_k}) =_{\mathrm{df}} \sigma_1 \ldots \sigma_k$ be the type of the operator (or quantifier) string $Q_1^{\tau_1} \ldots Q_k^{\tau_k}$, where $\sigma_i = 0$ if $\tau_i = \log$ and $\sigma_i = \tau_i$ otherwise ($i = 1, \ldots, k$). For $Q = Q_1^{\tau_1} \ldots Q_k^{\tau_k}$ and $X = (X_1, \ldots, X_k)$ we also write QX instead of $Q_1^{\tau_1}X_1 \ldots Q_k^{\tau_k}X_k$. Furthermore, we define $\overline{Q} =_{\mathrm{df}} \overline{Q}_1^{\tau_1} \ldots \overline{Q}_k^{\tau_k}$. The following proposition is evident.

Proposition 5.1. *Let* $\mu \in \{0, 1, 2\}^*$ *and* $Q \in \Gamma_{\log}^*$. *Then* $\mathrm{co}QL^{\mu\tau(Q)} = \overline{Q}L^{\mu\tau(Q)}$.

Proof. The proof follows as in Proposition 3.1. ❑

5.1.2 The Analytic Logarithmic-Space Hierarchy

As the analytic polynomial-time hierarchy, the analytic logarithmic-space hierarchy will consist of "ordinary" classes of languages, i.e. classes of type 0. In this case, we will also omit the superscripts to L, i.e. for quantifier string $\mathsf{Q} \in \Gamma_{\log}^*$ we define $\mathsf{Q}L =_{\mathrm{df}} \mathsf{Q}L^{0\tau(\mathsf{Q})}$. Next, the analytic logarithmic-space hierarchy is defined. For quantifier strings $\mathsf{Q} \in \Gamma_{\log}^*$, the classes $\mathsf{Q}L$ are called the classes of the *analytic logarithmic-space hierarchy*. Finally, the class *ALH* is defined as the union of all classes of the analytic logarithmic-space hierarchy.

5.2 Equivalence Rules and Normal Form

We will employ equivalence rules (§5.2.1) to show that every class of the analytic logarithmic-space hierarchy can be represented in a certain normal form (§5.2.2).

5.2.1 Inclusion and Equivalence Rules

We will use inclusion rules to relate classes of the analytic logarithmic-space hierarchy in a similar way that was made for the classes of the analytic polynomial-time hierarchy (§3.2.1). These inclusion rules mean the following: For $\mathsf{R}, \mathsf{S} \in \Gamma_{\log}^*$, the *inclusion rule* $\mathsf{R} \to_L \mathsf{S}$ is valid if the replacement of the quantifier string R by the string S in any context does not diminish the class in question, i.e. $\mathsf{RQ}L^{\mu\tau(\mathsf{R})\tau(\mathsf{Q})} \subseteq \mathsf{SQ}L^{\mu\tau(\mathsf{S})\tau(\mathsf{Q})}$ for all $\mathsf{Q} \in \Gamma_{\log}^*$ and $\mu \in \{0,1,2\}^*$. We say that the *equivalence rule* $\mathsf{R} \leftrightarrow_L \mathsf{S}$ is valid if the replacement of the quantifier string R by the string S in any context does not change the class in question, i.e. $\mathsf{RQ}L^{\mu\tau(\mathsf{R})\tau(\mathsf{Q})} = \mathsf{SQ}L^{\mu\tau(\mathsf{S})\tau(\mathsf{Q})}$ for all $\mathsf{Q} \in \Gamma_{\log}^*$ and $\mu \in \{0,1,2\}^*$. Obviously, we have $\mathsf{R} \leftrightarrow_L \mathsf{S}$ if and only if $\mathsf{R} \to_L \mathsf{S}$ and $\mathsf{S} \to_L \mathsf{R}$.

The complementation observation is also valid for the rules "$\to_L$" and "$\leftrightarrow_L$".

Proposition 5.2 (Complementation). *Let* $\mathsf{R}, \mathsf{S} \in \Gamma_{\log}^*$. *If* $\mathsf{R} \to_L \mathsf{S}$ *then* $\overline{\mathsf{R}} \to_L \overline{\mathsf{S}}$.

Proof. The proof follows as in Proposition 3.3. ❑

Again, our first rules show relations between the existential (universal, respectively) quantifiers of different types.

Lemma 5.3. *The following inclusion rules are valid:*

(1) $\varepsilon \to_L \exists^{\log}$ *and* $\varepsilon \to_L \forall^{\log}$;

(2) $\exists^{\log} \to_L \exists^1$ *and* $\forall^{\log} \to_L \forall^1$;

(3) $\exists^1 \to_L \exists^2$ *and* $\forall^1 \to_L \forall^2$.

Proof. The proof follows as in Lemma 3.4. Let $\mathsf{Q} \in \Gamma_{\log}^*$ and $\mu \in \{0,1,2\}^*$.

(1) This is the classical case of introducing a dummy word quantifier.

(2) Only $\exists^{\log} \to_L \exists^1$ has to be proved, since $\forall^{\log} \to_L \forall^1$ follows by complementation. For a language $L \in \exists^{\log}Q\mathcal{L}^{\mu0\tau(Q)}$ there exist an $L_1 \in \mathcal{L}^{\mu0\tau(Q)}$ and a constant $c \in \mathbb{N}$ such that

$$\begin{aligned} X \in L &\iff \exists^{\log}u\, QY\,(|u| = c \cdot \log|X| \wedge (X, u, Y) \in L_1) \\ &\iff \exists^1 U\, QY\,((X, U, Y) \in L_2)\,, \end{aligned}$$

where $L_2 =_{\mathrm{df}} \{(X, U, Y) : (X, \langle U, c \cdot \log|X|\rangle, Y) \in L_1\}$. Let M be a logarithmic-space machine of type $\mu0\tau(Q)$ accepting L_1. Consider a machine M' of type $\mu1\tau(Q)$ that on input (X, U, Y) computes $c \cdot \log|X|$ and then $\langle U, c \cdot \log|X|\rangle$ by asking $1, 11, \ldots, 1^{c\,\log|X|}$ to the oracle U. Then, the machine M' works as machine M on input $(X, \langle U, c \cdot \log|X|\rangle, Y)$. Therefore, $L(M') = L_2$ and $L_2 \in \mathcal{L}^{\mu1\tau(Q)}$, i.e. $L \in \exists^1 Q\mathcal{L}^{\mu1\tau(Q)}$.

(3) This is obvious since a logarithmic-space machine of type $\mu1\tau(Q)$ can also be considered to be a machine of type $\mu2\tau(Q)$. ❑

The next lemma shows how to melt neighboured existential (universal, respectively) quantifiers.

Lemma 5.4. *For $\sigma \in \{\log, 1, 2\}$ the following equivalence rules are valid*

(1) $\exists^{\log}\exists^\sigma \leftrightarrow_L \exists^\sigma$ *and* $\forall^{\log}\forall^\sigma \leftrightarrow_L \forall^\sigma$;

(2) $\exists^\sigma\exists^{\log} \leftrightarrow_L \exists^\sigma$ *and* $\forall^\sigma\forall^{\log} \leftrightarrow_L \forall^\sigma$;

(3) $\exists^2\exists^\sigma \leftrightarrow_L \exists^2$ *and* $\forall^2\forall^\sigma \leftrightarrow_L \forall^2$;

(4) $\exists^\sigma\exists^2 \leftrightarrow_L \exists^2$ *and* $\forall^\sigma\forall^2 \leftrightarrow_L \forall^2$.

Proof. The proof follows as in Lemma 3.6. We prove the first rule of every statement, the other rules follow by complementation. By Lemma 5.3 it is enough to prove the following inclusion rules

(i) $\exists^{\log}\exists^{\log} \to_L \exists^{\log}$;

(ii) $\exists^{\log}\exists^1 \to_L \exists^1$ and $\exists^1\exists^{\log} \to_L \exists^1$;

(iii) $\exists^2\exists^2 \to_L \exists^2$.

In order to prove these inclusions, let $Q \in \Gamma^*_{\log}$ and $\mu \in \{0, 1, 2\}^*$.

(i) Let $L \in \exists^{\log}\exists^{\log}Q\mathcal{L}^{\mu00\tau(Q)}$. There exist an $L_1 \in \mathcal{L}^{\mu00\tau(Q)}$ and constants $c, k \in \mathbb{N}$ such that

$$\begin{aligned} X \in L &\iff \exists^{\log}u\, \exists^{\log}v\, QY\,(|u| = c \cdot \log|X| \wedge |v| = k \cdot \log|X| \wedge (X, u, v, Y) \in L_1) \\ &\iff \exists^{\log}w\, QY\,(|w| = 2 + 2 \cdot (c + k) \cdot \log|X| \wedge (X, w, Y) \in L_2)\,, \end{aligned}$$

where $L_2 =_{\mathrm{df}} \{(X, \widehat{u}01\widehat{v}, Y) : (X, u, v, Y) \in L_1\}$. Let M be a logarithmic-space machine of type $\mu00\tau(Q)$ accepting L_1, and let M' be a machine of type $\mu0\tau(Q)$ that on input (X, w, Y) computes u and v from $w = \widehat{u}01\widehat{v}$ (where it rejects if w does not have this form) and then works as M on input (X, u, v, Y). Therefore, $L(M') = L_2$ and $L_2 \in \mathcal{L}^{\mu0\tau(Q)}$, i.e. $L \in \exists^{\log}Q\mathcal{L}^{\mu0\tau(Q)}$.

(ii) It suffices to prove the first rule, because of the obvious rule $\exists^{\log}\exists^1 \leftrightarrow_L \exists^1\exists^{\log}$. For a language $L \in \exists^{\log}\exists^1 Q\mathcal{L}^{\mu01\tau(Q)}$ there exist an $L_1 \in \mathcal{L}^{\mu01\tau(Q)}$ and a constant $c \in \mathbb{N}$ such that

$$\begin{aligned} X \in L &\iff \exists^{\log} v\, \exists^1 U\, QY\, (|v| = c \cdot \log|X| \wedge (X, v, U, Y) \in L_1) \\ &\iff \exists^1 W\, QY\, ((X, W, Y) \in L_2)\,, \end{aligned}$$

where $L_2 =_{\mathrm{df}} \left\{(X, W, Y) : \left(X, \langle W, c \cdot \log|X|\rangle, 1^{1+c \log|X|}\backslash W, Y\right) \in L_1\right\}$. Let M be a logarithmic-space machine of type $\mu01\tau(Q)$ accepting L_1, and let M' be a machine of type $\mu1\tau(Q)$ that on input (X, W, Y) computes $c \cdot \log|X|$ and then $\langle W, c \cdot \log|X|\rangle$ by asking $1, 11, \ldots, 1^{c \log|X|}$ to W. Then, the machine M' works like machine M on input $\left(X, \langle W, c \cdot \log|X|\rangle, 1^{1+c \log|X|}\backslash W, Y\right)$ with the difference that instead of asking the query u to oracle $1^{1+c \log|X|}\backslash W$, the query $1^{1+c \log|X|}u$ is asked to W. Therefore, $L(M') = L_2$ and $L_2 \in \mathcal{L}^{\mu1\tau(Q)}$, i.e. $L \in \exists^1 Q\mathcal{L}^{\mu1\tau(Q)}$.

(iii) Let $L \in \exists^2\exists^2 Q\mathcal{L}^{\mu22\tau(Q)}$. There exists an $L_1 \in \mathcal{L}^{\mu22\tau(Q)}$ such that

$$\begin{aligned} X \in L &\iff \exists^2 U\, \exists^2 V\, QY\, ((X, U, V, Y) \in L_1) \\ &\iff \exists^2 W\, QY\, ((X, W, Y) \in L_2)\,, \end{aligned}$$

where $L_2 =_{\mathrm{df}} \{(X, W, Y) : (X, 0\backslash W, 1\backslash W, Y) \in L_1\}$. Let M be a logarithmic-space machine of type $\mu22\tau(Q)$ accepting L_1. Consider a machine M' of type $\mu2\tau(Q)$ working on input (X, W, Y) as M on input $(X, 0\backslash W, 1\backslash W, Y)$ with the difference that instead of asking the query w to oracle $0\backslash W$ (oracle $1\backslash W$), the query $0w$ ($1w$, respectively) to oracle W is asked. Therefore, $L(M') = L_2$ and $L_2 \in \mathcal{L}^{\mu2\tau(Q)}$, i.e. $L \in \exists^2 Q\mathcal{L}^{\mu2\tau(Q)}$. ❑

The next result shows how to shift a word quantifier followed by a set quantifier without diminishing the class in question.

Lemma 5.5. *For* $\sigma \in \{1, 2\}$ *the inclusion rules*

$$\exists^{\log}\forall^{\sigma} \rightarrow_L \forall^{\sigma}\exists^{\log} \quad \textit{and} \quad \forall^{\log}\exists^{\sigma} \rightarrow_L \exists^{\sigma}\forall^{\log}$$

are valid.

Proof. The proof is as in Lemma 3.7. We prove the second rule, the first follows by complementation. Let $Q \in \Gamma^*_{\log}$ and $\mu \in \{0, 1, 2\}^*$. For a language $L \in \forall^{\log}\exists^{\sigma} Q\mathcal{L}^{\mu0\sigma\tau(Q)}$ there exist an $L_1 \in \mathcal{L}^{\mu0\sigma\tau(Q)}$ and a constant $c \in \mathbb{N}$ such that

$$X \in L \iff \forall^{\log} u\, \exists^{\sigma} V\, QY\, (|u| = c \cdot \log|X| \rightarrow (X, u, V, Y) \in L_1)$$

Now, we define $L_2 =_{\mathrm{df}} \{(X, W, u, Y) : (X, u, u\backslash W, Y) \in L_1\}$ and we prove

$$X \in L \iff \exists^{\sigma} W\, \forall^{\log} u\, QY\, (|u| = c \cdot \log|X| \rightarrow (X, W, u, Y) \in L_2)$$

"$\Longrightarrow$": For every $u \in \{0, 1\}^{c \log|X|}$ let V_u be a set such that $(X, u, V_u, Y) \in L_1$ and define $W = \bigcup_{|u|=c \log|X|} \{uw : w \in V_u\}$. Then $(X, W, u, Y) \in L_2$ for every $u \in \{0, 1\}^{c \log|X|}$.

"$\Longleftarrow$": Let $(X, W, u, Y) \in L_2$ for every $u \in \{0,1\}^{c \log|X|}$. Hence, for each $u \in \{0,1\}^{c \log|X|}$ there exists a set $V_u =_{df} u\backslash W$ such that $(X, u, V_u, Y) \in L_1$.

Let M be a logarithmic-space machine of type $\mu 0 \sigma\tau(Q)$ accepting L_1, and let M' be a machine of type $\mu\sigma 0\tau(Q)$ working on input (X, W, u, Y) as M on input $(X, u, u\backslash W, Y)$ with the difference that instead of asking the query w to oracle $u\backslash W$ the query uw is asked to W. Since M asks $u\backslash W$ in a type σ manner, the machine M' does it as well. Therefore, $L(M') = L_2$ and $L_2 \in \mathcal{L}^{\mu\sigma 0\tau(Q)}$, i.e. $L \in \exists^\sigma\forall^{\log} Q\mathcal{L}^{\mu\sigma 0\tau(Q)}$. ❏

Combining previous results we obtain the following equivalence rules which show how to eliminate word quantifiers.

Corollary 5.6. *For* $k \geq 1$, $\sigma_1, \sigma_2, \dots, \sigma_k \in \{1,2\}$ *and* $\tau \in \{\log, 1, 2\}$ *the equivalence rules*

$$\exists^{\log}\forall^{\sigma_1}\forall^{\sigma_2}\dots\forall^{\sigma_k}\exists^{\tau} \leftrightarrow_{\mathcal{L}} \forall^{\sigma_1}\forall^{\sigma_2}\dots\forall^{\sigma_k}\exists^{\tau} \quad \textit{and}$$
$$\forall^{\log}\exists^{\sigma_1}\exists^{\sigma_2}\dots\exists^{\sigma_k}\forall^{\tau} \leftrightarrow_{\mathcal{L}} \exists^{\sigma_1}\exists^{\sigma_2}\dots\exists^{\sigma_k}\forall^{\tau}$$

are valid.

Proof. By rules of Lemmas 5.4 and 5.5 we can conclude the rule $\exists^{\log}\forall^{\sigma_1}\dots\forall^{\sigma_k}\exists^{\tau} \rightarrow_{\mathcal{L}} \forall^{\sigma_1}\dots\forall^{\sigma_k}\exists^{\tau}$. The rule $\forall^{\sigma_1}\dots\forall^{\sigma_k}\exists^{\tau} \rightarrow_{\mathcal{L}} \exists^{\log}\forall^{\sigma_1}\dots\forall^{\sigma_k}\exists^{\tau}$ is valid by Lemma 5.3. ❏

Next, we prove "equivalence rules" which are valid only in a special context. In the previous result was shown how to eliminate word quantifiers. Another way to do that is stated in the following result.

Proposition 5.7. *Let* $Q \in \left\{\exists^{\log}, \forall^{\log}\right\}^*$ *and* $\mu \in \{0,2\}^*$. *Then* $Q\mathcal{L}^{\mu\tau(Q)} = \mathcal{L}^{\mu}$.

Proof. The inclusion "$\supseteq$" is valid by Lemma 5.3. The inclusion $Q\mathcal{L}^{\mu\tau(Q)} \subseteq \mathcal{L}^{\mu}$ is evident, since the word quantifiers Q, which vary over words whose lengths are logarithmically bounded in the length of the input, can easily be simulated by an $\mathcal{L}$-computation. ❏

The next lemma says that a set quantifier of type 1 is exactly as powerful as the corresponding word quantifier when applied to $\mathcal{L}$.

Lemma 5.8. *Let* $\mu \in \{0,1,2\}^*$. *Then*

$$\exists^{\log}\mathcal{L}^{\mu 0} = \exists^1\mathcal{L}^{\mu 1} \quad \textit{and} \quad \forall^{\log}\mathcal{L}^{\mu 0} = \forall^1\mathcal{L}^{\mu 1}$$

Proof. We prove the first statement, the second follows by complementation. The inclusion "$\subseteq$" is valid by Lemma 5.3, thus only $\exists^1\mathcal{L}^{\mu 1} \subseteq \exists^{\log}\mathcal{L}^{\mu 0}$ has to be proved. By definition $L \in \exists^1\mathcal{L}^{\mu 1}$ if and only if there exists an $L' \in \mathcal{L}^{\mu 1}$, such that $X \in L \iff \exists^1 U\,((X, U) \in L')$. Let M be a logarithmic-space machine of type $\mu 1$ accepting L'. Without loss of generality we assume that M does not make a query twice. Since no query of M on input (X, U) to oracle U is longer than $c \cdot \log|X|$ for suitable constant $c \in \mathbb{N}$ and this oracle is of type 1, U is queried at most $c \cdot \log|X|$ times. Let M' be a machine of type $\mu 0$ working on input (X, u) as M on input (X, U) with the following difference: Instead of the answer of U to the i-th query of M the machine M' uses the i-th bit of u. Now, $\exists^1 U\,((X, U) \in L') \iff \exists^{\log} u\,(|u| = c \cdot \log|X| \wedge (X, u) \in L(M'))$ can be seen as in Lemma 3.5. Hence, $L(M') \in \mathcal{L}^{\mu 0}$ and $L \in \exists^{\log}\mathcal{L}^{\mu 0}$. ❏

5.2.2 A Normal Form Theorem

Now, we are ready to state the fact that every class of the analytic logarithmic-space hierarchy can be represented in a certain normal form. For $l \geq 1$, let $Q_l =_{\text{df}} \exists$ if l is odd and $Q_l =_{\text{df}} \forall$ otherwise. For $k, m \geq 0$, $\tau_1, \tau_2, \tau_3, \ldots, \tau_k \in \{1, 2\}$ and $l_1, l_2, l_3, \ldots, l_k \in \mathbb{N}$ we define

$$\mathcal{K}_L\big(\langle\tau_1\rangle_{l_1}\,\langle\tau_2\rangle_{l_2}\,\langle\tau_3\rangle_{l_3}\ldots\langle\tau_k\rangle_{l_k}, m\big) =_{\text{df}} \underbrace{\exists^{\tau_1}\ldots\exists^{\tau_1}}_{l_1\text{ times}}\underbrace{\forall^{\tau_2}\ldots\forall^{\tau_2}}_{l_2\text{ times}}\underbrace{\exists^{\tau_3}\ldots\exists^{\tau_3}}_{l_3\text{ times}}\ldots\underbrace{Q_k^{\tau_k}\ldots Q_k^{\tau_k}}_{l_k\text{ times}}\,Q_{k+1}^{\log}\ldots Q_{k+m}^{\log}L$$

and

$$\mathcal{K}_L^2\big(\langle\tau_1\rangle_{l_1}\,\langle\tau_2\rangle_{l_2}\,\langle\tau_3\rangle_{l_3}\ldots\langle\tau_k\rangle_{l_k}, m\big) =_{\text{df}} \underbrace{\exists^{\tau_1}\ldots\exists^{\tau_1}}_{l_1\text{ times}}\underbrace{\forall^{\tau_2}\ldots\forall^{\tau_2}}_{l_2\text{ times}}\underbrace{\exists^{\tau_3}\ldots\exists^{\tau_3}}_{l_3\text{ times}}\ldots\underbrace{Q_k^{\tau_k}\ldots Q_k^{\tau_k}}_{l_k\text{ times}}\,Q_{k+1}^{\log}\ldots Q_{k+m}^{\log}Q_{k+m+1}^{2}L$$

We also write τ instead of $\langle\tau\rangle_1$, $(\langle\tau\rangle_i)^j$ instead of $\underbrace{\langle\tau\rangle_i\ldots\langle\tau\rangle_i}_{j\text{ times}}$ and τ^j instead of $\underbrace{\tau\ldots\tau}_{j\text{ times}}$.

Theorem 5.9 (Normal Form Theorem). *Every class of the analytic logarithmic-space hierarchy ALH coincides with one of the classes L, $\mathcal{K}_L(\tau, m)$, co$\mathcal{K}_L(\tau, m)$, $\mathcal{K}_L^2(\tau, n)$ or co$\mathcal{K}_L^2(\tau, n)$, where $\tau \in (\{2\} \cup \{\langle 1\rangle_l : l \geq 1\})^+$, $m \geq 1$ and $n \geq 0$.*

Proof. Let us first prove the chain for $\tau \in (\{2\} \cup \{\langle 1\rangle_l : l \geq 1\})^*$. Consider an arbitrary class of the analytic logarithmic-space hierarchy. If this class is defined without quantifiers, then it is L. Otherwise it coincides with one of the classes $\mathcal{K}_L(\tau, m)$, co$\mathcal{K}_L(\tau, m)$, $\mathcal{K}_L^2(\tau, n)$ or co$\mathcal{K}_L^2(\tau, n)$ with $\tau \in (\{2\} \cup \{\langle 1\rangle_l : l \geq 1\})^*$, $m \geq 1$ and $n \geq 0$. This can be seen as follows: By Lemma 5.4, we bring the quantifier prefix in a form, where no quantifier substring $\exists^{\tau_1}\exists^{\tau_2}$ or $\forall^{\tau_1}\forall^{\tau_2}$ appears with $\tau_1, \tau_2 \in \{\log, 1, 2\}$ and either $\tau_1 \neq 1$ or $\tau_2 \neq 1$, i.e. $\exists^1\exists^1$ or $\forall^1\forall^1$, can appear. By Lemmas 5.8 and 5.4 a quantifier string $\exists^1\ldots\exists^1$ or $\forall^1\ldots\forall^1$ can be replaced by the corresponding word quantifier when applied to L. Hence, we ensure that after the last alternation in the sequence of $\exists$-$\forall$-quantifiers only either a word quantifier or a set quantifier of type 2 occurs. By Corollary 5.6, we eliminate all word quantifiers which are followed by a set quantifier not being of type 2 applied to L. This last step can generate quantifier substrings $\exists^{\tau_1}\exists^{\tau_2}$ or $\forall^{\tau_1}\forall^{\tau_2}$ with $\tau_1 \in \{\log, 1, 2\}$, $\tau_2 \in \{1, 2\}$ and either $\tau_1 \neq 1$ or $\tau_2 \neq 1$. However, applying repeatedly the rules of Lemma 5.4 and Corollary 5.6 we get the desired result.

Now, we show that it is enough to consider $\tau \in (\{2\} \cup \{\langle 1\rangle_l : l \geq 1\})^+$. We conclude

$$\mathcal{K}_L(\varepsilon, m) = L \qquad \text{by Proposition 5.7}$$

and

$$\mathcal{K}_L^2(\varepsilon, n) \subseteq \begin{cases} \mathcal{K}_L(2,1) & \text{if } n \text{ is even} \\ \text{co}\mathcal{K}_L(2,1) & \text{if } n \text{ is odd} \end{cases} \qquad \text{by Lemmas 5.5, 5.4 and 5.3}$$

$$\subseteq \mathcal{K}_L^2(\varepsilon, n) \qquad \text{by Proposition 5.7}$$

❑

5.3 Characterizing the Classes $\mathcal{K}_L(\cdot)$ and co$\mathcal{K}_L(\cdot)$

We will characterize the classes having the form $\mathcal{K}_L(\tau, m)$ or co$\mathcal{K}_L(\tau, m)$ by well-known complexity classes, where $\tau \in (\{2\} \cup \{\langle 1\rangle_l : l \geq 1\})^+$ and $m \geq 1$. The simplest classes of these types are those containing only one set quantifier of type 1 and one word quantifier, i.e. $\mathcal{K}_L(1,1) = \exists^1\forall^{\log}L$ and co$\mathcal{K}_L(1,1) = \forall^1\exists^{\log}L$. In a personal communication, Allender pointed out that these classes coincide with L.

Theorem 5.10. [All97] $\exists^1\forall^{\log}L = \forall^1\exists^{\log}L = L$.

Proof. Since L is closed under complementation, only $\exists^1\forall^{\log}L = L$ has to be proved. The inclusion "$\supseteq$" is validated by Lemma 5.3. For the other inclusion, the idea is the following: If we follow the proof of $\exists^1\forall^p P \subseteq PSPACE$ (Theorem 3.9) we can conclude $\exists^1\forall^{\log}L \subseteq NL$. However, Allender pointed out that the guesses of the machine M to the answers of the oracle U on path π can be replaced by a double recursion. Hence, $\exists^1\forall^{\log}L \subseteq L$. ❑

We next define a problem which is *DLOGSPACE*-complete for Σ_k^p. Let B_k be the following problem:

Given: A boolean expression ϕ with boolean variables partitioned into k sets $X_1, \ldots, X_k$.

Question: Is the expression $\exists X_1 \forall X_2 \exists X_3 \ldots Q_k X_k\ (\phi(X_1, X_2, \ldots, X_k) \equiv 1)$ true?

where, as usual, $Q_l =_{df} \exists$ if l is odd and $Q_l =_{df} \forall$ otherwise, and $\exists X_i$ ($\forall X_i$) means there exists an (for all, respectively) assignment for the variables in X_i.

Lemma 5.11. [Wra77] *The B_k problem is DLOGSPACE-complete for Σ_k^p with the expression ϕ being in conjunctive normal-form if k is odd and in disjunctive normal-form otherwise.*

It turns out that the remainder classes $\mathcal{K}_L(\cdot)$ and co$\mathcal{K}_L(\cdot)$ which we have not already been characterized coincide with classes of the (arithmetic) polynomial-time hierarchy. The following theorem shows which classes of the analytic logarithmic-space hierarchy contain a level of the (arithmetic) polynomial-time hierarchy.

Theorem 5.12. *For $k \geq 1$ the inclusions*

$$\Sigma_k^p \subseteq \mathcal{K}_L\big(1^k, 2\big) \cap \mathcal{K}_L\big(1^{k-1}\langle 1\rangle_2, 1\big) \cap \mathcal{K}_L\big(1^{k-1}2, 1\big) \quad \textit{and}$$
$$\Pi_k^p \subseteq \text{co}\mathcal{K}_L\big(1^k, 2\big) \cap \text{co}\mathcal{K}_L\big(1^{k-1}\langle 1\rangle_2, 1\big) \cap \text{co}\mathcal{K}_L\big(1^{k-1}2, 1\big)$$

are valid.

Proof. We prove the first inclusion, the second follows by complementation. Consider the case that k is odd.

"$\Sigma_k^p \subseteq \mathcal{K}_L\big(1^k, 2\big)$": We will prove $B_k \in \exists^1\forall^1 \ldots \exists^1\forall^{\log}\exists^{\log}L^{01^k00}$ (Lemma 5.11). Let ϕ be a boolean expression with boolean variables partitioned into k sets $X_1, \ldots, X_k$ (ϕ is in a conjunctive normal-form). The idea is the following: The i-th oracle contains a variable

u of set X_i if and only if "u is true". Thus, for each clause ($\forall$-word quantifier) it will be verified if there exists a literal ($\exists$-word quantifier) which makes this clause true. Formally, let M be a machine of type 01^k00 working on input $(\phi, U_1, \ldots, U_k, u, w)$ as follows (if ϕ is not in a conjunctive normal-form then M rejects the input): M checks whether the literal w or $\overline{w}$ appears in the u-th clause, rejects the input in a negative case and accepts the input if both appear in the u-th clause. Otherwise (i.e. either w or $\overline{w}$ appears in the u-th clause), choose $i \in \{1, \ldots, k\}$ such that $w \in X_i$. Then, M accepts the input if and only if w appears in the u-th clause and $w \in U_i$ or $\overline{w}$ appears in the u-th clause and $w \notin U_i$. Therefore,

$$\phi \in B_k \iff \exists^1 U_1 \forall^1 U_2 \ldots \exists^1 U_k \forall^{\log} u \, \exists^{\log} w \, ((\phi, U_1, U_2, \ldots, U_k, u, w) \in L(M))$$

and $L(M) \in \mathcal{L}^{01^k00}$, i.e. $B_k \in \mathcal{K}_L(1^k, 2)$.

"$\Sigma_k^p \subseteq \mathcal{K}_L(1^{k-1}\langle 1\rangle_2, 1)$": Consider a machine M' of type $01^{k+1}0$ working on input $(\phi, U_1, \ldots, U_{k+1}, u)$ as follows: First, it asks $u0^1, u0^2, \ldots, u0^{\lceil \log n \rceil}$ to U_{k+1}, where n is the number of boolean variables in ϕ. Let w be the sequence of answers and M be the machine of the previous case. Then M' works as M on input $(\phi, U_1, \ldots, U_k, u, w)$. Therefore,

$$\phi \in B_k \iff \exists^1 U_1 \forall^1 U_2 \ldots \exists^1 U_k \, \exists^1 U_{k+1} \, \forall^{\log} u \, ((\phi, U_1, U_2, \ldots, U_{k+1}, u) \in L(M'))$$

and $L(M') \in \mathcal{L}^{01^{k+1}0}$, i.e. $B_k \in \mathcal{K}_L(1^{k-1}\langle 1\rangle_2, 1)$.

"$\Sigma_k^p \subseteq \mathcal{K}_L(1^{k-1}2, 1)$": By the previous inclusion and Lemmas 5.3 and 5.4 it follows that $\Sigma_k^p \subseteq \mathcal{K}_L(1^{k-1}\langle 1\rangle_2, 1) \subseteq \mathcal{K}_L(1^{k-1}2, 1)$.

In the case that k is even, $\Pi_k^p \subseteq \mathrm{co}\mathcal{K}_L(1^k, 2) \cap \mathrm{co}\mathcal{K}_L(1^{k-1}\langle 1\rangle_2, 1) \cap \mathrm{co}\mathcal{K}_L(1^{k-1}2, 1)$ can be proved in the same way which yields the desired result by complementation. ❑

The next theorem shows which classes of the analytic logarithmic-space hierarchy are included in a level of the (arithmetic) polynomial-time hierarchy.

Theorem 5.13. *For $k \geq 1$ and $m \geq 0$ the inclusions*

$$\mathcal{K}_L(2^k, m) \cup \mathcal{K}_L(2^k1, 1) \subseteq \Sigma_k^p \quad \textit{and} \quad \mathrm{co}\mathcal{K}_L(2^k, m) \cup \mathrm{co}\mathcal{K}_L(2^k1, 1) \subseteq \Pi_k^p$$

are valid.

Proof. We prove the first statement, the second follows by complementation. For $l \geq 1$, let $Q_l =_{\mathrm{df}} \exists$ if l is odd and $Q_l =_{\mathrm{df}} \forall$ otherwise. By replacing oracle queries by consuming inputs bits, the following inclusion is evident (see Proposition 3.12)

$$\exists^2\forall^2\exists^2 \ldots Q_k^2 \mathcal{L} \subseteq \exists^p\forall^p\exists^p \ldots Q_k^p \mathcal{P} \tag{5.1}$$

Hence,

"$\mathcal{K}_L\big(2^k, m\big) \subseteq \Sigma_k^p$": We conclude

$$\begin{aligned}
\exists^2\forall^2\exists^2 \dots Q_k^2 Q_{k+1}^{\log} \dots Q_{k+m}^{\log} L &\subseteq \exists^2\forall^2\exists^2 \dots Q_k^2 L && \text{by Proposition 5.7}\\
&\subseteq \exists^p\forall^p\exists^p \dots Q_k^p P && \text{by Equation (5.1)}\\
&\subseteq \Sigma_k^p
\end{aligned}$$

"$\mathcal{K}_L\big(2^k 1, 1\big) \subseteq \Sigma_k^p$": The proof of $\exists^1\forall^{\log} L \subseteq L$ (Theorem 5.10) remains valid if the machines have additionally k oracles of type 2, i.e. $\exists^1\forall^{\log} L^{02^k 10} \subseteq L^{02^k}$. Therefore,

$$\begin{aligned}
\exists^2\forall^2\exists^2 \dots Q_k^2 Q_{k+1}^1 Q_{k+2}^{\log} L &\subseteq \exists^2\forall^2\exists^2 \dots Q_k^2 L\\
&\subseteq \exists^p\forall^p\exists^p \dots Q_k^p P && \text{by Equation (5.1)}\\
&\subseteq \Sigma_k^p
\end{aligned}$$

where we use the base $\forall^1\exists^{\log} L \subseteq L$ if k is odd. ❑

The following theorem summarizes the results showing a complete characterizations for the classes $\mathcal{K}_L(\tau, m)$ or co$\mathcal{K}_L(\tau, m)$ by well-known complexity classes, where $\tau \in (\{2\} \cup \{\langle 1\rangle_l : l \geq 1\})^+$ and $m \geq 1$.

Theorem 5.14. *Let* $k, m \geq 1$ *and* $\tau_1, \dots, \tau_k \in \{2\} \cup \{\langle 1\rangle_l : l \geq 1\}$. *Then*

$$\mathcal{K}_L(\tau_1 \dots \tau_k, m) = \begin{cases} L & \textit{if } k = 1, \tau_k = 1 \textit{ and } m = 1,\\ \Sigma_{k-1}^p & \textit{if } k \geq 2, \tau_k = 1 \textit{ and } m = 1,\\ \Sigma_k^p & \textit{if } k \geq 1, \tau_k = \langle 1\rangle_l, l \geq 2, \textit{and } m = 1,\\ \Sigma_k^p & \textit{if } k \geq 1, \tau_k = 2, \textit{and } m = 1,\\ \Sigma_k^p & \textit{if } k \geq 1 \textit{ and } m \geq 2. \end{cases}$$

Proof. The first line is valid by Theorem 5.10. By Lemmas 5.3 and 5.4 it follows that a substring of existential (universal) quantifiers of type 1 can be replaced by an existential (universal, respectively) quantifier of type 2 without diminish the class in question. Hence, the remaining lines follow by Lemma 5.3 and Theorems 5.12 and 5.13. ❑

5.4 Characterizing Classes $\mathcal{K}_L^2(\cdot)$ and co$\mathcal{K}_L^2(\cdot)$

In this section, we characterize classes having the form $\mathcal{K}_L^2(\tau, n)$ or co$\mathcal{K}_L^2(\tau, n)$ by well-known complexity classes, where $\tau \in (\{2\} \cup \{\langle 1\rangle_l : l \geq 1\})^+$ and $n \geq 0$. The simplest classes of these types are those containing an $\exists$-$\forall$-alternate sequence of two set quantifiers where the first is of type 1 and the second is of type 2, i.e. $\mathcal{K}_L^2(1, 0) = \exists^1\forall^2 L$ and co$\mathcal{K}_L^2(1, 0) = \forall^1\exists^2 L$, which turn out to coincide with co*NP* and *NP*, respectively.

Theorem 5.15. $\exists^1\forall^2 L =$ co*NP* *and* $\forall^1\exists^2 L =$ *NP*.

Proof. We prove the first statement, the second follows by complementation. By Theorem 5.12 and Proposition 5.7 follows co*NP* $\subseteq \forall^2\exists^{\log} L \subseteq \forall^2 L$ which is include in $\exists^1\forall^2 L$ by

Lemma 5.3. Thus, it remains to prove $\exists^1\forall^2 L \subseteq$ co*NP*. Let $L \in \exists^1\forall^2 L$. There exists an $L_1 \in L^{012}$ such that $x \in L \iff \exists^1 U \forall^2 W((x, U, W) \in L_1)$. Let M_1 be a logarithmic-space machine of type 012 accepting L_1. Therefore, no query of M_1 on input (x, U, W) to oracles U and W is longer than $c \cdot \log|x|$ for a suitable constant c. For a given input (x, U, W), it is important that M_1 asks the oracle U only queries from one *oracle path* u_1, $u_1u_2, \ldots, u_1u_2\ldots u_{c\ \log|x|}$. Let M be a machine of type 0000 that on input (x, u_p, u_a, z) works as M_1 on input(x, U, W) with the following differences:

(a) When the machine M_1 asks a query to oracle U from oracle path u_p then M uses the oracle answer encoded in u_a. The machine M stops the simulation if a query is asked to U which is not from the oracle path u_p.

(b) Instead of the answer of W to query w of M_1 the machine M uses the i-th bit of z, where $w = \text{lex}(i)$.

The machine M accepts if and only if the simulation is stopped in (a) or the simulation of M_1 ends accepting. Obviously, $L(M) \in P^{0000}$ and for a suitable polynomial p

$$\begin{aligned} x \in L \iff & \forall^{\log} u_p\, \exists^{\log} u_a\, \forall^p z\big(|u_p| = c \cdot \log|x| \to (|u_a| = c \cdot \log|x| \\ & \qquad\qquad \wedge (|z| = p(|x|) \to (x, u_p, u_a, z) \in L(M)))\big) \\ \iff & \forall^{\log} u_p\, \forall^p y\, \big(\big(|u_p| = c \cdot \log|x| \wedge |y| = 2^{c\ \log|x|} \cdot p(|x|)\big) \to (x, y) \in L(M_2)\big) \end{aligned}$$

where M_2 is a machine of type 000 working on input (x, u_p, y) as follows (co*NP* is closed under $\vee$ and $\wedge$): M_2 considers step by step the words u_a in lexicographical order. For each $u_a \in \{0, 1\}^{c\ \log|x|}$ (see Figure 5.1):

(1) Choose i such that $u_a = \text{lex}_{c\ \log|x|}(i)$ and let z be the i-th fragment of size $p(|x|)$ of the word y.

(2) Now M_2 simulates M on input (x, u_p, u_a, z). If the simulation ends accepting, then M_2 accepts the input.

If M_2 has not accepted the input in (2), then the input is rejected. This shows $L \in \forall^{\log}\forall^p P$ which is included in co*NP*. ❑

The following result shows which classes of the analytic logarithmic-space hierarchy whose last set quantifier is of type 2 contain a level of the (arithmetic) polynomial-time hierarchy.

Theorem 5.16. *For* $k \geq 2$ *the inclusions*

$$\Sigma_k^p \subseteq \mathcal{K}_L^2(1^{k-2}2, 0) \quad \textit{and} \quad \Pi_k^p \subseteq \text{co}\mathcal{K}_L^2(1^{k-2}2, 0)$$

are valid.

Proof. We prove the first inclusion, the second follows by complementation. Consider the case that k is even. We will prove $B_k \in \exists^1\forall^1 \ldots \forall^1\exists^2\forall^2 L^{01^{k-2}22}$ (Lemma 5.11). Let ϕ be a boolean expression with boolean variables partitioned into k sets $X_1, \ldots, X_k$ (ϕ

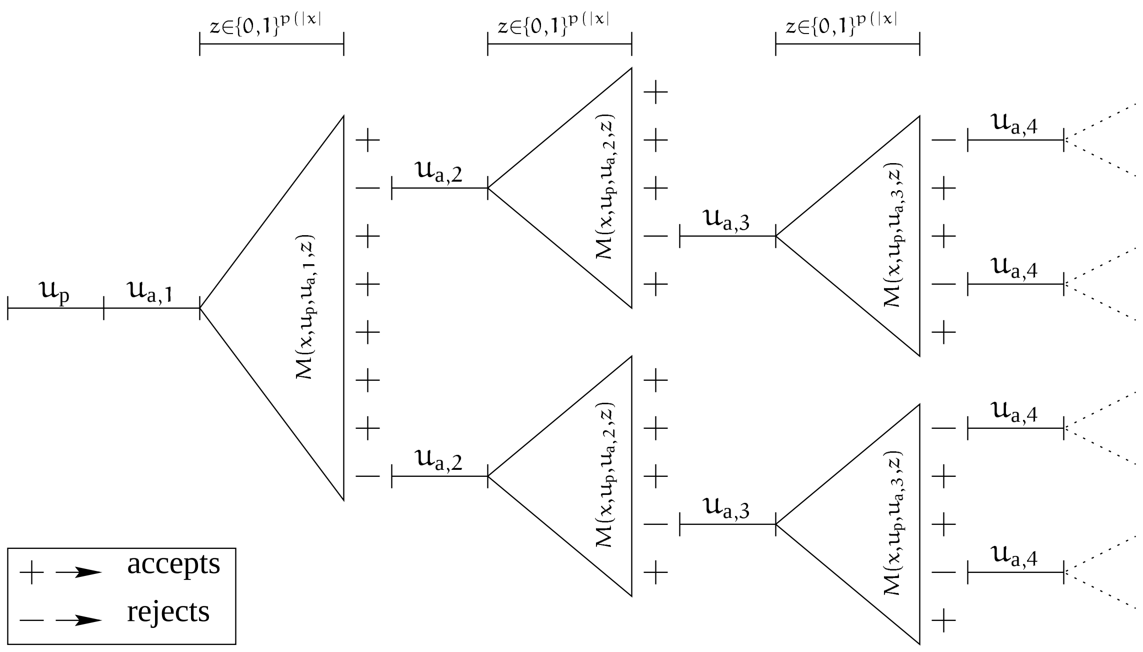


Figure 5.1: *Simulation of machine* M_2 *on input* (x, u_p, y) *for all* $y \in \{0,1\}^{2^{c\cdot\log|x|}\,p(|x|)}$. *For short we write* $u_{a,i}$ *instead of* $\mathrm{lex}_{c\ \log|x|}(i)$.

is in disjunctive normal-form). The idea is similar to the proof of Theorem 5.12: The i-th oracle contains a variable u of set X_i if and only if "u is true". We will construct a machine which may query for two or more variables of a set X_i (that is not a type 1 querying). To overcome this difficulty, the oracle bounded by the $\exists^2$ quantifier will also reflect the $k-2$ first oracles, i.e. the oracles of type 1. Formally, let M be a machine of type $01^{k-2}22$ working on input $(\phi, U_1, \ldots, U_k)$ as follows (if ϕ is not in disjunctive normal-form then M rejects the input):

"$1 \in U_k$": M checks whether U_{k-1} reflects the $k-2$ first oracles. Let u be the sequence of answers of U_k to the queries $1^2, 1^3, \ldots, 1^{1+\lceil\log n\rceil}$, where n is the number of boolean variables in ϕ. Then, M accepts the input if and only if for $i = 1, \ldots, k-2$

$$u \in U_i \longleftrightarrow \widehat{\mathrm{lex}(i)}01u \in U_{k-1}$$

"$1 \notin U_k$": M checks whether a clause in ϕ is satisfiable. The machine M accepts the input if and only if there exists a clause $(l_1 \wedge \cdots \wedge l_m)$ in ϕ such that for $j = 1, \ldots, m$

$$1 \le i < k \longrightarrow \begin{cases} \widehat{\mathrm{lex}(i)}01u \in U_{k-1} & \text{if } l_j = u, \\ \widehat{\mathrm{lex}(i)}01u \notin U_{k-1} & \text{if } l_j = \overline{u}, \end{cases}$$

$$i = k \longrightarrow \begin{cases} 0u \in U_k & \text{if } l_j = u, \\ 0u \notin U_k & \text{if } l_j = \overline{u}, \end{cases}$$

where $l_j = u$ or $l_j = \overline{u}$ for a variable u in ϕ, and $i \in \{1, \ldots, k\}$ such that $u \in X_i$.

Hence, the oracles $U_1, \dots, U_{k-2}$ are queried in a type 1 manner and

$$\phi \in B_k \iff \exists^1 U_1 \forall^1 U_2 \dots \forall^1 U_{k-2} \exists^2 U_{k-1} \forall^2 U_k \left((\phi, U_1, U_2, \dots, U_k) \in L(M)\right)$$

Therefore, $L(M) \in L^{01^{k-2}22}$, i.e. $B_k \in \mathcal{K}_L^2\big(1^{k-2}2, 0\big)$.

In the case that k is odd, $\Pi_k^p \subseteq \mathrm{co}\mathcal{K}_L^2\big(1^{k-2}2, 0\big)$ can be proved in the same way which yields the desired result by complementation. ❑

The next theorem shows which classes of the analytic logarithmic-space hierarchy whose last set quantifier is of type 2 are included in a level of the (arithmetic) polynomial-time hierarchy.

Theorem 5.17. *For* $k \geq 1$ *the inclusions*

$$\mathcal{K}_L^2\big(2^k 1, 0\big) \subseteq \Sigma_k^p \quad \textit{and} \quad \mathrm{co}\mathcal{K}_L^2\big(2^k 1, 0\big) \subseteq \Pi_k^p$$

are valid.

Proof. We prove the first statement, the second follows by complementation. For $l \geq 1$, let $Q_l =_{\mathrm{df}} \exists$ if l is odd and $Q_l =_{\mathrm{df}} \forall$ otherwise. By replacing oracle queries by consuming inputs bits (see Proposition 3.12), the proof of $\exists^1\forall^2 L \subseteq \forall^p P$ (Theorem 5.15) remains valid if the machines have additionally k oracles of type 2 and k polynomially length bounded word quantifiers, respectively, i.e. $\exists^1\forall^2 L^{02^k12} \subseteq \forall^p P^{00^k0}$. Therefore,

$$\begin{aligned}
\mathcal{K}_L^2\big(2^k 1, 0\big) &\subseteq \exists^2\forall^2\exists^2 \dots Q_k^2 Q_{k+1}^1 Q_{k+2}^2 L \\
&\subseteq \exists^p\forall^p\exists^p \dots Q_k^p Q_{k+2}^p P \\
&\subseteq \exists^p\forall^p\exists^p \dots Q_k^p P \qquad\qquad \text{by Lemma 3.6} \\
&\subseteq \Sigma_k^p
\end{aligned}$$

where we use the base $\forall^1\exists^2 L \subseteq \exists^p P$ if k is odd. ❑

Next, we characterize classes having the form $\mathcal{K}_L^2(\tau, n)$ or co$\mathcal{K}_L^2(\tau, n)$ by well-known complexity classes, where $\tau \in \left(\{2\} \cup \{\langle 1 \rangle_l : l \geq 1\}\right)^+$ and $n \geq 0$.

Theorem 5.18. *Let* $k \geq 1, \tau_1, \dots, \tau_k \in \{2\} \cup \{\langle 1 \rangle_l : l \geq 1\}$ *and* $n \geq 0$*. Then*

$$\mathcal{K}_L^2(\tau_1 \dots \tau_k, n) = \begin{cases} \Sigma_k^p & \textit{if } n \textit{ is odd,} \\ \mathrm{co}NP & \textit{if } n = 0,\, k = 1 \textit{ and } \tau_k = 1, \\ \Sigma_{k-1}^p & \textit{if } n = 0,\, k \geq 2 \textit{ and } \tau_k = 1, \\ \Sigma_{k+1}^p & \textit{if } n \textit{ is even and } \tau_k = 2. \end{cases}$$

Proof. For $l \geq 1$, let $Q_l =_{\mathrm{df}} \exists$ if l is odd and $Q_l =_{\mathrm{df}} \forall$ otherwise. For the first line, we conclude (if n is odd, then $Q_k = Q_{k+n+1}$)

$$\begin{aligned}
\mathcal{K}_L^2(\tau_1 \ldots \tau_k, n) &\subseteq \exists^2\forall^2\exists^2 \ldots Q_k^2 Q_{k+1}^{\log} \ldots Q_{k+n}^{\log} Q_{k+n+1}^2 L && \text{by Lemmas 5.3 and 5.4}\\
&\subseteq \exists^2\forall^2\exists^2 \ldots Q_k^2 Q_{k+n+1}^2 Q_{k+1}^{\log} L && \text{by Lemmas 5.4 and 5.5}\\
&\subseteq \exists^2\forall^2\exists^2 \ldots Q_k^2 Q_{k+1}^{\log} L && \text{by Lemma 5.4}\\
&\subseteq \Sigma_k^p && \text{by Theorem 5.13}\\
&\subseteq \mathcal{K}_L\big(1^k, 2\big) && \text{by Theorem 5.12}\\
&\subseteq \mathcal{K}_L^2(\tau_1 \ldots \tau_k, n) && \text{by Lemma 5.3}
\end{aligned}$$

The second line is valid by Theorem 5.15. For the remaining lines the direction $\subseteq$ is valid by Lemmas 5.3, 5.4 and 5.5 and Theorems 5.17 and 5.13. The direction $\supseteq$ is valid by Lemma 5.3 and Theorems 5.12 and 5.16. ❑

5.5 Conclusions

In §5.2 we showed that every class of the analytic logarithmic-space hierarchy can be represented in a certain normal form, where the last quantifier is either a word quantifier or a set quantifier of type 2. Next, we showed that each class in this normal form, whose last quantifier is a word quantifier, coincides with L or one of the classes Σ_k^p and Π_k^p ($k \geq 1$) of the (arithmetic) polynomial-time hierarchy and vice versa (§5.3). The classes in this normal form, whose last quantifier is a set quantifier of type 2, were examined in §5.4. However, the following cases are still open for $k \geq 1$, $\tau_1, \ldots, \tau_{k-1} \in \{2\} \cup \{\langle 1 \rangle_l : l \geq 1\}$ and $n \geq 0$ being even (see Theorem 5.18): $\mathcal{K}_L^2(\tau_1 \ldots \tau_k, n)$ and co$\mathcal{K}_L^2(\tau_1 \ldots \tau_k, n)$ such that $n \geq 2$ and $t_k = 1$, or $t_k = \langle 1 \rangle_l$ with $l \geq 2$.

We have proved $\exists^1\forall^2 L = $ co*NP* (Theorem 5.15). However, the class $\exists^1\exists^1\forall^2 L$ (an open case) is probably more powerful. This is shown in the following simple observation.

Proposition 5.19. $\Theta_2^p \subseteq \exists^1\exists^1\forall^2 L \subseteq \Sigma_2^p$.

Proof. The last inclusion follows by Lemmas 5.3 and 5.4 and Theorem 5.13. Thus, it remains to prove $\Theta_2^p \subseteq \exists^1\exists^1\forall^2 L$. We use Wagner's characterization of $\Theta_2^p = NP\big(n^{O(1)}\big)$ [Wag90]. For a language $L \in \Theta_2^p$ there exist an $L' \in NP$ and a polynomial p such that $(x, j+1) \in L' \rightarrow (x, j) \in L'$ for all j and

$$x \in L \Longleftrightarrow \max\{i : 1 \leq i \leq p(|x|) \text{ and } (x, i) \in L'\} \equiv 1 \mod 2$$

The max-function returns an i such that $(x, i) \in L'$ and $(x, i) \notin L'$. Furthermore, $(x, k) \in L'$ for all $1 \leq k \leq i$ and $(x, l) \notin L'$ for all $l > i$. Thus, giving an i we can verify if it is the maximum by an (*NP* $\wedge$ co*NP*)-computation. By Lemma 5.3, Proposition 5.7 and Theorem 5.12 follow $NP \subseteq \exists^1\exists^1\forall^{\log} L$ and co$NP \subseteq \forall^2 L$. Therefore, $NP \wedge \mathrm{co}NP \subseteq \exists^1\exists^1\forall^{\log} L \wedge \forall^2 L \subseteq \exists^1\exists^1\forall^{\log}\forall^2 L \subseteq \exists^1\exists^1\forall^2 L$ (Lemma 5.4). Since i can be encoded in a logarithmically length bounded word, then $\Theta_2^p \subseteq \exists^{\log}\exists^1\exists^1\forall^2 L \subseteq \exists^1\exists^1\forall^2 L$ (Lemma 5.4). ❑

CHAPTER 6

Probabilistic Bounded Error Operators

"O amor nasce do conhecimento mútuo e se fortalece na compreensão das diferenças."

G. Marques

In the present chapter, we consider probabilistic bounded error quantifiers. We show under which general conditions the type 2 of a bounded error set quantifier can be reduced. Furthermore, interesting characterizations are presented. For example, we characterize (one prover) interactive proof systems by an existential set quantifier of type 1 and a probabilistic bounded error word quantifier applied to *P*, and show that a bounded error set quantifier of type 1 applied to *PSPACE* can be eliminated without changing the class in question. We also discuss the relativizability of the results.

An outline of this chapter follows: We start defining the probabilistic bounded error quantifiers and giving some more notations (§6.1). Inclusion rules are also shown (§6.2). Then, we examine classes obtained by applying existential, universal or probabilistic bounded error quantifiers to well-known complexity classes (§6.3). Results on interactive proof systems are presented in a separate section (§6.4). Finally, we make some remarks about the results and discuss the relativizability of results presented so far (§6.5).

6.1 The Probabilistic Bounded Error Quantifiers

We start with relating languages with infinite words. Let $\{0,1\}^\omega$ denote the set of infinite binary words. We also write ω-word instead of infinite word. Using lexicographic ordering of $\{0,1\}^*$, we identify as usual languages over $\{0,1\}$ with binary ω-words (see proof of Proposition 3.12).

We will define probabilistic quantifiers varying over infinite objects, namely set of words. The following probability field is used: Define $C_u =_{df} u{\cdot}\{0,1\}^\omega$ as the set of all binary ω-words prefixed by $u \in \{0,1\}^*$, a so called cylinder set. Let $C \cdot U =_{df} \{C_u : u \in U\}$ and $K =_{df} \{C \cdot U : ||U|| < \infty\}$, and let $\sigma(K)$ be the least σ-algebra containing K. Thus, $(\{0,1\}^\omega, \sigma(K), \mu)$ is a probability field, where $\mu : \sigma(K) \to [0,1]$ is the probability measure which is uniquely generated by $\mu(C_u) = 2^{-|u|}$ for all $u \in \{0,1\}^*$. It is well-known that it is equivalent to take the product measure $\mu : 2^{\{0,1\}^\omega} \to [0,1]$ based on the measure $\mu_0 : 2^{\{0,1\}} \to [0,1]$ which is defined by $\mu_0(\{0\}) = \mu_0(\{1\}) = \frac{1}{2}$. For brevity, if Π

is a predicate taking binary ω-words as instance, then $\mu U(\Pi(U))$ is written instead of $\mu(\{U : \Pi(U)\})$.

Following previous quantifier definitions, we define inductively new classes and in parallel the probabilistic bounded error quantifiers. Let $k \geq 1$ and $\sigma_1, \ldots, \sigma_k, \sigma \in \{0, 1, 2\}$. If $\mathcal{K}$ is a class of type $\sigma_1 \ldots \sigma_k\sigma$ then

For $\sigma = 0$: $\mathrm{BP}^p\mathcal{K}$, $\mathrm{R}^p\mathcal{K}$ and $\overline{\mathrm{R}}^p\mathcal{K}$ are classes of type $\sigma_1 \ldots \sigma_k$ which are defined as follows

$L \in \mathrm{BP}^p\mathcal{K} \Longleftrightarrow_{\mathrm{df}}$ there exist an $L' \in \mathcal{K}$ and a polynomial p, such that

$$(X_1, \ldots, X_k) \in L \rightarrow \mathrm{prob}\Big\{z : |z| = p\Big(\sum_{\substack{\sigma_i=0\\ i\leq k}} |X_i|\Big) \wedge (X_1, \ldots, X_k, z) \in L'\Big\} \geq 2/3$$
$$(X_1, \ldots, X_k) \notin L \rightarrow \mathrm{prob}\Big\{z : |z| = p\Big(\sum_{\substack{\sigma_i=0\\ i\leq k}} |X_i|\Big) \wedge (X_1, \ldots, X_k, z) \in L'\Big\} \leq 1/3$$

$L \in \mathrm{R}^p\mathcal{K} \Longleftrightarrow_{\mathrm{df}}$ there exist an $L' \in \mathcal{K}$ and a polynomial p, such that

$$(X_1, \ldots, X_k) \in L \rightarrow \mathrm{prob}\Big\{z : |z| = p\Big(\sum_{\substack{\sigma_i=0\\ i\leq k}} |X_i|\Big) \wedge (X_1, \ldots, X_k, z) \in L'\Big\} \geq 2/3$$
$$(X_1, \ldots, X_k) \notin L \rightarrow \mathrm{prob}\Big\{z : |z| = p\Big(\sum_{\substack{\sigma_i=0\\ i\leq k}} |X_i|\Big) \wedge (X_1, \ldots, X_k, z) \in L'\Big\} = 0$$

$L \in \overline{\mathrm{R}}^p\mathcal{K} \Longleftrightarrow_{\mathrm{df}}$ there exist an $L' \in \mathcal{K}$ and a polynomial p, such that

$$(X_1, \ldots, X_k) \in L \rightarrow \mathrm{prob}\Big\{z : |z| = p\Big(\sum_{\substack{\sigma_i=0\\ i\leq k}} |X_i|\Big) \wedge (X_1, \ldots, X_k, z) \in L'\Big\} = 1$$
$$(X_1, \ldots, X_k) \notin L \rightarrow \mathrm{prob}\Big\{z : |z| = p\Big(\sum_{\substack{\sigma_i=0\\ i\leq k}} |X_i|\Big) \wedge (X_1, \ldots, X_k, z) \in L'\Big\} \leq 1/3$$

where $z \in \{0, 1\}^{p\left(\sum_{i\leq k\ \sigma_i=0}|X_i|\right)}$ is randomly chosen under uniform distribution.

For $\sigma = 1, 2$: $\mathrm{BP}^\sigma\mathcal{K}$, $\mathrm{R}^\sigma\mathcal{K}$ and $\overline{\mathrm{R}}^\sigma\mathcal{K}$ are classes of type $\sigma_1 \ldots \sigma_k$ which are defined as follows

$L \in \mathrm{BP}^\sigma\mathcal{K} \Longleftrightarrow_{\mathrm{df}}$ there exists an $L' \in \mathcal{K}$, such that

$$(X_1, \ldots, X_k) \in L \rightarrow \mu X((X_1, \ldots, X_k, X) \in L') \geq 2/3$$
$$(X_1, \ldots, X_k) \notin L \rightarrow \mu X((X_1, \ldots, X_k, X) \in L') \leq 1/3$$

$L \in R^{\sigma}\mathcal{K} \Longleftrightarrow_{df}$ there exists an $L' \in \mathcal{K}$, such that

$$(X_1, \dots, X_k) \in L \to \mu X((X_1, \dots, X_k, X) \in L') \geq 2/3$$
$$(X_1, \dots, X_k) \notin L \to \mu X((X_1, \dots, X_k, X) \in L') = 0$$

$L \in \overline{R}^{\sigma}\mathcal{K} \Longleftrightarrow_{df}$ there exists an $L' \in \mathcal{K}$, such that

$$(X_1, \dots, X_k) \in L \to \mu X((X_1, \dots, X_k, X) \in L') = 1$$
$$(X_1, \dots, X_k) \notin L \to \mu X((X_1, \dots, X_k, X) \in L') \leq 1/3$$

Following our notation, the set of probabilistic bounded error quantifiers is denoted by $\Gamma_{\%p} =_{df} \{BP^p, BP^1, BP^2, R^p, R^1, R^2, \overline{R}^p, \overline{R}^1, \overline{R}^2\}$. Next, we extend the definition of the quantifier string function to comprise also probabilistic bounded error quantifiers: For $k \geq 0$, $Q_1, \dots, Q_k \in \{BP, R, \overline{R}, \exists, \forall\}$ and $\tau_1, \dots, \tau_k \in \{p, 1, 2\}$, let $\tau(Q_1^{\tau_1} \dots Q_k^{\tau_k}) =_{df} \sigma_1 \dots \sigma_k$ be the type of the operator (or quantifier) string $Q_1^{\tau_1} \dots Q_k^{\tau_k}$, where $\sigma_i = 0$ if $\tau_i = p$ and $\sigma_i = \tau_i$ otherwise ($i = 1, \dots, k$). For $Q = Q_1^{\tau_1} \dots Q_k^{\tau_k}$ and $X = (X_1, \dots, X_k)$ we also write QX instead of $Q_1^{\tau_1}X_1 \dots Q_k^{\tau_k}X_k$. Furthermore, we define $\overline{BP} =_{df} BP$, $\overline{\overline{R}} =_{df} R$ and $\overline{Q} =_{df} \overline{Q}_1^{\tau_1} \dots \overline{Q}_k^{\tau_k}$. The following proposition is evident.

Proposition 6.1. *Let* $\mu \in \{0, 1, 2\}^*$ *and* $Q \in (\Gamma_{\%p} \cup \Gamma_p)^*$. *Then* $\mathrm{co}QP^{\mu\tau(Q)} = \overline{Q}P^{\mu\tau(Q)}$.

Proof. The proof follows as in Proposition 3.1. ❑

6.2 Inclusion Rules

We start observing that the rules "$\to_P$" and "$\leftrightarrow_P$" (§3.2.1) and the "equivalence rule" of Lemma 3.5 remain valid when applied to the classes $BP^{\sigma}QP^{\mu\sigma\tau(S)}$ and $R^{\sigma}QP^{\mu\sigma\tau(S)}$ for $\mu \in \{0, 1, 2\}^*$ and quantifier string $Q \in \Gamma_p^+$. Now, we define inclusion rules to relate classes involving existential, universal or probabilistic bounded error quantifiers applied to P. These rules are applied following our standard sense: For $R, S \in (\Gamma_{\%p} \cup \Gamma_p)^*$, the *inclusion rule* $R \to_{\%P} S$ is valid if the replacement of the quantifier string R by the string S in any context does not diminish the class in question, i.e. $RQP^{\mu\tau(R)\tau(Q)} \subseteq SQP^{\mu\tau(S)\tau(Q)}$ for all $Q \in (\Gamma_{\%p} \cup \Gamma_p)^*$ and $\mu \in \{0, 1, 2\}^*$. Obviously, the complementation observation is also valid for "$\to_{\%P}$" rules.

Proposition 6.2 (Complementation). *Let* $R, S \in (\Gamma_{\%p} \cup \Gamma_p)^*$. *If* $R \to_{\%P} S$ *then* $\overline{R} \to_{\%P} \overline{S}$.

Our first rules show relations between probabilistic bounded error quantifiers of different types.

Lemma 6.3. *The following inclusion rules are valid:*

(1) $\varepsilon \to_{\%P} BP^p$ *and* $\varepsilon \to_{\%P} R^p$;

(2) $BP^p \to_{\%P} BP^1$ *and* $R^p \to_{\%P} R^1$;

(3) $BP^1 \to_{\%P} BP^2$ *and* $R^1 \to_{\%P} R^2$.

Proof. The proof follows as in Lemma 3.4. Let $Q \in (\Gamma_{\%p} \cup \Gamma_p)^*$ and $\mu \in \{0,1,2\}^*$.

(1) This is the classical case of introducing a dummy word quantifier.

(2) We prove the first rule, the other follows in the same way. For a language $L \in \mathrm{BP}^p Q \mathcal{P}^{\mu 0 \tau(Q)}$ there exist an $L_1 \in \mathcal{P}^{\mu 0 \tau(Q)}$ and a polynomial p such that

$$X \in L \to \mathrm{prob}\{z : |z| = p(|X|) \wedge QY\,((X,z,Y) \in L_1)\} \geq 2/3$$
$$X \notin L \to \mathrm{prob}\{z : |z| = p(|X|) \wedge QY\,((X,z,Y) \in L_1)\} \leq 1/3$$

where $z \in \{0,1\}^{p(|X|)}$ is randomly chosen under uniform distribution. Now, define $L_2 =_{\mathrm{df}} \{(X,U,Y) : (X, \langle U, p(|X|)\rangle, Y) \in L_1\}$. Obviously, for each X we have

$$\mathrm{prob}\{z : |z| = p(|X|) \wedge QY\,((X,z,Y) \in L_1)\} = \mu U(QY\,((X,U,Y) \in L_2))$$

Let M be a polynomial-time machine of type $\mu 0 \tau(Q)$ accepting L_1. Consider a machine M' of type $\mu 1 \tau(Q)$ that on input (X,U,Y) computes $p(|X|)$ and then $\langle U, p(|X|)\rangle$ by asking $1, 11, \ldots, 1^{p(|X|)}$ to the oracle U. Then, M' works as M on input $(X, \langle U, p(|X|)\rangle, Y)$. Therefore, $L(M') = L_2$ and $L_2 \in \mathcal{P}^{\mu 1 \tau(Q)}$, i.e. $L \in \mathrm{BP}^1 Q \mathcal{P}^{\mu 1 \tau(Q)}$.

(3) This is obvious since a polynomial-time machine of type $\mu 1 \tau(Q)$ can also be considered to be a machine of type $\mu 2 \tau(Q)$. ❑

Evidently, a bounded error quantifier is at least as powerful as an one-sided bounded error quantifier of same type. This is shown in the following simple observation.

Proposition 6.4. *For* $\sigma \in \{p,1,2\}$ *the inclusion rules* $\mathrm{R}^\sigma \to_{\%P} \mathrm{BP}^\sigma$ *and* $\overline{\mathrm{R}}^\sigma \to_{\%P} \mathrm{BP}^\sigma$ *are valid.*

Proof. Directly from definition of the quantifiers. ❑

The following rules show how to reduce the type 2 of a probabilistic bounded error quantifier.

Lemma 6.5. *The following inclusion rules are valid:*

(1) $\mathrm{BP}^2 \to_{\%P} \mathrm{BP}^1 \exists^1 \forall^p$ *and* $\mathrm{BP}^2 \to_{\%P} \mathrm{BP}^1 \forall^1 \exists^p$;

(2) $\mathrm{R}^2 \to_{\%P} \mathrm{R}^1 \exists^1 \forall^p$ *and* $\overline{\mathrm{R}}^2 \to_{\%P} \overline{\mathrm{R}}^1 \forall^1 \exists^p$;

(3) $\overline{\mathrm{R}}^2 \to_{\%P} \overline{\mathrm{R}}^1 \exists^1 \forall^p$ *and* $\mathrm{R}^2 \to_{\%P} \mathrm{R}^1 \forall^1 \exists^p$.

Proof. The proof follows as in Lemma 4.5. We prove the first rule of every statement, since the other follows by complementation. Let $\mu \in \Phi^*$, $Q \in (\Gamma_{\%p} \cup \Gamma_p)^*$. For a language $L \in \mathrm{BP}^2 Q \mathcal{P}^{\mu 2 \tau(Q)}$ there exists an $L_1 \in \mathcal{P}^{\mu 2 \tau(Q)}$ such that for all X

$$X \in L \to \mu U(QY\,((X,U,Y) \in L_1)) \geq 2/3$$
$$X \notin L \to \mu U(QY\,((X,U,Y) \in L_1)) \leq 1/3$$

However, for each X and a suitable polynomial p

$$\begin{aligned}
&\mu U(QY((X,U,Y) \in L_1)) \\
&\quad = \mu U\Big(\exists^1 W \forall^p u \forall^p v\Big(|u|, |v| \le p(|X|) \to (u \in U \leftrightarrow v01\widehat{u} \in W)\Big) \\
&\qquad\qquad \wedge QY\Big((X,U,Y) \in L(M)\Big)\Big)
\end{aligned}$$

$\big($take $W = \{v01\widehat{u} : u, v \in \{0,1\}^* \wedge u \in U\}$ for example, and let M be a polynomial-time machine of type $\mu 2\tau(Q)$ accepting $L_1\big)$

$$\begin{aligned}
&\quad = \mu U\Big(\exists^1 W \forall^p a \forall^p u \forall^p v\, QY\Big(|u|, |v| \le p(|X|) \to \\
&\qquad\qquad \Big((a = 0 \to (u \in U \leftrightarrow v01\widehat{u} \in W)) \wedge \\
&\qquad\qquad (a = 1 \to (X, U, W, Y) \in L(M'))\Big)\Big)\Big)
\end{aligned}$$

$\big(M'$ works on input (X, U, W, Y) as M on input (X, U, Y) but it only asks the first query to U like M does. Instead of asking u to U after queries $u_1, \ldots, u_m$ ($m \ge 1$) it asks $01\widehat{u}_1 01\widehat{u}_2 01 \ldots 01\widehat{u}_m 01\widehat{u}$ to W. Note that for every (X, a, u, v, Y), the oracles U and W are asked in a type 1 manner.$\big)$

This shows $L \in \mathrm{BP}^1\exists^1\forall^p\forall^p\forall^p Q\mathcal{P}^{\mu 11000\tau(Q)}$. Observe that the inclusion rule $\exists^p\exists^p \to_{\mathcal{P}} \exists^p$ proved in Lemma 3.6 remains also valid in our new context, i.e. we have $\exists^p\exists^p \to_{\%\mathcal{P}} \exists^p$. Hence, $L \in \mathrm{BP}^1\exists^1\forall^p Q\mathcal{P}^{\mu 110\tau(Q)}$.

The proof of $\mathrm{R}^2 \to_{\%\mathcal{P}} \mathrm{R}^1\exists^1\forall^p$ and $\overline{\mathrm{R}}^2 \to_{\%\mathcal{P}} \overline{\mathrm{R}}^1\exists^1\forall^p$ follow in the same way. ❑

6.3 Applications to Well-Known Complexity Classes

We will examine classes of type 0, i.e. "ordinary" classes of languages. If no confusion can arise, in this case the superscripts to the base complexity class $\mathcal{K}$ are omitted, i.e. for quantifier string $Q \in (\Gamma_{\%p} \cup \Gamma_p)^*$ we define $Q\mathcal{K} =_{\mathrm{df}} Q\mathcal{K}^{0\tau(Q)}$. In a fundamental paper, Schöning [Sch89] introduced the word BP-quantifier and gave a more general definition of probabilistic complexity classes. This allows him to generalize many results. Probabilistic quantifiers of type 2 have been studied in [BVW96]. Next, we state previous results on probabilistic bounded error quantifiers and prove other ones. It is well-known that when applied to $\mathcal{P}$ the probabilistic bounded error word quantifiers yield as results the classical probabilistic complexity classes.

Lemma 6.6. [Sch89, BDG90] $\mathrm{BP}^p\mathcal{P} = \mathit{BPP}$ *and* $\mathrm{R}^p\mathcal{P} = \mathit{RP}$.

In 1994, Nisan and Wigderson showed that a probabilistic bounded error set quantifier is exactly as powerful as the corresponding word quantifier when applied to a class of the (arithmetic) polynomial-time hierarchy.

Theorem 6.7. [NW94] *Let* $k \ge 0$. *Then* $\mathrm{BP}^2\Sigma^p_k = \mathrm{BP}^p\Sigma^p_k$ *and* $\mathrm{BP}^2\Pi^p_k = \mathrm{BP}^p\Pi^p_k$.

In 1989, Schöning generalized Sipser's [Sip83] and Lautemann's [Lau83] inclusion $BPP \subseteq \Sigma_2^p \cap \Pi_2^p$ showing under which conditions a BP^p quantifier can be simulated by an $\exists^p$ quantifier followed by an $\forall^p$ quantifier. As the most important special case he proved:

Theorem 6.8. [Sch89] *Let* $k \geq 1$. *Then* $\mathrm{BP}^p\Sigma_k^p \subseteq \Pi_{k+1}^p$ *and* $\mathrm{BP}^p\Pi_k^p \subseteq \Sigma_{k+1}^p$.

Applying standard translational arguments to the previous result, we obtain the following result involving quantifiers of type 2.

Corollary 6.9. *Let* $k, m \geq 1$. *Then,*

$$\mathrm{BP}^2\mathcal{K}_P(2^k, m) \subseteq \Pi_{k+1}^{\mathrm{exp}} \quad \textit{and} \quad \mathrm{BP}^2\mathrm{co}\mathcal{K}_P(2^k, m) \subseteq \Sigma_{k+1}^{\mathrm{exp}}$$

Proof. We prove the first inclusion, the second follows by complementation. The proof of $\mathcal{K}_P(2^k, m) \subseteq \Sigma_k^{\mathrm{exp}}$ (Theorem 3.13) remains valid if the machines have additionally an oracle of type 2 (the exponential-time machine queries this oracle for words of length bounded by p where p is a polynomial) and in [BVW96] it was shown that the quantifier BP^2 can be replaced equivalently by a word BP-quantifier which varies over words of length $2^{q(n)}$ for some polynomial q. Let $\mathrm{BP}^{\mathrm{exp}}$ be this new word quantifier. Hence, $\mathrm{BP}^2\mathcal{K}_P(2^k, m) \subseteq \mathrm{BP}^{\mathrm{exp}}\Sigma_k^{\mathrm{exp}}$. Thus, the assumption follows by applying standard translational arguments to Theorem 6.8. ❑

The next result shows that a type 1 probabilistic bounded error quantifier applied to *PSPACE* can be eliminated without changing the class in question.

Lemma 6.10. BP^1*PSPACE* = *PSPACE.*

Proof. The inclusion *PSPACE* $\subseteq \mathrm{BP}^1$*PSPACE* is evident. For the other inclusion let $L \in \mathrm{BP}^1$*PSPACE.* There exists an $L_1 \in PSPACE^{01}$ such that

$$x \in L \to \mu U((x, U) \in L_1) \geq 2/3$$
$$x \notin L \to \mu U((x, U) \in L_1) \leq 1/3$$

Let M' be a machine of type 01 accepting L_1 with space bound p where p is a polynomial. Without loss of generality we assume that M' does not make a query twice. Since no query of M' on input (x, U) to oracle U is longer than $p(|x|)$ and this oracle is of type 1, then U is queried at most $p(|x|)$ times. It is important that M' asks for a given input (x, U) only queries from one *oracle path* $w_1, w_1w_2, \ldots, w_1w_2\ldots w_{p(|x|)}$. Let M be a machine that considers step by step all these oracle paths. For each oracle path $\pi \in \{0, 1\}^{p(|x|)}$ and each possible answer $\tau \in \{0, 1\}^{p(|x|)}$: M simulates M' on input x querying only queries from the oracle path π and uses the oracle answers encoded in τ. The machine M stops such a simulation if a query is asked which is not from the oracle path π. Finally, the machine M accepts if and only if the number of simulations, which are not stopped, ending in an accepting state is greater than the number of simulations, which are not stopped, ending in an rejecting state. Therefore, $L(M) = L$ and M uses polynomial-space, i.e. $L \in$ *PSPACE.* ❑

In 1996, Book, Vollmer and Wagner investigated the power of the probabilistic set quantifiers of type 2. They showed relationships between *ALMOST* classes and classes defined by BP^2 quantifiers. Let $\mathcal{K}$ be a relativized class. Then

$$L \in \mathit{ALMOST}\text{-}\mathcal{K} \Longleftrightarrow_{\mathrm{df}} \mu U\big(L \in \mathcal{K}^U\big) = 1$$

Theorem 6.11. [BVW96] $\mathit{ALMOST}\text{-}\mathit{PSPACE} = \mathrm{BP}^2\mathit{PSPACE} \subseteq \Sigma_2^{\exp} \cap \Pi_2^{\exp}$.

The following result shows relations between the class $\mathrm{BP}^2\mathit{PSPACE}$ and classes defined by set BP-quantifier on the base of classes of the analytic polynomial-time hierarchy.

Lemma 6.12. $\mathrm{BP}^2\mathit{PSPACE} = \mathrm{BP}^2\exists^1\forall^p P \cap \mathrm{BP}^2\forall^1\exists^p P \subseteq \mathrm{BP}^1\exists^2\forall^p P \cap \mathrm{BP}^1\forall^2\exists^p P$.

Proof. We prove $\mathrm{BP}^2\mathit{PSPACE} = \mathrm{BP}^2\exists^1\forall^p P \subseteq \mathrm{BP}^1\exists^2\forall^p P$, since the other part follows by complementation. The proof of $\mathit{PSPACE} = \exists^1\forall^p P$ (Theorem 3.9) remains valid if the machines have additionally an oracle of type 2, i.e. $\mathrm{BP}^2\mathit{PSPACE} = \mathrm{BP}^2\exists^1\forall^p P$ which is included in $\mathrm{BP}^1\exists^1\forall^p\exists^1\forall^p P$ by Lemma 6.5. Now, using the rules of Lemmas 3.7 and 3.6 we obtain the desired result. ❑

It turns out that type 2 bounded error quantifiers are not more powerful than the corresponding type 1 quantifier when applied to classes of the analytic polynomial-time hierarchy, whose first quantifier is of type 2.

Lemma 6.13. *Let* $\sigma \in \{1,2\}^*$ *and* $m \geq 1$. *Then, for* $\mathcal{K} \in \{\mathcal{K}_P(2\sigma, m)\,, \mathrm{co}\mathcal{K}_P(2\sigma, m)\}$

$$\mathrm{BP}^2\mathcal{K} = \mathrm{BP}^1\mathcal{K} \quad \textit{and} \quad \mathrm{R}^2\mathcal{K} = \mathrm{R}^1\mathcal{K}$$

Proof. We prove $\mathrm{BP}^2\mathcal{K} = \mathrm{BP}^1\mathcal{K}$ for $\mathcal{K} = \mathcal{K}_P(2\sigma, m)$, the other equalities follow in the same way. We conclude

$$\begin{aligned}
\mathrm{BP}^2\mathcal{K}_P(2\sigma, m) &\subseteq \mathrm{BP}^1\exists^1\forall^p\mathcal{K}_P(2\sigma, m) && \text{by Lemma 6.5}\\
&\subseteq \mathrm{BP}^1\mathcal{K}_P(2\sigma, m) && \text{by Lemmas 3.7 and 3.6}\\
&\subseteq \mathrm{BP}^2\mathcal{K}_P(2\sigma, m) && \text{by Lemma 6.3}
\end{aligned}$$
❑

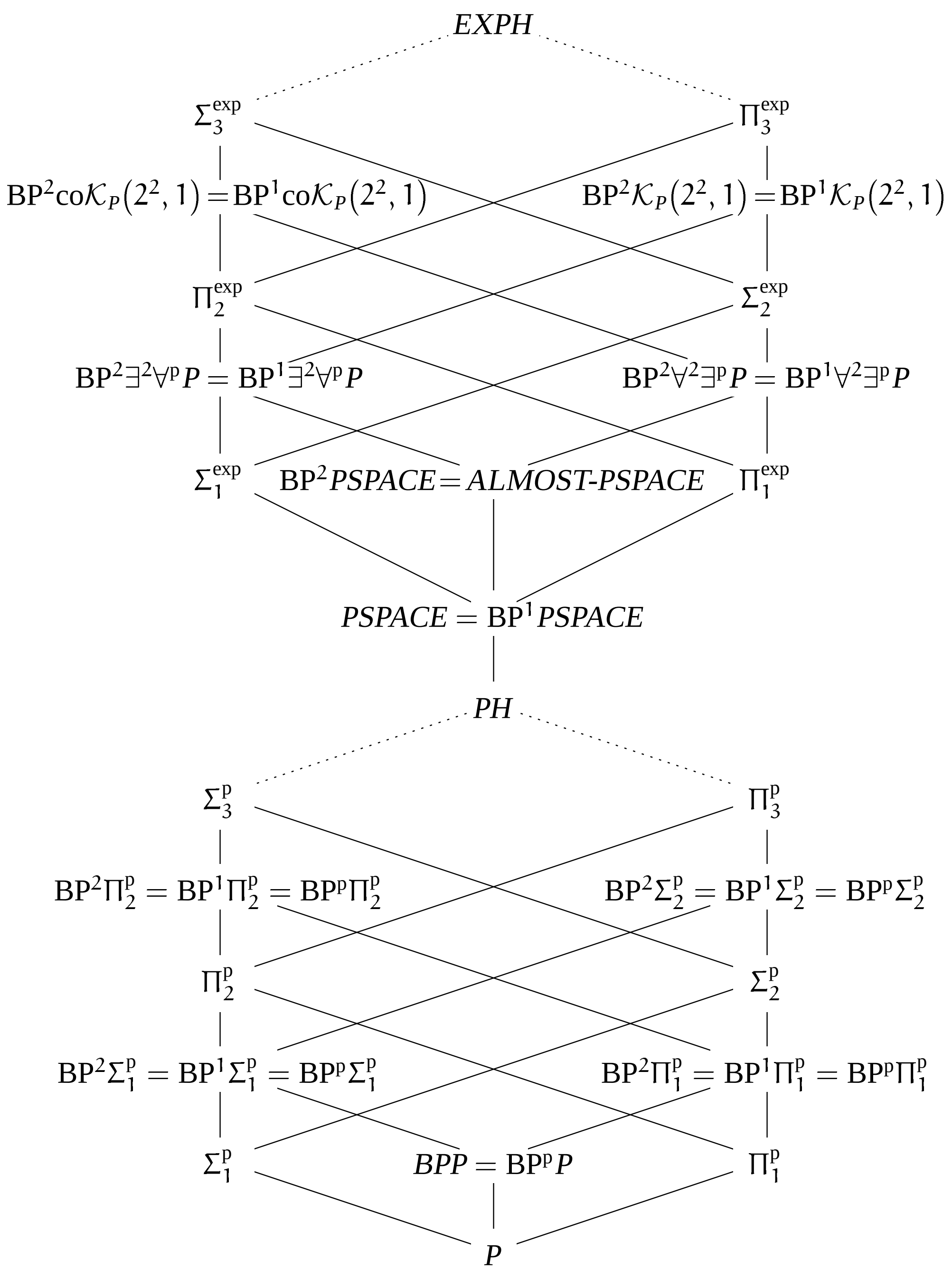
Combining the above results we obtain an inclusion structure which is represented in the Figure 6.1.

6.4 The Emergence of the Type 1 Quantifiers

In 1989, Fortnow, Rompel, and Sipser characterized the power of multi-prover interactive proof systems *MIP* by the class defined by an existential set quantifier (type 2) on the base of the polynomial-time bounded error probability class *BPP*.

Theorem 6.14. [FRS88, BFL90] $\exists^2\mathit{BPP} = \exists^2\mathrm{co}\mathit{RP} = \mathit{MIP}$.

This characterization of *MIP* has motivated us to answer the question of whether a Fortnow-Rompel-Sipser like result could also be established for (one-prover) interactive proof systems *IP*. This is possible by using the quantifier $\exists^1$. So we started the study of the quantifiers of type 1.



$EXPH$
Σ_3^{exp}
Π_3^{exp}
$\text{BP}^2\text{co}\mathcal{K}_P(2^2,1) = \text{BP}^1\text{co}\mathcal{K}_P(2^2,1)$
$\text{BP}^2\mathcal{K}_P(2^2,1) = \text{BP}^1\mathcal{K}_P(2^2,1)$
Π_2^{exp}
Σ_2^{exp}
$\text{BP}^2\exists^2\forall^{\text{p}}P = \text{BP}^1\exists^2\forall^{\text{p}}P$
$\text{BP}^2\forall^2\exists^{\text{p}}P = \text{BP}^1\forall^2\exists^{\text{p}}P$
Σ_1^{exp}
$\text{BP}^2 PSPACE = ALMOST\text{-}PSPACE$
Π_1^{exp}
$PSPACE = \text{BP}^1 PSPACE$
PH
Σ_3^{p}
Π_3^{p}
$\text{BP}^2\Pi_2^{\text{p}} = \text{BP}^1\Pi_2^{\text{p}} = \text{BP}^{\text{p}}\Pi_2^{\text{p}}$
$\text{BP}^2\Sigma_2^{\text{p}} = \text{BP}^1\Sigma_2^{\text{p}} = \text{BP}^{\text{p}}\Sigma_2^{\text{p}}$
Π_2^{p}
Σ_2^{p}
$\text{BP}^2\Sigma_1^{\text{p}} = \text{BP}^1\Sigma_1^{\text{p}} = \text{BP}^{\text{p}}\Sigma_1^{\text{p}}$
$\text{BP}^2\Pi_1^{\text{p}} = \text{BP}^1\Pi_1^{\text{p}} = \text{BP}^{\text{p}}\Pi_1^{\text{p}}$
Σ_1^{p}
$BPP = \text{BP}^{\text{p}}P$
Π_1^{p}
P

Figure 6.1: *Inclusion structure of classes involving* BP*-quantifiers.*

Theorem 6.15. $\exists^1 BPP = \exists^1 \mathrm{co}RP = IP$.

Proof. By definition, $L \in IP$ if and only if there exists a polynomial-time probabilistic verifier V such that

$$\begin{aligned} x \in L &\to \exists \text{ Prover } P\ (\text{prob}\ (V \text{ accepts } x \text{ with } P) \geq 2/3) \\ x \notin L &\to \forall \text{ Prover } P\ (\text{prob}\ (V \text{ accepts } x \text{ with } P) \leq 1/3) \end{aligned} \tag{6.1}$$

Here a *prover* is a function $P : \Sigma^* \times (\Sigma^*)^\omega \to \Sigma^*$ which determines for a given input x and a sequence of queries $u_1, \ldots, u_k$ of the verifier the answer $P(x, u_1, \ldots, u_k)$ to the query u_k. Now, in the same way define the class IP' with the difference that for words in the language the verifier is convinced with probability 1, i.e. $L \in IP'$ if and only if there exists a polynomial-time probabilistic verifier V such that

$$\begin{aligned} x \in L &\to \exists \text{ Prover } P\ (\text{prob}\ (V \text{ accepts } x \text{ with } P) = 1) \\ x \notin L &\to \forall \text{ Prover } P\ (\text{prob}\ (V \text{ accepts } x \text{ with } P) \leq 1/3) \end{aligned} \tag{6.2}$$

Without loss of the generality let V ask the prover only for one-bit answer.

The theorem's assumption is proved showing the following chain of inclusions:

$$\exists^1 BPP \subseteq IP \subseteq PSPACE \subseteq IP' \subseteq \exists^1 \mathrm{co}RP \subseteq \exists^1 BPP$$

The inclusions $IP \subseteq PSPACE \subseteq IP'$ have been shown in [Sha90] and the inclusion $\exists^1 \mathrm{co}RP \subseteq \exists^1 BPP$ follows by Proposition 6.4. Thus, it remains to prove $\exists^1 BPP \subseteq IP$ and $IP' \subseteq \exists^1 \mathrm{co}RP$.

"$\exists^1 BPP \subseteq IP$": Let $L \in \exists^1 BPP$. There exist an $L' \in P^{010}$ and a polynomial p such that

$$\begin{aligned} x \in L &\to \exists^1 U\ (\text{prob}\{z : |z| = p(|x|) \wedge (x, U, z) \in L'\} \geq 2/3) \\ x \notin L &\to \forall^1 U\ (\text{prob}\{z : |z| = p(|x|) \wedge (x, U, z) \in L'\} \leq 1/3) \end{aligned}$$

where $z \in \{0,1\}^{p(|x|)}$ is randomly chosen under uniform distribution. Let M be a polynomial-time machine of type 010 accepting L', and let V be a verifier that on input x and path z works as M on input (x, U, z) with the difference that if the machine M asks the query wu to U and the previous query was w, then the verifier V asks the queries $u(1), u(2), \ldots, u(|u|)$ to the prover, ignores the answers to $u(1), u(2), \ldots, u(|u| - 1)$, and uses the answer of $u(|u|)$ in the same way that M uses the answer of $wu \in U$. Hence, V is a probabilistic polynomial-time machine and for each x:

1. Let $U \subseteq \{0,1\}^*$ and define $P_U(x, a_1, \ldots, a_k) =_{df} \chi_U(a_1 \ldots a_k)$ for all $k \in \mathbb{N}$ and all $a_1, \ldots, a_k \in \{0,1\}$. Hence, V accepts x with P_U on path z if and only if M accepts the input (x, U, z). Therefore,

   $$\text{prob}\ (V \text{ accepts } x \text{ with } P_U) = \text{prob}\{z : |z| = p(|x|) \wedge (x, U, z) \in L'\}$$

2. Let P be a prover and define

   $$U_P =_{df} \{a_1 \ldots a_k : k \in \mathbb{N} \wedge a_1, \ldots, a_k \in \{0,1\} \wedge P(x, a_1, \ldots, a_k) = 1\}$$

   Hence, M accepts the input (x, U_P, z) if and only if V accepts x with P on path z. Therefore, $\text{prob}\{z : |z| = p(|x|) \wedge (x, U_P, z) \in L'\} = \text{prob}\ (V \text{ accepts } x \text{ with } P)$.

Thus, for every x

$$\max_P \operatorname{prob}(V \text{ accepts } x \text{ with } P) = \max_U \operatorname{prob}\{z : |z| = p(|x|) \wedge (x, U, z) \in L'\}$$

Therefore, V satisfies (6.1) and consequently $L \in$ *IP*.

"$IP' \subseteq \exists^1\mathrm{co}RP$": Let $L \in IP'$ and let V be a verifier satisfying (6.2). Consider a machine M of type 010 working on input (x, U, z) as the verifier V on input x and path z with the difference that if the verifier has already asked the queries $u_1, \ldots, u_{k-1}$ and asks then the query u_k to the prover, then M asks the query $\widehat{u}_1 01 \widehat{u}_2 01 \ldots 01 \widehat{u}_k$ to the oracle U and proceeds with the answer of the oracle in the same way as V with the answer of the prover. Hence, $L(M) \in P^{010}$, and for a suitable polynomial p and each x:

1. Let P be a prover and define $U_P =_{\mathrm{df}} \{\widehat{u}_1 01 \ldots 01 \widehat{u}_k : P(x, u_1, \ldots, u_k) = 1\}$. Thus, M accepts the input (x, U_P, z) if and only if V accepts x with P on path z. Therefore, $\operatorname{prob}\{z : |z| = p(|x|) \wedge (x, U_P, z) \in L(M)\} = \operatorname{prob}(V \text{ accepts } x \text{ with } P)$.

2. Let $U \subseteq \{0,1\}^*$ and define $P_U(x, u_1, \ldots, u_k) =_{\mathrm{df}} \chi_U(\widehat{u}_1 01 \ldots 01 \widehat{u}_k)$. Hence, V accepts x with P_U on path z if and only if M accepts the input (x, U, z). Therefore, $\operatorname{prob}(V \text{ accepts } x \text{ with } P_U) = \operatorname{prob}\{z : |z| = p(|x|) \wedge (x, U, z) \in L(M)\}$.

Thus, for every x

$$\max_U \operatorname{prob}\{z : |z| = p(|x|) \wedge (x, U, z) \in L(M)\} = \max_P \operatorname{prob}(V \text{ accepts } x \text{ with } P)$$

Hence,

$$\begin{aligned} x \in L &\rightarrow \exists^1 U\,(\operatorname{prob}\{z : |z| = p(|x|) \wedge (x, U, z) \in L(M)\} = 1) \\ x \notin L &\rightarrow \forall^1 U\,(\operatorname{prob}\{z : |z| = p(|x|) \wedge (x, U, z) \in L(M)\} \leq 1/3) \end{aligned}$$

Therefore, $L \in \exists^1\mathrm{co}RP$. ❏

6.5 Conclusions

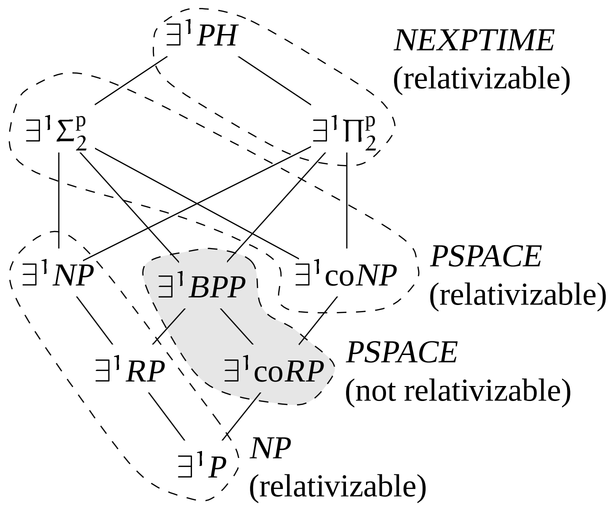
In §6.3 we examined classes obtained by applying BP-quantifiers to well-known complexity classes and classes of the analytic polynomial-time hierarchy. Figure 6.1 summarizes these results. In §6.4 we presented the results which motivated us to study type 1 quantifiers. A further interesting investigation on this theme could be a detailed study of the $\exists$-$\forall$-BP-hierarchy over *P* using word quantifiers as well as set quantifiers of type 1 and 2.

Finally, let us note that for the classes co*RP*, *BPP* and co*NP* we get the same result when applying the quantifiers $\exists^1$ and $\exists^2$:

$\exists^1\mathrm{co}RP = \exists^1 BPP = \exists^1\mathrm{co}NP = PSPACE$ by Theorems 6.15 and 3.9, and [Sha90]

$\exists^2\mathrm{co}RP = \exists^2 BPP = \exists^2\mathrm{co}NP = NEXPTIME$ by Theorems 6.14 and 3.14, and [BFL90]

However, with respect to relativizability these results are of completely different quality: Whereas the results $\exists^1\mathrm{co}RP = PSPACE$, $\exists^1 BPP = PSPACE$, $\exists^2\mathrm{co}RP = NEXPTIME$, and $\exists^2 BPP = NEXPTIME$ are not valid in every relativized world [FS88, FRS88] (§2.3.1) the results $\exists^1\text{co}NP = PSPACE$ and $\exists^2\text{co}NP = NEXPTIME$ are valid in every relativized world. The Figure 6.2 gives an overview on the results on classes $\exists^1\mathcal{K}$ and $\exists^2\mathcal{K}$ where $\mathcal{K}$ are interesting classes within the polynomial-time hierarchy.



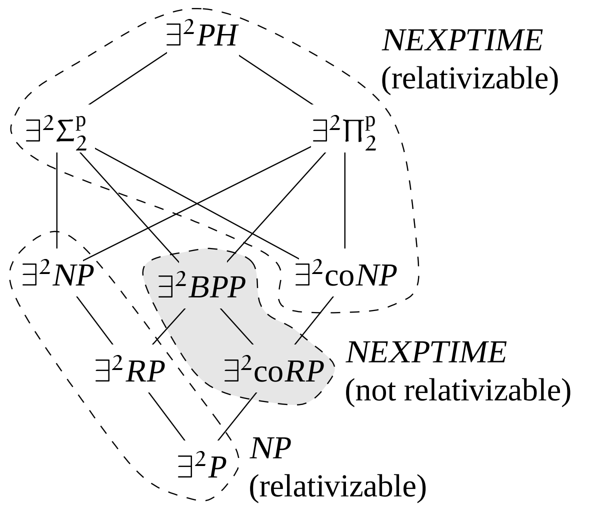


Figure 6.2: *Relativized world of classes $\exists^1\mathcal{K}$ and $\exists^2\mathcal{K}$.*

Bibliography

[All97] E. Allender. $\exists^1\forall^{\log}L = \forall^1\exists^{\log}L = L$. Personal communication, 1997.

[ALM$^+$92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *33rd Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs; a new characterization of NP. In *33rd Symposium on Foundations of Computer Science*, pages 2–13, 1992.

[Bai97] H. Baier. Operatoren höherer Ordnung in der Komplexitätstheorie. In H. Vollmer, editor, *Komplexitätstheorie: Maschinen und Operatoren*, pages 85–95. Cuvillier Verlag, Göttingen, 1997.

[Bar86] D. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *18th ACM Symposium on the Theory of Computing*, pages 1–5, 1986.

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

[BDG95] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. Springer-Verlag, second edition, 1995.

[BFL90] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. In *31st Symposium on Foundations of Computer Science*, pages 16–25, 1990.

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

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

[BLW94] R. V. Book, J. H. Lutz, and K. W. Wagner. An observation on probability versus randomness with applications to complexity classes. *Mathematical Systems Theory*, 27:201–209, 1994.

[BOGKW88] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *20th ACM Symposium on the Theory of Computing*, pages 113–131, 1988.

[BVW96] R. V. Book, H. Vollmer, and K. W. Wagner. On type-2 probabilistic quantifiers. In *23rd International Colloqium on Automata, Languages and Programming*, volume 1099 of *Lecture Notes in Computer Science*, pages 369–380. Springer Verlag, 1996.

[BW96] H. Baier and K. W. Wagner. The analytic polynomial-time hierarchy. Technical Report 148, Institut für Informatik, Universität Würzburg, Germany, September 1996. To appear in Mathematical Logic Quarterly (formerly: Zeitschrift für Mathematische Logik und Grundlagen der Mathematik).

[BW97] H. Baier and K. W. Wagner. Bounding queries in the analytic polynomial-time hierarchy. Technical Report 178, Institut für Informatik, Universität Würzburg, Germany, August 1997. To appear in Theoretical Computer Science.

[CF91] J. Cai and M. Furst. PSPACE survives constant-width bottlenecks. *International Journal of Foundations of Computer Sciences*, 2(1):67–76, 1991.

[CKS81] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, January 1981.

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

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

[FS88] L. Fortnow and M. Sipser. Are there interactive protocols for co-NP languages? *Information Processing Letters*, 28:249–251, 1988.

[Gil77] J. Gill. Computational complexity of probabilistic complexity classes. *SIAM Journal on Computing*, 6:675–695, 1977.

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

[GLM96] J. Goldsmith, M. A. Levy, and M. Mundhenk. Limited nondeterminism. *SIGACT (ACM press)*, 27(2):20–29, 1996.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *17th ACM Symposium on the Theory of Computing*, pages 291–304, 1985.

[HLS$^+$93] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, and K. W. Wagner. On the power of polynomial time bit-reductions. In *8th Structure in Complexity Theory Conference*, pages 200–207, 1993.

[HO94] L. A. Hemaspaandra and M. Ogihara. Universally serializable computation. Technical Report 520, University of Rochester – Computer Science, 1994.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company Inc, 1979.

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

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

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

[LP82] H. R. Lewis and C. H. Papadimitriou. Symmetric space-bounded computation. *Theoretical Computer Science*, 19:161–187, 1982.

[MS72] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *13rd IEEE Symposium on Switching and Automata Theory*, pages 125–129, 1972.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[Ogi94] M. Ogihara. On serializable languages. *International Journal of Foundations of Computer Sciences*, 5:303–318, 1994.

[Orp83] P. Orponen. Complexity classes of alternating machines with oracles. In *10th International Colloqium on Automata, Languages and Programming*, volume 154 of *Lecture Notes in Computer Science*, pages 573–584. Springer Verlag, 1983.

[Pap85] C. H. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31:288–301, 1985.

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company Inc, 1994.

[Sav70] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.

[Sch85] U. Schöning. Robust algorithms: A different approach to oracles. *Theoretical Computer Science*, 40:57–66, 1985.

[Sch89] U. Schöning. Probabilistic complexity classes and lowness. *Journal of Computer and System Sciences*, 39:84–100, 1989.

[Sha90] A. Shamir. IP=PSPACE. In *31st Symposium on Foundations of Computer Science*, pages 11–15, 1990.

[Sim77] J. Simon. On the difference between one and many. In *14th International Colloqium on Automata, Languages and Programming*, volume 52 of *Lecture Notes in Computer Science*, pages 480–491. Springer Verlag, 1977.

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

[SM73] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In *5th ACM Symposium on the Theory of Computing*, pages 1–9, 1973.

[Sto77] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[VW97] H. Vollmer and K. W. Wagner. On operators of higher types. In *12th Annual IEEE Conference on Computational Complexity*, pages 174–184, 1997.

[Wag86] K. W. Wagner. Some observations on the connection between counting and recursion. *Theoretical Computer Science*, 47:131–147, 1986.

[Wag90] K. W. Wagner. Bounded query classes. *SIAM Journal on Computing*, 19:833–846, 1990.

[Wra77] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3:23–33, 1977.

[WW86] K. W. Wagner and G. Wechsung. *Computational Complexity*. VEB Deutscher Verlag der Wissenschaften, 1986.

Index