The Analytic Polynomial-Time Hierarchy

Herbert Baier* Klaus W. Wagner
Lehrstuhl für Theoretische Informatik
Universität Würzburg, Germany
[baier,wagner]@informatik.uni-wuerzburg.de

September 1995

Abstract

We investigate an $\exists$-$\forall$-hierarchy over *P* using word quantifiers as well as two types of set quantifiers. This hierarchy, which extends the (arithmetic) polynomial-time hierarchy, is called the *analytic polynomial-time hierarchy*. It is shown that every class of this hierarchy coincides with one of the following classes: Σ_k^{p}, Π_k^{p} ($k \geq 0$), *PSPACE*, Σ_k^{exp} or Π_k^{exp} ($k \geq 1$). This improves previous results by Orponen [7] and gives rise to some comparisons with results on interactive proof systems.

1 Introduction

Complexity classes defined (or characterized) by quantifiers on the base of the class *P* are among the most intensively investigated subjects in complexity theory. Here the classes of the polynomial-time hierarchy [10, 9, 11], which are characterized by the existential and the universal quantifier, play an important role. These quantifiers vary over words whose lengths are polynomially bounded in the length of the input.

Starting in the eighties complexity classes were studied which are defined by quantifiers varying over sets of words. Using the Turing machine model this means that the quantifiers vary over all possible oracles. In 1983, Orponen [7] studied the hierarchy defined by existential and universal set quantifiers on the base of the class *PH*. He characterized this hierarchy with the classes of the exponential-time alternation hierarchy. In 1988, Fortnow, Rompel, and Sipser [5] characterized the power of multi-prover interactive proof systems by the class defined by an existential set quantifier on the base of the polynomial-time bounded error probability class *BPP*. In 1996, Book, Vollmer, and Wagner [3] investigated the power of probabilistic set quantifiers.

Our investigations were initialized by the question of whether a Fortnow-Rompel-Sipser like result could also be established for (one-prover) interactive proof systems. This turned out to be possible by restricting the access to the oracles in the following way: every query is an extension of the previous query. That means: the series of queries in any computation has the form

*Supported by **CNPq** grant 290020/92.1 (Brazil) and in part by **DAAD** (Germany).

$u_1, u_1u_2, u_1u_2u_3, \ldots$. We call quantifiers varying over oracles with this kind of restricted access quantifiers of type 1 whereas quantifiers varying over oracles with unrestricted access are called quantifiers of type 2. Word quantifiers are called quantifiers of type 0. Now we continue Orponen's investigations and study the hierarchy over *P* defined by existential and universal quantifiers of type 0, 1, and 2 which is called the *analytic polynomial-time hierarchy*. We prove that every class of this hierarchy coincides with one of the classes Σ_k^{p} and Π_k^{p} ($k \geq 0$) of the 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. An algorithm is established which allows to find out this class in an easy way.

These results improve Orponen's result on quantifiers of type 2, and we show relations like $\exists^1\mathrm{co}NP = \mathit{PSPACE}$ and $\exists^1\exists^1\mathrm{co}NP = \exists^2\mathrm{co}NP = \mathit{NEXPTIME}$ (generally, two quantifiers of type 1 are as powerful as one quantifier of type 2). Comparing these results with the oracle characterization of interactive proof systems we get $\mathit{IP} = \exists^1 \mathit{BPP} = \exists^1\mathrm{co}NP = \mathit{PSPACE}$ and $\mathit{MIP} = \exists^2 \mathit{BPP} = \exists^2\mathrm{co}NP = \mathit{NEXPTIME}$. However, the *BPP* part of these results is not relativizable whereas the *NP* part is valid under every relativization.

2 Preliminaries

We will study a polynomial-time hierarchy built up by word and set operators (defined by word and set quantifiers, respectively). Thus, we have to start with a suitable class of polynomial time languages. Since the instances of these languages consist of words and sets of words, we will use polynomial-time oracle Turing machines 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* (*2*) if the query on the corresponding query tape is not erased (erased, respectively) after every 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*.

For $k \geq 1$ and $\sigma_1, \sigma_2, \ldots, \sigma_k \in \{0, 1, 2\}$, we say that a Turing machine is of *type* $\sigma_1\sigma_2\ldots\sigma_k$ if it processes instances of the form $(X_1, \ldots, X_k)$, where X_i is an input of type σ_i ($i = 1, \ldots, k$). Such a machine has $\#\{i : \sigma_i = 0\}$ ordinary input tapes and $\#\{i : \sigma_i \in \{1, 2\}\}$ query tapes. For a Turing machine M of type $\sigma_1\sigma_2\ldots\sigma_k$ define

$$L(M) =_{\mathrm{df}} \{(X_1, \ldots, X_k) : X_i \text{ is an input of type } \sigma_i \text{ and } M \text{ halts on instance } (X_1, \ldots, X_k) \text{ with an accepting state}\}$$

as the *language accepted by* M.

A Turing machine M of type $\sigma_1\sigma_2\ldots\sigma_k$ is called *polynomial-time* (*polynomial-space*, respectively) if there exists a polynomial p such that M halts on every instance $X = (X_1, \ldots, X_k)$ in an accepting or rejecting state using no more than $p(|X|) =_{\mathrm{df}} p\left(\sum_{\sigma_i=0} |X_i|\right)$ steps (tape cells, respectively). In what follows we assume that X contains at least one word input. Note that the space bound applies also to the length of oracle queries. Now, for $k \geq 1$ and $\sigma_1, \sigma_2, \ldots, \sigma_k \in \{0, 1, 2\}$ define

$$\begin{aligned} P^{\sigma_1\sigma_2\ldots\sigma_k} &=_{\mathrm{df}} \{L(M) : M \text{ is a polynomial-time Turing machine of type } \sigma_1\sigma_2\ldots\sigma_k\} \\ \mathit{PSPACE}^{\sigma_1\sigma_2\ldots\sigma_k} &=_{\mathrm{df}} \{L(M) : M \text{ is a polynomial-space Turing machine of type } \sigma_1\sigma_2\ldots\sigma_k\} \end{aligned}$$

which are called *classes of type* $\sigma_1\sigma_2\ldots\sigma_k$. The classes $\boldsymbol{P}^{\sigma_1\sigma_2\ldots\sigma_k}$ are the starting point for building up the polynomial-time hierarchies. Next, we define inductively new classes: If $\mathcal{K}$ is a class of type $\sigma_1\sigma_2\ldots\sigma_k\sigma$ then $\exists^\sigma\mathcal{K}$ and $\forall^\sigma\mathcal{K}$ are classes of type $\sigma_1\sigma_2\ldots\sigma_k$ which are defined as follows.
For $\sigma = 0$:

$L \in \exists^0\mathcal{K} \Longleftrightarrow_{\mathrm{df}}$ there exists an $L' \in \mathcal{K}$ and a polynomial p, such that

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

$L \in \forall^0\mathcal{K} \Longleftrightarrow_{\mathrm{df}}$ there exists an $L' \in \mathcal{K}$ and a polynomial p, such that

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

(Using simple encoding arguments it is easy to see that one can use equivalently "$=$" instead of "$\leq$" in these definitions.)
For $\sigma = 1, 2$:

$L \in \exists^\sigma\mathcal{K} \Longleftrightarrow_{\mathrm{df}}$ 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

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

Obviously, a class of type $\sigma_1\sigma_2\ldots\sigma_k$ consists only of languages $L \subseteq \Sigma_1^{(\sigma_1)} \times \cdots \times \Sigma_k^{(\sigma_k)}$ for suitable finite alphabets $\Sigma_1,\ldots,\Sigma_k$, where we define $\Sigma^{(0)} =_{\mathrm{df}} \Sigma^*$ and $\Sigma^{(\sigma)} =_{\mathrm{df}} \mathcal{P}(\Sigma^*)$ for $\sigma = 1, 2$ (without loss of generality we restrict ourselves to the alphabet $\Sigma = \{0,1\}$). We are particularly interested in the classes of type 0, i.e. in "ordinary" classes of languages. In this case, the superscripts to $\boldsymbol{P}$ are omitted, i.e. $Q_1^{\sigma_1}Q_2^{\sigma_2}\ldots Q_k^{\sigma_k}\boldsymbol{P} =_{\mathrm{df}} Q_1^{\sigma_1}Q_2^{\sigma_2}\ldots Q_k^{\sigma_k}\boldsymbol{P}^{0\sigma_1\sigma_2\ldots\sigma_k}$ ($Q_i \in \{\exists,\forall\}$).

Define the classes of the *(arithmetic) polynomial-time hierarchy* as $\Sigma_0^{\mathrm{p}} =_{\mathrm{df}} \Pi_0^{\mathrm{p}} =_{\mathrm{df}} \boldsymbol{P}$ and, for $k \geq 0$, $\Sigma_{k+1}^{\mathrm{p}} =_{\mathrm{df}} \exists^0\Pi_k^{\mathrm{p}}$ and $\Pi_{k+1}^{\mathrm{p}} =_{\mathrm{df}} \forall^0\Sigma_k^{\mathrm{p}}$. Furthermore define *PH* as the union of all classes of the polynomial-time hierarchy. It is well-known that each of the classes $Q_1^0Q_2^0\ldots Q_k^0\boldsymbol{P}$ coincides with a class of the polynomial-time hierarchy. For $k \geq 0$ and $\sigma_1,\sigma_2,\ldots,\sigma_k \in \{0,1,2\}$, the classes $Q_1^{\sigma_1}Q_2^{\sigma_2}\ldots Q_k^{\sigma_k}\boldsymbol{P}$ 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.

For $k \geq 0$, $Q_1,Q_2,\ldots,Q_k \in \{\exists,\forall\}$ and $\sigma_1,\sigma_2,\ldots,\sigma_k \in \{0,1,2\}$, let $\tau\left(Q_1^{\sigma_1}Q_2^{\sigma_2}\ldots Q_k^{\sigma_k}\right) =_{\mathrm{df}} \sigma_1\sigma_2\ldots\sigma_k$ be the type of the operator (or quantifier) string $Q_1^{\sigma_1}Q_2^{\sigma_2}\ldots Q_k^{\sigma_k}$. Furthermore, for $Q = Q_1^{\sigma_1}Q_2^{\sigma_2}\ldots Q_k^{\sigma_k}$ and $X = (X_1,\ldots,X_k)$ we write QX instead of $Q_1^{\sigma_1}X_1Q_2^{\sigma_2}X_2\ldots Q_k^{\sigma_k}X_k$. We define $\overline{\exists} =_{\mathrm{df}} \forall$ and $\overline{\forall} =_{\mathrm{df}} \exists$, and for $Q = Q_1^{\sigma_1}Q_2^{\sigma_2}\ldots Q_k^{\sigma_k}$ we define $\overline{Q} =_{\mathrm{df}} \overline{Q}_1^{\sigma_1}\overline{Q}_2^{\sigma_2}\ldots\overline{Q}_k^{\sigma_k}$. For a class $\mathcal{K}$ of languages, we define $\mathrm{co}\mathcal{K} =_{\mathrm{df}} \left\{\overline{L} : L \in \mathcal{K}\right\}$. Obviously, we have $\mathrm{co}Q\boldsymbol{P}^{\mu\tau(Q)} = \overline{Q}\boldsymbol{P}^{\mu\tau(Q)}$ for all $\mu \in \{0,1,2\}^*$. Define $\Gamma =_{\mathrm{df}} \{\exists^0,\exists^1,\exists^2,\forall^0,\forall^1,\forall^2\}$.

Finally, some more notations are given to help us in the proofs of the results: For $a \in \{0,1\}$ and $u \in \{0,1\}^*$, define the encodings $\widetilde{ua} =_{\mathrm{df}} \tilde{u}1a$ ($\tilde{\epsilon} =_{\mathrm{df}} \epsilon$) and $\widehat{ua} =_{\mathrm{df}} \hat{u}aa$ ($\hat{\epsilon} =_{\mathrm{df}} \epsilon$). For $u \in \{0,1\}^*$ and $U \subseteq \{0,1\}^*$, define $u\backslash U =_{\mathrm{df}} \{w : uw \in U\}$. For a set $U \subseteq \{0,1\}^*$, let c_U be the characteristic function of U. For every set $U \subseteq \{0,1\}^*$ and every $m \in \mathbb{N}$ we define the word $\langle U, m\rangle =_{\mathrm{df}} c_U(1)c_U(1^2)c_U(1^3)\dots c_U(1^m)$. For $u \in \{0,1\}^*$ we denote the i-th bit of u by $u(i)$.

3 Previous Results on Set Operators

To our knowledge, Orponen was the first who studied in 1983 operators of type 2. He related the $\exists^2$-$\forall^2$-hierarchy to the exponential-time hierarchy. For $k \geq 1$, let $\Sigma_k^{\exp}$ be the class of all languages which can be accepted by an exponential-time (i.e. $2^{p(|\cdot|)}$ time-bounded for some polynomial p) alternating Turing machine starting with an existential state and having at most $k-1$ alternations on every computation path. Obviously, $\Sigma_1^{\exp} = \mathit{NEXPTIME}$ (i.e. the class of languages which can be accepted by nondeterministic exponential-time Turing machines).

Theorem 1. [7] *For every $k \geq 1$ let $Q_k = \exists$ if k is odd and $Q_k = \forall$ otherwise. Then,*

$$\exists^2\forall^2\exists^2 \dots Q_k^2 \mathit{PH} = \Sigma_k^{\exp}. \qquad ❑$$

In 1989 Fortnow, Rompel, and Sipser characterized the power of multi-prover interactive proof systems by type 2 operators. Let *BPP* (*RP*) be the class of all languages which can be accepted by polynomial-time bounded error (one-sided bounded error, respectively) probabilistic Turing machines, and let *IP* (*MIP*) be the class of languages which can be accepted by interactive proof systems (multi-prover interactive proof systems, respectively) in polynomial time. It has been shown [8, 1] that *IP* $=$ *PSPACE* and *MIP* $=$ *NEXPTIME*.

Theorem 2. [5, 1] $\exists^2\mathit{BPP} = \exists^2\mathit{RP} = \mathit{MIP}$. ❑

Now we were interested in the question of whether one could give such an characterization also for the class *IP*. This is possible by using the operator $\exists^1$.

Theorem 3. $\exists^1\mathit{BPP} = \exists^1\mathit{RP} = \mathit{IP}$.

Proof. By definition, $L \in \mathit{IP}$ if and only if there exists a polynomial-time probabilistic verifier V such that

$$\begin{aligned} x \in L &\rightarrow \exists \text{ Prover } P\,(\text{prob}\,(V \text{ accepts } x \text{ with } P) \geq 2/3)\\ x \notin L &\rightarrow \forall \text{ Prover } P\,(\text{prob}\,(V \text{ accepts } x \text{ with } P) \leq 1/3)\end{aligned} \tag{1}$$

Here a prover is a function $P : \Sigma^* \times (\Sigma^*)^\omega \rightarrow \Sigma^*$ which determines for a given input x and a sequence of queries $u_1, \dots, u_k$ of the verifier the answer $P(x, u_1, \dots, u_k)$ 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. $L \in \mathit{IP}'$ if and only if there exists a polynomial-time probabilistic verifier V such that

$$\begin{aligned} x \in L &\rightarrow \exists \text{ Prover } P\,(\text{prob}\,(V \text{ accepts } x \text{ with } P) = 1)\\ x \notin L &\rightarrow \forall \text{ Prover } P\,(\text{prob}\,(V \text{ accepts } x \text{ with } P) \leq 1/3)\end{aligned} \tag{2}$$

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:

$$\exists^1\mathit{BPP} \subseteq \mathit{IP} \subseteq \mathit{PSPACE} \subseteq \mathit{IP}' \subseteq \exists^1\mathit{RP} \subseteq \exists^1\mathit{BPP}$$

The inclusion *PSPACE* $\subseteq$ *IP'* has been shown in [8]. Thus, we prove only $\exists^1$*BPP* $\subseteq$ *IP* and *IP'* $\subseteq \exists^1$*RP*, since the inclusions *IP* $\subseteq$ *PSPACE* and $\exists^1$*RP* $\subseteq \exists^1$*BPP* are evident.
"$\exists^1$*BPP* $\subseteq$ *IP*": Let $L \in \exists^1$*BPP*. There exists an $L' \in$ *P*^{010} and a polynomial p such that

$$x \in L \rightarrow \exists^1 U \left(\text{prob}\left\{z : |z| = p\left(|x|\right) \wedge (x, U, z) \in L'\right\} \geq 2/3\right)$$
$$x \notin L \rightarrow \forall^1 U \left(\text{prob}\left\{z : |z| = p\left(|x|\right) \wedge (x, U, z) \in L'\right\} \leq 1/3\right)$$

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 $P_U(x, a_1, \ldots, a_k) =_{\text{df}} c_U(a_1 \ldots a_k)$ 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,

$$\text{prob}\left(V \text{ accepts } x \text{ with } P_U\right) = \text{prob}\left\{z : |z| = p\left(|x|\right) \wedge (x, U, z) \in L'\right\}$$

2. Let P be a prover and define

$$U_P =_{\text{df}} \{a_1 \ldots a_k : k \in \mathbb{N} \wedge a_1, \ldots, a_k \in \{0,1\} \wedge P(x, a_1, \ldots, a_k) = 1\}$$

Hence, M accepts the input (x, U_P, z) if and only if V accepts x with P on path z. Therefore,

$$\text{prob}\left\{z : |z| = p\left(|x|\right) \wedge (x, U_P, z) \in L'\right\} = \text{prob}\left(V \text{ accepts } x \text{ with } P\right)$$

Thus, $\max_P \text{prob}\left(V \text{ accepts } x \text{ with } P\right) = \max_U \text{prob}\left\{z : |z| = p\left(|x|\right) \wedge (x, U, z) \in L'\right\}$ for every x. Therefore, V satisfies (1) and consequently $L \in$ *IP*.
"*IP'* $\subseteq \exists^1$*RP*": Let $L \in$ *IP'* and let V be a verifier satisfying (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 $U_P =_{\text{df}} \{\widehat{u}_1 01 \ldots 01 \widehat{u}_k : P(x, u_1, \ldots, u_k) = 1\}$. Hence, M accepts the input (x, U_P, z) if and only if V accepts x with P on path z. Therefore,

$$\text{prob}\left\{z : |z| = p\left(|x|\right) \wedge (x, U_P, z) \in L(M)\right\} = \text{prob}\left(V \text{ accepts } x \text{ with } P\right)$$

2. Let $U \subseteq \{0,1\}^*$ and define $P_U(x, u_1, \ldots, u_k) =_{\text{df}} c_U(\widehat{u}_1 01 \ldots 01 \widehat{u}_k)$. Hence, V accepts x with P_U on path z if and only if M accepts the input (x, U, z). Therefore,

$$\text{prob}\left(V \text{ accepts } x \text{ with } P_U\right) = \text{prob}\left\{z : |z| = p\left(|x|\right) \wedge (x, U, z) \in L(M)\right\}$$

Thus, $\max\limits_{U} \text{prob}\{z : |z| = p(|x|) \wedge (x, U, z) \in L(M)\} = \max\limits_{P} \text{prob}(V \text{ accepts } x \text{ with } P)$ for every x. Hence,

$$x \in L \rightarrow \exists^1 U\,(\text{prob}\{z : |z| = p(|x|) \wedge (x, U, z) \in L(M)\} = 1)$$
$$x \notin L \rightarrow \forall^1 U\,(\text{prob}\{z : |z| = p(|x|) \wedge (x, U, z) \in L(M)\} \leq 1/3)$$

Therefore, $L \in \exists^1$***RP***. ❑

These characterizations of ***MIP*** and ***IP*** have motivated us to study the set operators $\exists^2$ and $\exists^1$ (as well as $\forall^2$ and $\forall^1$) in more detail.

Finally let us mention that probabilistic operators of type 2 have been studied in [3].

4 Inclusion and Equivalence Rules and a Normal Form

To establish our Normal Form Theorem for the classes of the analytic polynomial-time hierarchy, we will use inclusion rules in the following sense: For $R, S \in \Gamma^*$, the *inclusion rule* $R \rightarrow S$ is valid if the replacement of the operator string R by the string S in any context does not diminish the class in question, i.e. $RQ\boldsymbol{P}^{\mu\tau(R)\tau(Q)} \subseteq SQ\boldsymbol{P}^{\mu\tau(S)\tau(Q)}$ for all $Q \in \Gamma^*$ and $\mu \in \{0,1,2\}^*$. We say that the *equivalence rule* $R \leftrightarrow S$ is valid if the replacement of the operator string R by the string S in any context does not change the class in question, i.e. if $RQ\boldsymbol{P}^{\mu\tau(R)\tau(Q)} = SQ\boldsymbol{P}^{\mu\tau(S)\tau(Q)}$ for all $Q \in \Gamma^*$ and $\mu \in \{0,1,2\}^*$. Obviously, we have $R \leftrightarrow S$ if and only if $R \rightarrow S$ and $S \rightarrow R$.

Throughout this paper, a rule $R \rightarrow S$ and "its complement" $\overline{R} \rightarrow \overline{S}$ have to be proved. The following proposition shows that we have to prove only one of them.

Proposition 4 (Complementation). *Let $R, S \in \Gamma^*$. If $R \rightarrow S$ then $\overline{R} \rightarrow \overline{S}$.*

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

$$\begin{aligned} L \in \overline{R}Q\boldsymbol{P}^{\mu\tau(R)\tau(Q)} &\Longrightarrow \overline{L} \in \text{co}\overline{R}Q\boldsymbol{P}^{\mu\tau(R)\tau(Q)} \\ &\Longrightarrow \overline{L} \in R\overline{Q}\boldsymbol{P}^{\mu\tau(R)\tau(Q)} \\ &\Longrightarrow \overline{L} \in S\overline{Q}\boldsymbol{P}^{\mu\tau(S)\tau(Q)} && \text{by } R \rightarrow S \\ &\Longrightarrow \overline{L} \in \text{co}\overline{S}Q\boldsymbol{P}^{\mu\tau(S)\tau(Q)} \\ &\Longrightarrow L \in \overline{S}Q\boldsymbol{P}^{\mu\tau(S)\tau(Q)} \end{aligned}$$

❑

Our first rules show relations between the existential (universal, respectively) operators of different types.

Lemma 5. *The following inclusion rules are valid:*

(1) $\epsilon \rightarrow \exists^0$ *and* $\epsilon \rightarrow \forall^0$*;*

(2) $\exists^0 \rightarrow \exists^1$ *and* $\forall^0 \rightarrow \forall^1$*;*

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

Proof. We prove the first rule of every statement, the other rules follow by complementation. Let $Q \in \Gamma^*$ and $\mu \in \{0,1,2\}^*$.

(1) This is the classical case of introducing a type 0 quantifier.

(2) Let $L \in \exists^0 Q\mathbf{P}^{\mu 0\tau(Q)}$. There exists an $L_1 \in \mathbf{P}^{\mu 0\tau(Q)}$ and a polynomial p such that

$$\begin{aligned} X \in L &\Longleftrightarrow \exists^0 u\, QY\, (|u| = p\,(|X|) \wedge (X, u, Y) \in L_1) \\ &\Longleftrightarrow \exists^1 U\, QY\, ((X, U, Y) \in L_2)\,, \end{aligned}$$

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 $\mu 0\tau\,(Q)$ 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, ... , $1^{p(|X|)}$ to the oracle U. Then, M' works as M on input $(X, \langle U, p\,(|X|)\rangle\,, Y)$. Therefore, $L\,(M') = L_2$ and $L_2 \in \mathbf{P}^{\mu 1\tau(Q)}$, i.e. $L \in \exists^1 Q\mathbf{P}^{\mu 1\tau(Q)}$.

(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 $\mu 2\tau(Q)$. ❑

The next lemma shows that set quantifiers applied to **P** are not more powerful than word quantifiers.

Lemma 6. *Let* $\mu \in \{0, 1, 2\}^*$. *Then*

(1) $\exists^0\mathbf{P}^{\mu 0} = \exists^1\mathbf{P}^{\mu 1} = \exists^2\mathbf{P}^{\mu 2}$;

(2) $\forall^0\mathbf{P}^{\mu 0} = \forall^1\mathbf{P}^{\mu 1} = \forall^2\mathbf{P}^{\mu 2}$.

Proof. We prove the first statement, the second statement follows by complementation. The inclusions "$\subseteq$" are valid by Lemma 5, thus only $\exists^2\mathbf{P}^{\mu 2} \subseteq \exists^0\mathbf{P}^{\mu 0}$ has to be proved. By definition $L \in \exists^2\mathbf{P}^{\mu 2}$ if and only if there exists an $L' \in \mathbf{P}^{\mu 2}$, such that $X \in L \Longleftrightarrow \exists^2 U\, ((X, U) \in L')$. Next, let M be a machine of type $\mu 2$ accepting L' and being time-bounded by some polynomial p, and let M' be a machine of type $\mu 0$ working on input (X, u) as M on input (X, U) with the following difference: If the i-th query of M to U has not been queried before then M' uses the i-th bit of u as the answer; otherwise M' uses the first answer to this query. Thus, $\exists^2 U\, ((X, U) \in L') \Longleftrightarrow \exists^0 u\, (|u| \leq p\,(|X|) \wedge (X, u) \in L\,(M'))$ and $L\,(M') \in \mathbf{P}^{\mu 0}$, i.e. $L \in \exists^0\mathbf{P}^{\mu 0}$. ❑

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

Lemma 7. *For* $\sigma, \tau \in \{0, 1, 2\}$ *the equivalence rules*

$$\exists^\sigma\exists^\tau \leftrightarrow \exists^{\min\{\sigma+\tau,2\}} \quad \textit{and} \quad \forall^\sigma\forall^\tau \leftrightarrow \forall^{\min\{\sigma+\tau,2\}}$$

are valid.

Proof. We prove the first rule, the second follows by complementation. Using Lemma 5, it is easy to see that we have to prove only the inclusion rules

(1) $\exists^0\exists^0 \rightarrow \exists^0$;

(2) $\exists^0\exists^1 \rightarrow \exists^1$ and $\exists^1\exists^0 \rightarrow \exists^1$;

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

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

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

(1) Let $L \in \exists^0\exists^0 Q\boldsymbol{P}^{\mu 00\tau(Q)}$. There exists an $L_1 \in \boldsymbol{P}^{\mu 00\tau(Q)}$ and polynomials p_1, p_2 such that

$$\begin{aligned} X \in L &\iff \exists^0 u\, \exists^0 v\, QY\left(|u| = p_1(|X|) \wedge |v| = p_2(|X|) \wedge (X,u,v,Y) \in L_1\right) \\ &\iff \exists^0 w\, QY\left(|w| = 2p_1(|X|) + 2p_2(|X|) + 2 \wedge (X,w,Y) \in L_2\right), \end{aligned}$$

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. $L \in \exists^0 Q\boldsymbol{P}^{\mu 0\tau(Q)}$.

(2) It suffices to prove the first rule, because of the obvious rule $\exists^0\exists^1 \leftrightarrow \exists^1\exists^0$. For a language $L \in \exists^0\exists^1 Q\boldsymbol{P}^{\mu 01\tau(Q)}$ there exists an $L_1 \in \boldsymbol{P}^{\mu 01\tau(Q)}$ and a polynomial p such that

$$\begin{aligned} X \in L &\iff \exists^0 v\, \exists^1 U\, QY\left(|v| = p(|X|) \wedge (X,v,U,Y) \in L_1\right) \\ &\iff \exists^1 W\, QY\left((X,W,Y) \in L_2\right), \end{aligned}$$

where $L_2 =_{\text{df}} \left\{(X,W,Y) : \left(X, \langle W, p(|X|)\rangle, 1^{p(|X|)}\backslash W, Y\right) \in L_1\right\}$. 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, ... , $1^{p(|X|)}$ to W. Then, M' works like M on input $\left(X, \langle W, p(|X|)\rangle, 1^{p(|X|)}\backslash W, Y\right)$ with the difference that instead of asking the query u to oracle $1^{p(|X|)}\backslash W$, the query $1^{p(|X|)}u$ 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

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

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 $\exists^2 \to \exists^1\exists^1\forall^0$. Let $L \in \exists^2 Q\boldsymbol{P}^{\mu 2\tau(Q)}$. There exists an $L_1 \in \boldsymbol{P}^{\mu 2\tau(Q)}$ such that for a suitable polynomial p

$$
\begin{aligned}
X \in L \Longleftrightarrow & \exists^2 U\, QY\, ((X,U,Y) \in L_1) \\
\Longleftrightarrow & \exists^2 U\, \exists^1 W \Big(\forall^0 u\, \forall^0 v \Big(|u|, |v| \le p(|X|) \to (u \in U \leftrightarrow v01\widehat{u} \in W) \Big) \\
& \qquad \wedge QY \Big((X,U,Y) \in L(M) \Big) \Big)
\end{aligned}
$$

(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)

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

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

$$
\Longleftrightarrow \exists^1 U\, \exists^1 W\, \forall^0 u\, \forall^0 v\, QY \Big(|u|, |v| \le p(|X|) \to (X,U,W,u,v,Y) \in L(M_L) \Big)
$$

(M_L on input (X,U,W,u,v,Y) first asks $u \in U$ and $v01\widehat{u} \in W$. 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 $v01\widehat{u}w$ 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 $\mu 1100 \tau(Q)$)

This shows $L \in \exists^1\exists^1\forall^0\forall^0 Q\boldsymbol{P}^{\mu 1100\tau(Q)}$ and, by Statement (1), $L \in \exists^1\exists^1\forall^0 Q\boldsymbol{P}^{\mu 110\tau(Q)}$.

Now, consider the class $\exists^2 Q_1^{\sigma_1} Q_2^{\sigma_2} \ldots Q_k^{\sigma_k} \boldsymbol{P}^{\mu 2\sigma_1\sigma_2\ldots\sigma_k}$, where $k \ge 0$, $Q_1, Q_2, \ldots, Q_k \in \{\exists, \forall\}$ and $\sigma_1, \sigma_2, \ldots, \sigma_k \in \{0,1,2\}$.

Case 1. If $Q_1 = Q_2 = \cdots = Q_k = \exists$ we can conclude by Lemma 5, Lemma 6, and (3)

$$
\exists^2\exists^{\sigma_1}\exists^{\sigma_2}\ldots\exists^{\sigma_k}\boldsymbol{P}^{\mu 2\sigma_1\sigma_2\ldots\sigma_k} \subseteq \exists^2\boldsymbol{P}^{\mu 2} \subseteq \exists^0\boldsymbol{P}^{\mu 0} \subseteq \exists^1\exists^1\exists^{\sigma_1}\exists^{\sigma_2}\ldots\exists^{\sigma_k}\boldsymbol{P}^{\mu 11\sigma_1\sigma_2\ldots\sigma_k}
$$

Case 2. If there exists an $l \in \{1, \ldots, k\}$ such that $Q_1 = Q_2 = \cdots = Q_{l-1} = \exists$ and $Q_l = \forall$, then we can conclude

$$
\begin{aligned}
& \exists^2\exists^{\sigma_1}\ldots\exists^{\sigma_{l-1}}\forall^{\sigma_l} Q_{l+1}^{\sigma_{l+1}}\ldots Q_k^{\sigma_k}\boldsymbol{P}^{\mu 2\sigma_1\sigma_2\ldots\sigma_k} \\
& \subseteq \exists^2\forall^{\sigma_l} Q_{l+1}^{\sigma_{l+1}}\ldots Q_k^{\sigma_k}\boldsymbol{P}^{\mu 2\sigma_l\sigma_{l+1}\ldots\sigma_k} && \text{by Lemma 5 and (3)} \\
& \subseteq \exists^1\exists^1\forall^0\forall^{\sigma_l} Q_{l+1}^{\sigma_{l+1}}\ldots Q_k^{\sigma_k}\boldsymbol{P}^{\mu 110\sigma_l\sigma_{l+1}\ldots\sigma_k} && \text{by the new rule} \\
& \subseteq \exists^1\exists^1\forall^{\sigma_l} Q_{l+1}^{\sigma_{l+1}}\ldots Q_k^{\sigma_k}\boldsymbol{P}^{\mu 11\sigma_l\sigma_{l+1}\ldots\sigma_k} && \text{by (1), (2) or Lemma 5 and (3)} \\
& \subseteq \exists^1\exists^1\exists^{\sigma_1}\ldots\exists^{\sigma_{l-1}}\forall^{\sigma_l} Q_{l+1}^{\sigma_{l+1}}\ldots Q_k^{\sigma_k}\boldsymbol{P}^{\mu 11\sigma_1\ldots\sigma_k} && \text{by Lemma 5}
\end{aligned}
$$

This completes the proof. ❑

The next result shows that word quantifiers can be eliminated if a set quantifier and at least one more quantifier follow.

Lemma 8. *For $\sigma \in \{1,2\}$ and $\tau \in \{0,1,2\}$ the equivalence rules*

$$\exists^0\forall^\sigma\exists^\tau \leftrightarrow \forall^\sigma\exists^\tau \quad \textit{and} \quad \forall^0\exists^\sigma\forall^\tau \leftrightarrow \exists^\sigma\forall^\tau$$

are valid.

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

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

$$X \in L \Longleftrightarrow \forall^0 u\, \exists^\sigma V\, QY\, (|u| = p\,(|X|) \to (X,u,V,Y) \in L_1)$$

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

$$X \in L \Longleftrightarrow \exists^\sigma W\, \forall^0 u\, QY\, (|u| = p\,(|X|) \to (X,W,u,Y) \in L_2)$$

"$\Rightarrow$": 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 $u \in \{0,1\}^{p(|X|)}$.
"$\Leftarrow$": 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 $L_2 \in \boldsymbol{P}^{\mu\sigma 0\tau(Q)}$, i.e. $L \in \exists^\sigma\forall^0 Q\boldsymbol{P}^{\mu\sigma 0\tau(Q)}$.

Now, using this rule and Lemma 7 we can conclude the rule $\forall^0\exists^\sigma\forall^\tau \to \exists^\sigma\forall^\tau$. The rule $\exists^\sigma\forall^\tau \to \forall^0\exists^\sigma\forall^\tau$ is valid by Lemma 5. ❑

Now 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 $Q_l =_{\mathrm{df}} \exists$ if l is odd and $Q_l =_{\mathrm{df}} \forall$ otherwise. For $k,m \geq 0$ and $\sigma_1,\sigma_2,\ldots,\sigma_k \in \{1,2\}$ we define

$$\mathcal{K}\,(\sigma_1\sigma_2\ldots\sigma_k, m) =_{\mathrm{df}} \exists^{\sigma_1}\forall^{\sigma_2}\ldots Q_k^{\sigma_k} Q_{k+1}^0 Q_{k+2}^0 \ldots Q_{k+m}^0 \boldsymbol{P}$$

Theorem 9 (Normal Form Theorem). *Every class of the analytic polynomial-time hierarchy coincides with one of the classes P, $\mathcal{K}\,(\sigma,m)$ or co$\mathcal{K}\,(\sigma,m)$, where $\sigma \in \{1,2\}^*$ and $m \geq 1$.*

Proof. We start from an arbitrary class of the analytic polynomial-time hierarchy. If this class is defined without operators, then it is *P*. Otherwise it coincides with one of the classes $\mathcal{K}\,(\sigma,m)$ or co$\mathcal{K}\,(\sigma,m)$ with $\sigma \in \{1,2\}^*$ and $m \geq 1$. This can be seen as follows: By Lemma 7, we bring the quantifier prefix in a form, where no operator substring $\exists^{\sigma_1}\exists^{\sigma_2}$ and $\forall^{\sigma_1}\forall^{\sigma_2}$ appears ($\sigma_1,\sigma_2 \in \{0,1,2\}$). By Lemma 6, we ensure that the last operator is a word operator. By Lemma 8, we eliminate all word operators which are not followed by a word operator. This last step can generate operator substrings $\exists^{\sigma_1}\exists^{\sigma_2}$ or $\forall^{\sigma_1}\forall^{\sigma_2}$ ($\sigma_1,\sigma_2 \in \{1,2\}$). However, applying repeatedly the rules of Lemmas 7 and 8 we get the desired result. ❑

5 Characterization by Well-Known Complexity Classes

In this section, we will prove that each of the classes of the analytic polynomial-time hierarchy coincides with a well-known complexity class. Classes built without set operators obviously coincide with classes of the (arithmetic) polynomial-time hierarchy, i.e. $\mathcal{K}(\epsilon, k) = \Sigma_k^{\mathrm{p}}$ and co$\mathcal{K}(\epsilon, k) = \Pi_k^{\mathrm{p}}$ for $k \geq 0$. Thus, the interesting cases are those where set operators are involved. The simplest classes of this type are the classes $\mathcal{K}(1,1) = \exists^1\forall^0 \textit{P}$ and co$\mathcal{K}(1,1) = \forall^1\exists^0 \textit{P}$, which turn out to coincide with *PSPACE*.

Theorem 10. $\exists^1\forall^0 \textit{P} = \forall^1\exists^0 \textit{P} = \textit{PSPACE}$.

Proof. Since *PSPACE* is closed under complementation, only $\exists^1\forall^0 \textit{P} = \textit{PSPACE}$ has to be proved. "$\textit{PSPACE} \subseteq \exists^1\forall^0 \textit{P}$": We use Cai and Furst's characterization of *PSPACE* [4]. Let A_5 be the group of even permutations on $[5] =_{\mathrm{df}} \{0, \dots, 4\}$, and let $\circ$ be the multiplication of this group. For $u \in \{0,1\}^n$, let $\overleftarrow{u} \in \{0,1\}^n$ denote the predecessor of u in $\{0,1\}^n$ in lexicographical order. Cai and Furst proved that for every language $L \in \textit{PSPACE}$ there exists a function $f : \{0,1\}^* \times \{0,1\}^* \to A_5$ computable in polynomial time and a polynomial p such that

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

Consequently,

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

However, for each (x, U, u, i, j) such that $u \notin \{0^{p(|x|)}, 1^{p(|x|)}\}$ the two queries $\overleftarrow{u}0^i$ and $u0^j$ are asked to U which is not a type 1 querying. To overcome this difficulty we encode the words from $(\{0,1\}^{p(|x|)} \setminus \{1^{p(|x|)}\}) \times \{0\}^{\leq 4}$ by an injective function α which has the property that, for every $u \notin \{0^{p(|x|)}, 1^{p(|x|)}\}$ and $0 \leq i,j \leq 4$, either $\alpha(\overleftarrow{u}, 0^i)$ is an initial word of $\alpha(u, 0^j)$ or vice versa. (For example take $\alpha(a_1 a_2 \dots a_m 0 1^{p(|x|)-m-1}, 0^i) =_{\mathrm{df}} 0^4 a_1 0^4 a_2 \dots 0^4 a_m 0^i$ where $m \geq 0$ and $a_1, a_2, \dots, a_m \in \{0,1\}$.) Now instead of querying $\overleftarrow{u}0^i$ and $u0^j$ to U the queries $\alpha(\overleftarrow{u}, 0^i)$ and $\alpha(u, 0^j)$ are made to the oracle $V =_{\mathrm{df}} \{\alpha(u, 0^i) : 0 \leq i \leq 4 \wedge u0^i \in U\}$ in a type 1 manner. Therefore,

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

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

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

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, $L_1 \in \boldsymbol{P}^{01000}$ and $L \in \exists^1\forall^0\forall^0\forall^0\boldsymbol{P}$. Now, using Lemma 7 we conclude $L \in \exists^1\forall^0\boldsymbol{P}$.

"$\exists^1\forall^0\boldsymbol{P} \subseteq$ *PSPACE*": Let $L \in \exists^1\forall^0\boldsymbol{P}$. There exists an $L_1 \in \boldsymbol{P}^{010}$ and a polynomial p such that $x \in L \Longleftrightarrow \exists^1 U\,\forall^0 u\,(|u| = p\,(|x|) \rightarrow (x, U, u) \in L_1)$. Since deterministic polynomial space is as powerful as nondeterministic polynomial space, it is 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, $\ldots$, $w_1w_2\ldots w_{q(|x|)}$. 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 oracle path. This has to be made in accordance with the guesses for the previous path. Let π_g be this path with the guessed answers.

2. For all $u \in \{0,1\}^{p(|x|)}$, M simulates M' on input (x, π_g, u) with the difference that, if M' asks a query from oracle path π, the guessed answer is used. If M' asks a query not belonging to this oracle path π, then M halts this simulation and signalizes positive. Otherwise, M signalizes positive if and only if the simulation of M' ends in an accepting state.

Finally, the machine M accepts if and only if for all path π all signals are positive. 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 $a^1 =_{\mathrm{df}} a$ and $a^0 =_{\mathrm{df}} 1 - a$.

Lemma 11. *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*

$$x \in L \Longleftrightarrow \exists a_1a_2\ldots a_{p(|x|)}\left(\bigwedge_{i=1}^{|x|}(a_i = x(i)) \wedge \bigwedge_{i=1}^{p(|x|)}\bigvee_{j=1}^{3} a_{f(1^{|x|},i,j)}^{g(1^{|x|},i,j)}\right)$$

Each class of the analytic polynomial-time hierarchy that we have not already characterized turns out to coincide with one of the classes of the exponential-time hierarchy. The next theorem shows which classes of the analytic polynomial-time hierarchy contain a level of the exponential-time hierarchy.

Theorem 12. *For $k \geq 1$ the following inclusions are valid:*

1. $\Sigma_k^{\exp} \subseteq \mathcal{K}\left(1^k, 2\right) \cap \mathcal{K}\left(1^{k-1}2, 1\right)$ *and;*

2. $\Pi_k^{\exp} \subseteq \mathrm{co}\mathcal{K}\left(1^k, 2\right) \cap \mathrm{co}\mathcal{K}\left(1^{k-1}2, 1\right)$.

Proof. We prove the first inclusion, the second follows by complementation. Consider the case that k is odd. Let $L \in \Sigma_k^{\exp}$, 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 exists a language $L_1 \in$ *NP* and a polynomial r such that

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

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

Let $n =_{\mathrm{df}} |x| + 1 + 2^{|x|}$ and $N =_{\mathrm{df}} n + (k-1) \cdot 2^{r(|x|)}$. By Lemma 11, there exist polynomials p, q and functions $f, g \in$ *DLOGTIME* such that

$$\begin{aligned} x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots \forall u_{k-1} \exists a_1 a_2 \ldots a_{p(N)} \Big(& \bigwedge_{i=1}^{|x|} (a_i = x(i)) \wedge (a_{|x|+1} = 1) \wedge \bigwedge_{i=|x|+2}^{n} (a_i = 0) \\ & \wedge \bigwedge_{j=1}^{k-1} \bigwedge_{i=1}^{2^{r(|x|)}} \left(a_{n+(j-1)\cdot 2^{r(|x|)}+i} = u_j(i)\right) \wedge \bigwedge_{i=1}^{q(N)} \bigvee_{j=1}^{3} a_{f(1^N,i,j)}^{g(1^N,i,j)} \Big) \end{aligned}$$

where the first $k-1$ quantifiers vary over words of length $2^{r(|x|)}$. Obviously, the functions $f'(x,i,j) =_{\mathrm{df}} f(1^N, i, j)$ and $g'(x,i,j) =_{\mathrm{df}} g(1^N, i, j)$ 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 of length $2^{r(|x|)}$ by an oracle. There exists a polynomial s such that

$$\begin{aligned} x \in L \Leftrightarrow\ & \exists U_1 \forall U_2 \ldots \forall U_{k-1} \exists U \Big(\bigwedge_{i=1}^{|x|} (i \in U \leftrightarrow x(i) = 1) \wedge ((|x|+1) \in U) \wedge \bigwedge_{i=|x|+2}^{n} (i \notin U) \\ & \wedge \bigwedge_{j=1}^{k-1} \bigwedge_{i=1}^{2^{r(|x|)}} \left((n + (j-1)\cdot 2^{r(|x|)} + i) \in U \leftrightarrow i \in U_j\right) \\ & \wedge \bigwedge_{i=1}^{2^{s(|x|)}} \bigvee_{j=1}^{3} (f'(x,i,j) \in U \leftrightarrow g'(x,i,j) = 1) \Big) \\ \Leftrightarrow\ & \exists^1 U_1 \forall^1 U_2 \ldots \forall^1 U_{k-1} \exists^1 U \forall^0 i \Big(1 \le i \le N + 2^{s(|x|)} \rightarrow \\ & \qquad \exists^0 j \left(1 \le j \le 3 \wedge (x, U_1, U_2, \ldots, U_{k-1}, U, i, j) \in L_2\right) \Big) \end{aligned}$$

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

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

It is obvious that L_2 can be accepted by a deterministic polynomial-time oracle machine which, on input $(x, U_1, U_2, \ldots, U_{k-1}, U, i, j)$, queries each of the oracles $U_1, U_2, \ldots, U_{k-1}, U$ at most once. Hence, $L_2 \in \boldsymbol{P}^{01^k00}$ and $L \in \mathcal{K}\left(1^k, 2\right)$.

However, since the quantifier $\exists^0 j$ ranges only over the set $\{1, 2, 3\}$, it can be eliminated:

$$x \in L \Leftrightarrow \exists^1 U_1 \, \forall^1 U_2 \ldots \forall^1 U_{k-1} \, \exists^1 U \, \forall^0 i \Big(1 \leq i \leq N + 2^{s(|x|)} \rightarrow (x, U_1, U_2, \ldots, U_{k-1}, U, i) \in L_3\Big)$$

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

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

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

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

The next theorem shows which classes of the analytic polynomial-time hierarchy are included in a level of the exponential-time hierarchy.

Theorem 13. *For $k, m \geq 1$ the inclusions*

$$\mathcal{K}\left(2^k, m\right) \cup \mathcal{K}\left(2^k 1, 1\right) \subseteq \Sigma_k^{\mathrm{exp}} \quad \textit{and} \quad \mathrm{co}\mathcal{K}\left(2^k, m\right) \cup \mathrm{co}\mathcal{K}\left(2^k 1, 1\right) \subseteq \Pi_k^{\mathrm{exp}}$$

are valid.

Proof. We prove the first inclusion, the second follows by complementation. Let $Q_k =_{\mathrm{df}} \exists$ if k is odd and $Q_k =_{\mathrm{df}} \forall$ otherwise.
"$\mathcal{K}\left(2^k, m\right) \subseteq \Sigma_k^{\mathrm{exp}}$": The inclusion $\mathcal{K}\left(2^k, m\right) \subseteq \exists^2 \forall^2 \ldots Q_k^2 \textit{PSPACE}^{02^k}$ is evident, since the last m operators, which are of type 0, can be easily simulate by a *PSPACE*-computation. However, $\exists^2 \forall^2 \ldots Q_k^2 \textit{PSPACE}^{02^k} \subseteq \Sigma_k^{\mathrm{exp}}$, since an alternating exponential-time machine with $k-1$ alternations which start with an existential state can accept a language from $\exists^2 \forall^2 \ldots Q_k^2 \textit{PSPACE}^{02^k}$.
"$\mathcal{K}\left(2^k 1, 1\right) \subseteq \Sigma_k^{\mathrm{exp}}$": Let k be even. The proof of $\exists^1 \forall^0 \boldsymbol{P} \subseteq \textit{PSPACE}$ (Theorem 10) remains valid if the machines have additionally k oracles of type 2, i.e. $\exists^1 \forall^0 \boldsymbol{P}^{02^k01} \subseteq \textit{PSPACE}^{02^k}$, and hence $\exists^2 \forall^2 \ldots \forall^2 \exists^1 \forall^0 \boldsymbol{P} \subseteq \exists^2 \forall^2 \ldots \forall^2 \textit{PSPACE}^{02^k} \subseteq \Sigma_k^{\mathrm{exp}}$. The case of odd k is treated analogously on the base of $\forall^1 \exists^0 \boldsymbol{P} \subseteq \textit{PSPACE}$. ❑

Now, we are ready to state our main result.

Theorem 14. *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 9, each class of the analytic polynomial-time hierarchy coincides either with $\boldsymbol{P}$ or with a class in normal form. Thus, we need only to prove that for $k \geq 0$, $m \geq 1$ and

$\sigma_1, \sigma_2, \dots, \sigma_k \in \{1, 2\}$, the class $\mathcal{K}(\sigma_1\sigma_2 \dots \sigma_k, m)$ coincides with one of the mentioned classes, since co$\mathcal{K}(\sigma_1\sigma_2 \dots \sigma_k, m)$ can be treated by complementation and $\boldsymbol{P} = \Sigma_0^{\mathrm{p}} = \Pi_0^{\mathrm{p}}$. However,

$$\mathcal{K}(\sigma_1\sigma_2 \dots \sigma_k, m) = \begin{cases} \Sigma_m^{\mathrm{p}}, & \text{if } k = 0 \\ \textit{PSPACE}, & \text{if } k = 1,\ \sigma_k = 1 \text{ and } m = 1 \\ \Sigma_{k-1}^{\exp}, & \text{if } k \geq 2,\ \sigma_k = 1 \text{ and } m = 1 \\ \Sigma_k^{\exp}, & \text{if } k \geq 1,\ \sigma_k = 2 \text{ and } m = 1 \\ \Sigma_k^{\exp}, & \text{if } k \geq 1 \text{ and } m \geq 2 \end{cases}$$

where the first line is valid by definition, the second by Theorem 10, and the remaining lines by Lemma 5, Theorem 12, and Theorem 13, which completes the proof. ❑

Finally, we present an algorithm which solves the problem investigated in this paper. It summarizes the proofs of Theorem 9 and Theorem 14.

Given: A class $\mathcal{K} = Q_1^{\tau_2} Q_1^{\tau_2} \dots Q_r^{\tau_r} \boldsymbol{P}$ of the analytic polynomial-time hierarchy where $r \geq 0$, $Q_1, Q_2, \dots, Q_r \in \{\exists, \forall\}$, and $\tau_1, \tau_2, \dots, \tau_r \in \{0, 1, 2\}$.
Question: With which complexity class $\mathcal{K}$ coincides?

Algorithm:
If $r = 0$ then $\mathcal{K} = \boldsymbol{P}$. Otherwise repeat the following steps until the form $\mathcal{K} = \mathcal{K}(\sigma_1\sigma_2 \dots \sigma_k, m)$ or $\mathcal{K} = $ co$\mathcal{K}(\sigma_1\sigma_2 \dots \sigma_k, m)$ for some $m \geq 1$, $k \geq 0$ and $\sigma_1, \sigma_2, \dots, \sigma_k \in \{1, 2\}$ is reached:

1. Eliminate all substrings $\exists^\sigma\exists^\tau$ and $\forall^\sigma\forall^\tau$ using the equivalence rules $\exists^\sigma\exists^\tau \leftrightarrow \exists^{\min\{\sigma+\tau,2\}}$ and $\forall^\sigma\forall^\tau \leftrightarrow \forall^{\min\{\sigma+\tau,2\}}$ (Lemma 7).

2. Replace the rightmost operator Q^τ with Q^0 (Lemma 6).

3. For $\sigma \in \{1, 2\}$ and $\tau \in \{0, 1, 2\}$, eliminate all substrings $\exists^0\forall^\sigma\exists^\tau$ and $\forall^0\exists^\sigma\forall^\tau$ using the equivalence rules $\exists^0\forall^\sigma\exists^\tau \leftrightarrow \forall^\sigma\exists^\tau$ and $\forall^0\exists^\sigma\forall^\tau \leftrightarrow \exists^\sigma\forall^\tau$ (Lemma 8).

Now get

$$\mathcal{K}(\sigma_1\sigma_2 \dots \sigma_k, m) = \begin{cases} \Sigma_m^{\mathrm{p}}, & \text{if } k = 0 \\ \textit{PSPACE}, & \text{if } k = 1,\ \sigma_k = 1 \text{ and } m = 1 \\ \Sigma_{k-1}^{\exp}, & \text{if } k \geq 2,\ \sigma_k = 1 \text{ and } m = 1 \\ \Sigma_k^{\exp} & \text{otherwise} \end{cases}$$

6 Conclusions

In the proof of Theorem 14 and the algorithm at the end of the previous section, we gave a complete characterization of all classes of the analytic polynomial-time hierarchy. On the other hand, we could improve Orponen's [7] characterization $\Sigma_k^{\exp} = \bigcup_{m\geq 1} \mathcal{K}(2^k, m)$ of k-th level of the exponential-time hierarchy by showing $\Sigma_k^{\exp} = \mathcal{K}(1^{k-1}2, 1) = \mathcal{K}(1^k, 2)$.

Let us further consider the following table that includes some results on the analytic polynomial-time hierarchy as well as the operator characterizations of interactive proof systems (Theorems 3 and 2). It shows the power of $\exists^1$ and $\exists^2$ when applied to the complexity classes P, RP, BPP, coNP, and Π_2^{p} (for which we know $P \subseteq RP \subseteq \mathrm{co}NP \subseteq \Pi_2^{\mathrm{p}}$ and $RP \subseteq BPP \subseteq \Pi_2^{\mathrm{p}}$).

$\exists^1 P = NP$	$\exists^2 P = NP$
$\exists^1 RP = PSPACE$	$\exists^2 RP = NEXPTIME$
$\exists^1 BPP = PSPACE$	$\exists^2 BPP = NEXPTIME$
$\exists^1 \mathrm{co}NP = PSPACE$	$\exists^2 \mathrm{co}NP = NEXPTIME$
$\exists^1 \Pi_2^{\mathrm{p}} = NEXPTIME$	$\exists^2 \Pi_2^{\mathrm{p}} = NEXPTIME$

Note that for the classes RP, BPP and coNP we get the same result when applying the operators $\exists^1$ and $\exists^2$:

$$\exists^1 RP = \exists^1 BPP = \exists^1 \mathrm{co}NP = PSPACE \quad \text{and} \quad \exists^2 RP = \exists^2 BPP = \exists^2 \mathrm{co}NP = NEXPTIME$$

However, these results are of completely different quality with respect to their relativizability: Whereas the results $\exists^1 RP = PSPACE$, $\exists^1 BPP = PSPACE$, $\exists^2 RP = NEXPTIME$, and $\exists^2 BPP = NEXPTIME$ are not valid in every relativized world [6] the results $\exists^1 \mathrm{co}NP = PSPACE$ and $\exists^2 \mathrm{co}NP = NEXPTIME$ are valid in every relativized world (as all the results on the analytic polynomial-time hierarchy of this paper).

Consider furthermore the equalities

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

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 $\exists^2 \mathrm{co}NP^{02} = NEXPTIME$ from which one can conclude only $P^{02} \neq NP^{02}$. This unequality 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 [2].

Acknowledgments. We would like to thank Heribert Vollmer, Würzburg, and Johannes Köbler, Ulm, for some helpful hints.

References

[1] 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.

[2] 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.

[3] 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.

[4] J. Cai and M. Furst. PSPACE survives constant-width bottlenecks. *International Journal of Foundations of Computer Sciences*, 2(1):67–76, 1991.

[5] 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.

[6] L. Fortnow and M. Sipser. Are there interactive protocols for co-NP languages? *Information Processing Letters*, 28:249–251, 1988.

[7] 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.

[8] A. Shamir. IP=PSPACE. In *31st Symposium on Foundations of Computer Science*, pages 11–15, 1990.

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

[10] 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.

[11] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3:23–33, 1977.