On Operators of Higher Types

Heribert Vollmer and Klaus W. Wagner*
Theoretische Informatik
Universität Würzburg
Am Exerzierplatz 3
D-97072 Würzburg, Germany
{vollmer|wagner}@informatik.uni-wuerzburg.de

Abstract

We discuss the use of operators of higher types in complexity theory. These are operators ranging over sets of words, i. e. over oracles. Depending on different oracle access mechanisms we consider two types of operators.

In particular, we examine existential, universal, and bounded-error probabilistic operators. We identify some of the emerging classes and we interpret recent results about interactive protocols in terms of these operators.

1. Introduction

Operators always played an important rôle in complexity theory, think e.g. of the characterization of the classes of the polynomial time hierarchy in terms of polynomial length-bounded existential and universal quantifiers [35, 38].

Toda's theorem [36] showing that the polynomial hierarchy is randomly reducible to $\oplus$ P and Turing reducible to PP made heavy use of operator theoretic arguments. This paper initiated a flourishing examination of relations among different operators. However, mostly only operators ranging over words of polynomial length were considered.

Nevertheless though not very well known, operators ranging over oracles (i.e., sets) have also been examined. Orponen [29] characterized the classes of the exponential time hierarchy by existential and universal operators ranging over oracles applied to the base classes PH (the union of all classes in the polynomial hierarchy) and PSPACE. Later Fortnow, Rompel, and Sipser [19] characterized the power of multi-prover interactive proof systems by an existential set operator applied to coRP.

In this paper we survey these and similar results. As it turns out, interactive proof systems with one prover can be characterized by an existential set operator applied to coRP, but here we have to use a restricted oracle access mechanism. We examine classes obtained from these operators as well as a bounded-error probabilistic (i.e., BP like) operator ranging over oracles.

In the last two sections of this paper we investigate whether results from the preceding sections remain valid when they are scaled down logarithmically. The problem whether the MIP $=$ NEXPTIME result [6] can be scaled down logarithmically to obtain a similar result for NP was one of the motivations in a sequence of papers [5, 18, 3, 2] which finally lead to the PCP characterization of NP. We will see that some of the NEXPTIME and PSPACE characterizations by operators of higher type have scaled down analogues while others lead to open questions about complexity classes in the range $\mathrm{NC}^1 \subseteq \mathrm{NL} \subseteq \mathrm{NTIME\text{-}SPACE}(n^{O(1)}, \log^{O(1)} n) \subseteq \mathrm{NP}$.

2. Characterizing IP and MIP

Fortnow, Rompel, and Sipser in 1988 characterized the power of multi-prover interactive proofs by an existential operator ranging over oracles.

Theorem 2.1 [19]. *A language L is in* MIP *if and only if there is a polynomial time probabilistic oracle Turing machine M such that*

- *if $x \in L$ then there is an oracle Y such that M with oracle Y accepts x with probability 1*
- *if $x \notin L$ then for all oracles Y, M with oracle Y will accept x with probability less than $\frac{1}{2}$.*

Proof sketch. (This proof sketch is taken from [30, p. 506].) Let $L \in$ MIP. Since there is no communication between the provers we can think of them as having agreed beforehand

*Work supported by DFG grant no. Wa847/1-2

on all answers each of them will give during the computation. These answers can then be expressed as an oracle.

On the other hand given M and Y as in the theorem, define the following MIP protocol for L: The verifier will follow the computation of M and ask one prover for the answers to the oracle queries. After the computation is done, the verifier will ask a second prover about a randomly chosen query in order to check that the first prover behaves according to a (fixed) oracle and not based on the previous interaction. Repeat this second step to accumulate high enough probability. ❑

What we have in this result is a characterization of MIP by an existential quantifier ranging over oracles which is applied to the class coRP. One can now raise the question if IP also allows a similar characterization. The answer is yes—we just have to take into account that a prover is not "memoryless" as a usual oracle but bases his answers on the preceding interaction. We can model this in terms of a special oracle access mechanism.

Theorem 2.2 [8]. *A language L is in* IP *if and only if there is a polynomial time probabilistic oracle Turing machine M such that*

- *M in any computation queries its oracle only in a specialized way, namely the series of queries is of the form $u_1, u_1u_2, u_1u_2u_3, \ldots$, i.e. every query is a prolongation of the previous query*
- *if $x \in L$ then there is an oracle Y such that M with oracle Y accepts x with probability 1*
- *if $x \notin L$ then for all oracles Y, M with oracle Y will accept x with probability less than $\frac{1}{2}$.*

Proof sketch. Let $L \in$ IP. Then L is accepted by an interactive proof system where for every $x \in L$ the verifier accepts with probability one. A probabilistic polynomial time oracle machine M simulates the verifier where the interaction with the prover is handled as follows: If the verifier asks the query u_k with previous queries $u_1, u_2, \ldots, u_{k-1}$, then M asks $u_1\#u_2\#u_3\#\cdots\#u_k$ and proceeds with the answer in the same way as the verifier does with the previous answers to u_k.

For the other direction, we simply let the verifier simulate M by handling its queries as follows: If M asks wu after previous queries w, then the verifier asks the prover 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 to $u(|u|)$ in the same way that M uses the answer to wu. ❑

We want to define the quantifiers implicit in the preceding two theorems more formally. Since later we are going to apply sequences of oracle quantifiers to some base class, we will need oracle machines with more than one oracle tape.

Definition 2.3. A relativized class $\mathcal{K}$ of type $\sigma_1 \cdots \sigma_k$ is given by a recursive enumeration $M_0, M_1, M_2, \ldots$ of oracle machines with k oracle tapes each where $\sigma_j \in \{0, 1, 2\}$ is the type of M_i's jth oracle ($i \geq 0$, $1 \leq j \leq k$). Here, an oracle of type 2 is a usual oracle with no restriction. An oracle of type 1 is an oracle where after every query the oracle tape is not erased (hence every query is a prolongation of the previous query). An oracle of type 0 is simply a word. Access to this word is by using the oracle tape as an index tape to address bits at specified positions. Now $L(M_i)$ consists of exactly those tuples $(x, X_1, \ldots, X_k)$ where M halts accepting on the "actual" input x with oracles $X_1, \ldots, X_k$ where X_i is of type σ_i.

For sake of clarity we explicitly remark that resource bounds are measured in the length of the actual input. In the case of space bounds, all oracle/index tapes are subject to the space restriction.

Now we define the following operators [8]:

Definition 2.4. Let $\mathcal{K}$ be a relativized class of type $\sigma_1 \cdots \sigma_k \sigma$. Then we define the following classes of type $\sigma_1 \cdots \sigma_k$: For $\sigma = 0$, $L \in \exists^0 \cdot \mathcal{K}$ iff there is a $B \in \mathcal{K}$ and a polynomial p such that

$$(x, X_1, \ldots, X_k) \in L \iff (\exists y, |y| = p(|x|)) \\ (x, X_1, \ldots, X_k, y) \in B,$$

and $L \in \forall^0 \cdot \mathcal{K}$ iff there is a $B \in \mathcal{K}$ and a polynomial p such that

$$(x, X_1, \ldots, X_k) \in L \iff (\forall y, |y| = p(|x|)) \\ (x, X_1, \ldots, X_k, y) \in B.$$

For $\sigma = 1$ or $\sigma = 2$, $L \in \exists^\sigma \cdot \mathcal{K}$ iff there is a $B \in \mathcal{K}$ such that

$$(x, X_1, \ldots, X_k) \in L \iff (\exists Y)(x, X_1, \ldots, X_k, Y) \in B,$$

and $L \in \forall^\sigma \cdot \mathcal{K}$ iff there is a $B \in \mathcal{K}$ such that

$$(x, X_1, \ldots, X_k) \in L \iff (\forall Y)(x, X_1, \ldots, X_k, Y) \in B.$$

In the following we will apply sequences of operators to base classes (mostly P) such that the resulting class is of type ϵ (without any oracle), i.e. in the terminology of Definition 2.3 we have $k = 0$ and we deal with a usual class of languages of simple words, not of tuples consisting of words and sets. The type of the base class will always be implicitly given from the context. If we write e.g. $Q_1^{\sigma_1} \cdots Q_k^{\sigma_k} \cdot$ P, then P is a class of languages of $k+1$ tuples $(x, X_1, \ldots, X_k)$ where for $1 \leq j \leq k$, X_j is of type σ_j.

Let BPP denote the class of all languages A for which there is some polynomial time probabilistic Turing machine M such that if $x \in A$ then M accepts x with probability at least $\frac{2}{3}$ and if $x \notin A$ then M rejects x with probability at least $\frac{2}{3}$. Let RP denote the class of all languages A for which there is some polynomial time probabilistic Turing machine M such that if $x \in A$ then M accepts x with probability at least $\frac{1}{2}$ and if $x \notin A$ then M rejects x with probability 1.

Since we know that IP = PSPACE [25, 33] and MIP = NEXPTIME [6], the above two theorems now read as follows:

Theorem 2.5. PSPACE = IP = $\exists^1 \cdot$ BPP = $\exists^1 \cdot$ coRP.

Theorem 2.6. NEXPTIME = MIP = $\exists^2 \cdot$ BPP = $\exists^2 \cdot$ coRP.

Since coRP is a subclass of coNP $= \forall^0 \cdot \mathrm{P}$ one can easily conclude the inclusions PSPACE $\subseteq \exists^1 \cdot \forall^0 \cdot \mathrm{P}$ and NEXPTIME $\subseteq \exists^2 \cdot \forall^0 \cdot \mathrm{P}$ (in fact we have equality in both cases). However, since the results 2.5 and 2.6 are not relativizable, this would be a nonrelativizing proof. But these equalities are even valid in every relativized world, as the next two theorems show.

Theorem 2.7 [8]. PSPACE = IP = $\exists^1 \cdot \forall^0 \cdot \mathrm{P} = \forall^1 \cdot \exists^0 \cdot \mathrm{P}$.

Proof sketch. The following (relativizable) characterization of PSPACE is known [16, 20]: A language L is in PSPACE iff there is a polynomial time computable function $f\colon \{0,1\}^* \times \{0,1\}^* \to A_5$ (A_5 is the group of all even permutations on 5 positions) and a polynomial p such that $x \in L \iff f(x,0) \circ f(x,1) \circ \cdots \circ f(x, 2^{p(x)} - 1)(1) = 1$. Now the sequence of intermediate results of the evaluation of this term can be encoded as an oracle. For all $i = 0, 1, \ldots, 2^{p(|x|)} - 1$ it has to be checked whether $f(x,i)$ transforms the previous intermediate result into the next one. This is a $\forall^0 \cdot \mathrm{P}$ predicate. Without further care however, the two necessary queries for every i are not of type 1, but one can overcome this difficulty by choosing an appropriate encoding of the queries (cf. the last step in the proof of Lemma 8.4). ❑

Theorem 2.8 [8]. NEXPTIME = MIP = $\exists^2 \cdot \forall^0 \cdot \mathrm{P}$.

Proof sketch. The proof builds on a translation of the completeness of 3SAT for NP under DLOGTIME-reductions into the exponential time context (see also Sect. 7 below, where we address translational methods more detailed, cf. especially 7.1). ❑

3. The Exists-Forall-Hierarchy

As already remarked earlier, the alternating exponential time hierarchy has been characterized by type 2 operators by Orponen [29]. Baier and Wagner [8] attacked the question what classes are obtained if an arbitrary sequence of universal and existential operators of all three types is applied to the base class P. This hierarchy of classes is called the *analytic polynomial-time hierarchy*. The following results, even sharpening some of the characterizations of [29], were obtained.

First, one gets a certain normal form:

Theorem 3.1. *Every class of the analytic polynomial-time hierarchy is either the class* P, *one of the classes* $\exists^{\sigma_1}\forall^{\sigma_2}\exists^{\sigma_3}\cdots Q_k^{\sigma_k} Q_{k+1}^0 \cdots Q_{k+m}^0 \mathrm{P}$ *for* $k \geq 0$, $m \geq 1$ *and* $\sigma_1, \cdots, \sigma_k \in \{1,2\}$ *(where* Q_j *is* $\exists$ *if* j *is odd and* Q_j *is* $\forall$ *if* j *is even), or the complement of one of these classes.*

Proof sketch. The proof consists of the following steps: First it is relatively easy to see that one single operator of any type applied to P is equivalently to the corresponding type 0 operator. Second, one can show that $\exists^\sigma\exists^\tau$ is equivalent (in this context) to $\exists^{\min\{\sigma+\tau,2\}}$ and $\forall^\sigma\forall^\tau$ is equivalent to $\forall^{\min\{\sigma+\tau,2\}}$. Third, a type 0 operator can be eliminated after a set quantifier if at least one more quantifier follows. Repeatedly applying these simplification rules results in the desired normal form. ❑

Next one can identify the classes in normal form. For this, we define the classes of the exponential time hierarchy (see e.g. [29]) as follows: For $k \geq 1$, let $\Sigma_k^{\exp} =_{\mathrm{def}} \mathrm{NEXPTIME}^{\Sigma_{k-1}^p}$ and $\Pi_k^{\exp} =_{\mathrm{def}} \mathrm{coNEXPTIME}^{\Sigma_{k-1}^p}$. (Equivalently, $\Sigma_k^{\exp}$ is the class of languages accepted by exponential time alternating Turing machines starting in an existential state and making at most $k-1$ alternations between existential and universal states.) Then the following holds: The class

$$\exists^{\sigma_1}\forall^{\sigma_2}\exists^{\sigma_3}\cdots Q_k^{\sigma_k} Q_{k+1}^0 \cdots Q_{k+m}^0 \mathrm{P}$$

is equal to

- Σ_m^p, if $k = 0$
- PSPACE, if $k = 1$, $\sigma_k = 1$, and $m = 1$ (see 2.7)
- $\Sigma_{k-1}^{\exp}$, if $k \geq 2$, $\sigma_k = 1$, and $m = 1$
- $\Sigma_k^{\exp}$, if $k \geq 1$, $\sigma_k = 2$, and $m = 1$
- $\Sigma_k^{\exp}$, if $k \geq 1$ and $m \geq 2$

For the proof of the latter three statements one has to express paths as oracles of appropriate type; the details can be found in [8].

Thus we identify all classes of the analytic polynomial-time hierarchy. Observe that the proof of the above theorem plus the just given case distinction gives even an effective procedure to do so.

Theorem 3.2. *Every class of the analytic polynomial-time hierarchy coincides with one of the following:* Σ_k^p, Π_k^p *(*$k \geq 0$*),* PSPACE, $\Sigma_k^{\exp}$ *or* $\Pi_k^{\exp}$ *(*$k \geq 1$*).*

4. The Bounded-Error Probabilistic Operator

Book, Vollmer, and Wagner in [14, 15] examined the bounded-error probabilistic operator of type 2. For convenience we also repeat the definition of the corresponding type 0 operator (see e.g. [32]).

Definition 4.1. Let $\mathcal{K}$ be a relativized class of type $\sigma_1 \cdots \sigma_k 0$. Then $\mathrm{BP}^0\mathcal{K}$ is a class of type $\sigma_1 \cdots \sigma_k$, and $L \in \mathrm{BP}^0\mathcal{K}$ iff there exists a set $B \in \mathcal{K}$ and a polynomial p such that $\mathrm{prob}_{|Z|=p(|x|)}((x, X_1, \ldots, X_k) \in L \leftrightarrow (x, X_1, \ldots, X_k, Z) \in B) \geq \frac{2}{3}$.

Definition 4.2. Let $\mathcal{K}$ be a relativized class of type $\sigma_1 \cdots \sigma_k 2$. Then $\mathrm{BP}^2\mathcal{K}$ is a class of type $\sigma_1 \cdots \sigma_k$, and $L \in \mathrm{BP}^2\mathcal{K}$ iff there exists a set $B \in \mathcal{K}$ such that $\mu_A\,[(x, X_1, \ldots, X_k) \in L \leftrightarrow (x, X_1, \ldots, X_k, A) \in B] \geq \frac{2}{3}$. The measure $\mu\colon 2^{\{0,1\}^\omega} \to [0,1]$ is the product measure based on $\mu_0\colon 2^{\{0.1\}} \to [0,1]$, where $\mu_0(\{0\}) = \mu_0(\{1\}) = \frac{1}{2}$. Here we identify as usual languages over $\{0,1\}$ as infinitely long sequences from $\{0,1\}^\omega$.

It turns out that classes defined by this operator have a close relationship to so called ALMOST classes.

Definition 4.3. Let $\mathcal{K}$ be a relativized class of type $\sigma_1 \cdots \sigma_k 2$. Then we say that a language L belongs to ALMOST-$\mathcal{K}$ if and only if the set of all A for which there exists a set $B \in \mathcal{K}$ such that $L = \{\, (x, X_1, \ldots, X_k) \mid (x, X_1, \ldots, X_k, A) \in B \,\}$ has measure one, i.e., $\mu_A\left[L \in \mathcal{K}^A\right] = 1$.

Thus, ALMOST is also a probabilistic type 2 operator. Let RAND denote the set of all random languages in the sense of Martin-Löf, see [13]. ALMOST-$\mathcal{K}$ can be characterized as the recursive part of $\mathcal{K}^{\mathrm{RAND}}$:

Theorem 4.4 [13]. *If $\mathcal{K}$ is closed under finite variations of the oracle, then*

$$\text{ALMOST-}\mathcal{K} = \mathcal{K}^{\mathrm{RAND}} \cap \mathrm{REC}.$$

Book even gave the following stronger result:

Theorem 4.5. *Let $\mathcal{K}$ be closed under finite variations of the oracle.*

1. *For every $B \in \mathrm{RAND}$ we have:*

$$\text{ALMOST-}\mathcal{K} = \mathcal{K}^B \cap \mathrm{REC}.$$

2. *For every pair of languages A, B such that $A \oplus B =_{\mathrm{def}} \{\, x0 \mid x \in A \,\} \cup \{\, x1 \mid x \in B \,\} \in \mathrm{RAND}$, we have:*

$$\text{ALMOST-}\mathcal{K} = \mathcal{K}^A \cap \mathcal{K}^B.$$

Taking a closer look at the above definitions we see that they just differ in the quantifier structure:

$$L \in \mathrm{BP}^2\mathcal{K} \iff (\exists B)(\forall x)\big(\mu_A[x \in L \leftrightarrow (x, A) \in B] \text{ is high}\big)$$
$$L \in \text{ALMOST-}\mathcal{K} \iff (\exists B)\big(\mu_A[(\forall x)x \in L \leftrightarrow (x, A) \in B] \text{ is high}\big)$$

Building on tools from probability theory (the Lebesgue Density Theorem and the Kolmogorov 0-1-Law) and a recursion theoretic argument by Sacks [31, page 272] one can in fact show that under certain assumptions a quantifier swapping is possible:

Theorem 4.6 [26, 14]. *If $\mathcal{K}$ is a relativized class which has the amplification property*[1] *and is closed under finite variations of the oracle, then*

$$\text{ALMOST-}\mathcal{K} = \mathrm{BP}^2 \cdot \mathcal{K}.$$

Thus what we have is an alternative characterization of ALMOST classes in terms of a BP like operator. Especially interesting results are obtained if we combine this characterization with the following result by Nisan and Wigderson stating that for classes from the polynomial time hierarchy type 2 quantifiers and usual word quantifiers coincide:

Theorem 4.7 [28]. *For $k \geq 0$,*

$$\mathrm{BP}^2 \cdot \Sigma_k^p = \mathrm{BP}^0 \cdot \Sigma_k^p$$
$$\mathrm{BP}^2 \cdot \Pi_k^p = \mathrm{BP}^0 \cdot \Pi_k^p$$

Now we have:

Theorem 4.8. *1.* ALMOST-L $= \mathrm{BP}^0 \cdot \mathrm{L}$ [27].

2. ALMOST-P = BPP [11, 1].
3. ALMOST-NP $= \mathrm{BP}^0 \cdot \mathrm{NP} = \mathrm{AM}$ [12, 28].
4. ALMOST-$\Sigma_k^p = \mathrm{BP}^0 \cdot \Sigma_k^p \subseteq \Pi_{k+1}^p$ *for* $k \geq 1$ [28, 32].
5. ALMOST-$\Pi_k^p = \mathrm{BP}^0 \cdot \Pi_k^p \subseteq \Sigma_{k+1}^p$ *for* $k \geq 1$ [28, 32].
6. ALMOST-PH = PH [28].
7. ALMOST-$\mathrm{P}^{\mathrm{QBF}}$ = ALMOST-$\mathrm{NP}^{\mathrm{QBF}}$ = ALMOST-$\mathrm{PH}^{\mathrm{QBF}}$ = PSPACE [17].
8. ALMOST-PP $= \mathrm{BP}^2\mathrm{PP}$.
9. ALMOST-PSPACE $= \mathrm{BP}^2\mathrm{PSPACE}$.

[1] We say that a relativized class $\mathcal{K}$ has the *amplification property*, if $L \in \mathrm{BP}^2\mathcal{K}$ iff for every polynomial p there exists an $B \in \mathcal{K}$ such that for all x, $\mu_A\,[x \in L \leftrightarrow (A, x) \in B] \geq 1 - 2^{-p(|x|)}$.

The proofs of Statements 3–6 rely on Theorem 4.7 as well as Schöning's results $\mathrm{BP}^0 \cdot \Sigma_k^p \subseteq \Pi_{k+1}^p$ and $\mathrm{BP}^0 \cdot \Pi_k^p \subseteq \Sigma_{k+1}^p$, which are generalizations of Lautemann [24] and Sipser's [34] inclusion $\mathrm{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$ [32]. Statement 9 allows a machine characterization of ALMOST-PSPACE—it corresponds to polynomial space on probabilistic checking stack automata, see [14]. Moreover it easily allows one to get the upper bound ALMOST-PSPACE $\subseteq \Sigma_2^{\exp} \cap \Pi_2^{\exp}$.

Baier [7] examined BP and ALMOST operators of type 1 and obtained results similar to the above. While it is known that for classes $\mathcal{K}$ from the polynomial time hierarchy the BP^2 and BP^0 operators coincide [28], he showed that for $\mathcal{K}$ from the exponential time hierarchy the BP^2 and BP^1 operators coincide. However, what still remains to be done is a thorough examination of the hierarchy obtained by all three types of $\exists$, $\forall$, and BP operators applied to P, similar to the examination of the restriction to type 0 in [32].

Finally we would like to mention that results about the BP^2 operator sometimes easily lead to consequences about measure 1 inclusions between complexity classes. For a survey, see [37].

5. Other operators ranging over words

In Definition 2.4 we defined type 0 operators as operators ranging over words of polynomial length. We now define type 0 like operators ranging over words of other fixed lengths.

Definition 5.1. Let $h\colon \mathbb{N} \to \mathbb{N}$. Let $\mathcal{K}$ be recursively presentable and of type $\sigma_1 \cdots \sigma_k 0$. Then we say that

1. $L \in \exists_h \mathcal{K}$ iff there is a $B \in \mathcal{K}$ such that
$$(x, X_1, \ldots, X_k) \in L \iff (\exists y, |y| = p(|x|))\ (x, X_1, \ldots, X_k, y) \in B.$$

2. $L \in \forall_h \mathcal{K}$ iff there is a $B \in \mathcal{K}$ such that
$$(x, X_1, \ldots, X_k) \in L \iff (\forall y, |y| = p(|x|))\ (x, X_1, \ldots, X_k, y) \in B.$$

3. $L \in \mathrm{BP}_h \mathcal{K}$ iff there exists a $B \in \mathcal{K}$ such that
$$\mathrm{prob}_{|y|=h(|x|)}\big[(x, X_1, \ldots, X_k) \in L \leftrightarrow (x, X_1, \ldots, X_k, y) \in B\big] \geq \tfrac{2}{3}.$$

4. $L \in \overline{\mathrm{R}}_h \mathcal{K}$ iff there is a $B \in \mathcal{K}$ such that
$$(x, X_1, \ldots, X_k) \in L \rightarrow \mathrm{prob}_{|y|=h(|x|)}\big[(x, X_1, \ldots, X_k, y) \in B\big] = 1$$
$$(x, X_1, \ldots, X_k) \notin L \rightarrow \mathrm{prob}_{|y|=h(|x|)}\big[(x, X_1, \ldots, X_k, y) \in B\big] \leq \tfrac{1}{2}.$$

Here, we assume as in Sect. 2 that the input y is always given to the underlying Turing machine on a random-access input tape and there is an index tape to address word positions. We use the superscript log to denote the union of all classes obtained by choosing $h \in O(\log n)$, p to denote the union of all classes obtained by choosing $h \in n^{O(1)}$, and exp to denote the union of all classes obtained by choosing $h \in 2^{n^{O(1)}}$. Thus, $\exists^p$, $\forall^p$, and BP^p are just the type 0 operators from Definitions 2.4 and 4.1. If $\mathcal{K}$ is a class defined by nondeterministic polynomial time machines (technically, $\mathcal{K}$ is *leaf language definable*, i.e, $\mathcal{K} = \mathrm{Leaf}^{\mathrm{P}}(B)$ for some set B [20, 23]), then $Q^2 \cdot \mathcal{K} = Q^{\exp} \cdot \mathcal{K}$ (for Q one of the above four operators) [15].

6. Restricting the number of queries

Let us now consider restrictions to the number of oracle queries or the number of spot checks on a random access input tape.

Definition 6.1. Let $k \geq 0$, let $\sigma_1, \sigma_2, \ldots, \sigma_k \in \{1, 2, \exp, \mathrm{p}, \log\}$, and let $r_1, r_2, \ldots, r_k\colon \mathbb{N} \to \mathbb{N}$. Let
$$L \in Q_1^{\sigma_1}[r_1] Q_2^{\sigma_2}[r_2] \cdots Q_k^{\sigma_k}[r_k] \cdot \mathcal{K}$$
iff $L \in Q_1^{\sigma_1} Q_2^{\sigma_2} \cdots Q_k^{\sigma_k} \cdot \mathcal{K}$ via a $\mathcal{K}$ machine M which on input $(x, X_1, \ldots, X_k)$ makes at most $r_i(|x|)$ queries to the i-th oracle or random access input tape (for $1 \leq i \leq k$). We omit $[r_i]$ if it does not constitute a real restriction (for example if it is greater than the runtime of the underlying machine).

A quick look at the proof sketches for Theorems 2.7 and 2.8 shows that we have even:

Theorem 6.2. *1.* PSPACE $= \exists^1 \cdot \forall^p \cdot \mathrm{P} = \exists^1[2] \cdot \forall^p \cdot \mathrm{P}$.

2. NEXPTIME $= \exists^2 \cdot \forall^p \cdot \mathrm{P} = \exists^2[3] \cdot \forall^p \cdot \mathrm{P}$.

In [9] it has been proved that this presentation of PSPACE and NEXPTIME cannot be improved with respect to the number of queries.

Theorem 6.3. *1.* $\exists^1[1] \cdot \forall^p \cdot \mathrm{P} = \exists^2[1] \cdot \forall^p \cdot \mathrm{P} = \mathrm{coNP}$.

2. $\exists^2[2] \cdot \forall^p \cdot \mathrm{P} = \mathrm{PSPACE}$.

7. Scaling down and up: NEXPTIME vs. NP

Knowing an inclusion result relating NP with another class it is usually easy to scale this up to obtain a similar result for NEXPTIME. This is done by translational methods (exponential padding). However, here we are in a different situation. We have a result for NEXPTIME such as

$$\begin{aligned}\mathrm{NEXPTIME} &= \exists^2 \cdot \forall^0 \cdot \mathrm{P} = \exists^2[3] \cdot \forall^0 \cdot \mathrm{P} \\ &= \exists^2 \cdot \mathrm{BP}^{\mathrm{p}} \cdot \mathrm{P} = \exists^2 \cdot \overline{\mathrm{R}}^{\mathrm{p}} \cdot \mathrm{P}\end{aligned}$$

Can we obtain similar results for NP? What would a similar result be in this context? The intuition is to reduce all involved resource bounds and and operator ranges logarithmically such that then, the original result can be obtained by translation (i.e., scaling up) of the down-scaled result.

However this will not work in each case in this simple form. Let us start with an example where it works.

Theorem 7.1.

$$\begin{aligned}\mathrm{NP} &= \exists^2 \cdot \forall^{\log} \cdot \mathrm{POLYLOGTIME} \\ &= \exists^2[3] \cdot \forall^{\log} \cdot \mathrm{DLOGTIME}.\end{aligned}$$

Proof sketch. The proof relies on the fact that 3-SAT is NP-complete under DLOGTIME-uniform projections. To accept a 3-SAT formula, the existential quantifier guesses an assignment. The universal quantifier guesses a clause. The DLOGTIME computation checks that the clause is satisfied by the assignment. Since we have 3 literals per clause, we get that every DLOGTIME-computation looks at at most 3 bits in the assignment. ❑

From this result one easily gets the result NEXPTIME $= \exists^2 \cdot \forall^{\mathrm{p}} \cdot \mathrm{P} = \exists^2[3] \cdot \forall^{\mathrm{p}} \cdot \mathrm{P}$ by translation. In fact, the proof of Theorem 2.8 does just this.

Next, how to scale down the NEXPTIME $= \exists^2 \cdot \mathrm{BPP} = \exists^2 \cdot \mathrm{coRP}$ result? Observe that it was just this question which was part of the motivation for the PCP-characterization of NP [3, 2], see also [5, 18, 4].

Formally scaling down the above NEXPTIME characterization would yield something like

$$\mathrm{NP} \stackrel{?}{=} \exists^2 \cdot \overline{\mathrm{R}}^{\log} \cdot \mathrm{DLOGTIME}$$

or

$$\mathrm{NP} \stackrel{?}{=} \exists^2 \cdot \overline{\mathrm{R}}^{\log} \cdot \mathrm{POLYLOGTIME}$$

But these equalities are not very likely because a polylogarithmic time computation cannot distinguish a "bad input" from a "good input" close to it (w.r.t. Hamming distance) with high probability.

Hence a corresponding characterization of NP must relax the strong time bound. The PCP result can be formulated in terms of our operators as follows:

Theorem 7.2 [3, 2].

$$\mathrm{NP} = \exists^2 \cdot \mathrm{BP}^{\log} \cdot \mathrm{P} = \exists^2[O(1)] \cdot \overline{\mathrm{R}}^{\log} \cdot \mathrm{P}.$$

However from this we cannot conclude the above NEXPTIME characterization, because the polynomial time bound is too large (this would result in an exponential time bound in the up-scaled theorem). One can reduce this time bound to polylogarithmic time if the input is encoded in an error-correcting code, see e.g. [5], [4, Theorem 2] or [22, Sect. 4.8]. The encoding is a polynomial time procedure, but one can hope that such an encoding is no more time consuming if an additional padding has to be encoded.

To make this precise we define *polynomial time strong many-one reducibility* between sets as follows: Let $A \in \{0,1\}^*\#^*$. (Later, # will be our padding symbol.) Then $A \leq_m^{p,s} B$ iff there exists a function f with the following properties: First, f is polynomially length bounded. Second, given (x, m) (m in binary) we can compute every bit of $f(x\#^m)$ in polynomial time. Finally, f reduces A to B, i.e. $x\#^m \in A \iff f(x\#^m) \in B$. The interesting case is that for two classes $\mathcal{K}_1$ and $\mathcal{K}_2$, we have $\mathcal{K}_2 \subseteq \mathcal{K}_1$ but for the converse we only know that every set in $\mathcal{K}_1$ is reducible to a set in $\mathcal{K}_2$ by strong many-one reductions. In this case we write $\mathcal{K}_1 \simeq \mathcal{K}_2$.

Open Question 1. *Is*

$$\mathrm{NP} \simeq \exists^2[O(1)] \cdot \overline{\mathrm{R}}^{\log} \cdot \mathrm{POLYLOGTIME}? \qquad (1)$$

From the statement under consideration one can easily conclude both the PCP theorem NP $= \exists^2[O(1)] \cdot \overline{\mathrm{R}}^{\log} \cdot \mathrm{P}$ and (by translation) the MIP theorem NEXPTIME $= \exists^2 \cdot \overline{\mathrm{R}}^{\mathrm{p}} \cdot \mathrm{P}$ (2.6). The reason for this is the following: Applying standard translational (padding) techniques, given a language in NEXPTIME the padded language in NP will (assuming (1)) reduce to a language in $\exists^2[O(1)] \cdot \overline{\mathrm{R}}^{\log} \cdot \mathrm{POLYLOGTIME}$ under strong many-one reductions. The reduction can actually be performed in time polynomial in the length of the prefix (without padding symbols) which is polylogarithmic in the input length. Translating this up shows NEXPTIME $\subseteq \exists^2 \cdot \overline{\mathrm{R}}^{\mathrm{p}} \cdot \mathrm{P}$.

The "equation" (1) should therefore be considered as a good scaled-down form of the MIP theorem. In fact, it implies even a stronger result.

Theorem 7.3. *If*

$$\mathrm{NP} \simeq \exists^2[O(1)] \cdot \overline{\mathrm{R}}^{\log} \cdot \mathrm{POLYLOGTIME},$$

then

$$\mathrm{NEXPTIME} = \exists^2[O(1)] \cdot \overline{\mathrm{R}}^{\mathrm{p}} \cdot \mathrm{P}.$$

8. Scaling down and up: PSPACE vs. ?

As done for NEXPTIME in the previous section we want to examine in this section scaled-down versions of the

PSPACE characterizations from Theorem 6.2 and 2.5:

$$\text{PSPACE} = \exists^1[2] \cdot \forall^p \cdot \text{P} = \exists^1 \cdot \forall^p \cdot \text{P} \tag{2}$$

$$\text{PSPACE} = \exists^1 \cdot \text{BP}^p \cdot \text{P} = \exists^1 \cdot \overline{\text{R}}^p \cdot \text{P} \tag{3}$$

As in the previous section we have to reduce all resource bounds and operator ranges logarithmically. But what is the scaled-down counterpart $\mathcal{K}$ of PSPACE? When we scale up $\mathcal{K}$ we should arrive at PSPACE, i.e. for every language A, $A \in$ PSPACE iff there is some $k \in \mathbb{N}$ such that $A_k \in \mathcal{K}$, where $A_k =_{\text{def}} \left\{ x\#^{2^{|x|^k}-|x|} \mid x \in A \right\}$. It is obvious that every class $\mathcal{K}$ such that $\text{NC}^1 \subseteq \mathcal{K} \subseteq$ POLYLOGSPACE fulfills this condition and hence is a scaled-down counterpart of PSPACE.

For the case of equation (2) this means: If we can prove

$$\begin{aligned} \text{NC}^1 &\subseteq \exists^1[2] \cdot \forall^{\log} \cdot \text{DLOGTIME} \\ &\subseteq \exists^1 \cdot \forall^{\log} \cdot \text{DLOGTIME} \\ &\subseteq \text{POLYLOGSPACE} \end{aligned}$$

or

$$\begin{aligned} \text{NC}^1 &\subseteq \exists^1[2] \cdot \forall^{\log} \cdot \text{POLYLOGTIME} \\ &\subseteq \exists^1 \cdot \forall^{\log} \cdot \text{POLYLOGTIME} \\ &\subseteq \text{POLYLOGSPACE} \end{aligned}$$

then (2) can be obtained from that result by translation via exponential padding. In fact we can even prove a stronger result. Let SC denote Steve's class, i.e. SC $=_{\text{def}}$ DTIME-SPACE$(n^{O(1)}, \log^{O(1)} n)$. Let NSC denote the nondeterministic analogue, i.e. NSC $=_{\text{def}}$ NTIME-SPACE$(n^{O(1)}, \log^{O(1)} n)$.

Theorem 8.1.

$$\begin{aligned} \text{NC}^1 &\subseteq \exists^1[2] \cdot \forall^{\log} \cdot \text{DLOGTIME} \\ &\subseteq \exists^1 \cdot \forall^{\log} \cdot \text{DLOGTIME} \\ &\subseteq \text{NL}. \end{aligned}$$

Theorem 8.2.

$$\begin{aligned} &\exists^1[3] \cdot \forall^{\log} \cdot \text{POLYLOGTIME} \\ &\quad = \exists^1 \cdot \forall^{\log} \cdot \text{POLYLOGTIME} \\ &\quad = \exists^1 \cdot \forall^{\log} \cdot \text{SC} = \text{NSC}. \end{aligned}$$

Note that NL $\subseteq$ NSC $\subseteq$ POLYLOGSPACE. Hence in this context, NSC is the natural scaled-down analogue of PSPACE.

Thus we see that applied to SC the quantifier sequences $\exists^1 \cdot \forall^{\log}$ and $\exists^1[O(1)] \cdot \forall^{\log}$ add exactly the power of nondeterminism. If this is also the case for logarithmic space is not clear to us.

Open Question 2. *Is* NL $= \exists^1 \cdot \forall^{\log} \cdot$ L *or even* NL $= \exists^1[O(1)] \cdot \forall^{\log} \cdot$ L*?*

The inclusion $\text{NC}^1 \subseteq \exists^1[2] \cdot \forall^{\log} \cdot$ DLOGTIME can be obtained with essentially the same proof as for Theorem 2.7 using Barrington's characterization of NC^1 [10]. All the other inclusions in Theorems 8.1 and 8.2 are immediate consequences of the following two lemmas.

Lemma 8.3. *For every* s *such that* $\log n \leq s(n) \leq n^{O(1)}$, *we have*

$$\begin{aligned} &\exists^1 \cdot \forall^{\log} \cdot \text{DTIME-SPACE}(n^{O(1)}, s) \\ &\qquad \subseteq \text{NTIME-SPACE}(n^{O(1)}, s). \end{aligned}$$

Proof. The proof is similar to the proof of IP $\subseteq$ PSPACE. We give a sketch. We simulate a computation of the form $\exists^1 \cdot \forall^{\log} \cdot$ DTIME-SPACE$(n^{O(1)}, s)$ by making a depth first search in the query tree. This means we first guess a queries-answer-sequence and check that this is the leftmost one in the tree of all possible such sequences. Then we check that if this is a sequence which actually occurs on one path of the $\forall^{\log} \cdot$ DTIME-SPACE$(n^{O(1)}, s)$ computation, then this computation path is accepting. Next, we consider the lexicographically next queries-answer-sequence, check again that if it actually occurs it leads to an accepting computation. In this way we search the whole query tree. All in all we guess an oracle, but since access to this oracle is type 1, we only have to store one path in the tree in order to be able to be consistent with previous answers (we do not have to store the whole oracle up to the maximal query length). Thus the simulation is space bounded by $s(n)$ and since s is bounded by a polynomial and we have only polynomially many $\forall^{\log} \cdot$ DTIME-SPACE$(n^{O(1)}, s)$ computations the overall time is also polynomial. ❑

Lemma 8.4. *Let* $\log n \leq s(n) \leq n$ *such that* s *is computable in* DTIME(n^2). *Then*

$$\begin{aligned} &\text{NTIME-SPACE}(n^{O(1)}, s) \\ &\qquad \subseteq \exists^1[3] \cdot \forall^{\log} \cdot \text{DTIME}(s \cdot \log). \end{aligned}$$

Proof. The proof consists of three main ingredients: First we show that NTIME-SPACE$(n^{O(1)}, s)$ can without loss of generality be assumed to be oblivious in a certain weak form. Using this we can then describe the behavior of such machines by certain 3-CNF formulae. Third, we show how to evaluate this formula in $\exists^1[3] \cdot \forall^{\log}$DTIME$(s \cdot \log)$. The main point here is that we have to force type 1 behavior while evaluating the formula.

Let $A \in$ NTIME-SPACE$(n^{O(1)}, s)$ be recognized by machine M. We can assume that M is oblivious in the following sense: The input head movements only depend on the length of the input and not the actual input itself, and the function mapping (in binary) (n, t) into $e(n, t)$, the position of the input head after t time steps of M on an input of length n, is computable in quadratic time. This can be achieved as follows:

Starting with an arbitrary NTIME-SPACE($n^{O(1)}, s$) machine N for A, M starts a simulation as follows: Given an input x of length n, first the binary representation of n is computed in an oblivious way (in time $O(n \log n)$). Then $s(n)$ is computed in binary (in time $O((\log n)^2)$) and then $s(n)$ tape cells are marked (in time $O(n \log n)$). In this phase the input head does not move.

Now M simulates N in an oblivious way: The worktape head makes on complete right scan over the marked region of $s(n)$ cells, then a complete left scan followed again by a complete right scan, and so on. The input head rests during such right-left-scan phases and moves one step when the worktape head returns to the left end of the marked region. In this way the input head performs on its own complete right-left-scans of the input. This behavior allows M to keep track of the input head position by a binary counter on a track of the worktape. One step of N can be simulated during a complete right-left-scan of the input head, i.e. in $O(n \cdot s(n))$ steps of M.

Next, we describe the behavior of M by a certain formula. Consider an input x, $|x| = n$. Let the runtime of M be bounded by n^k for $k \in \mathbb{N}$. Let $x(k)$ be the k-th symbol in x. Let $a(t, i)$ denote the contents of M's worktape at position i in step t, where we assume that M's state is attached to that tape symbol that is scanned by the worktape head; i.e. every $a(t, i)$ is either an alphabet symbol or a pair consisting of an alphabet symbol and a state of M. Let H be a predicate such that $H(a, b, c, d, e)$ iff according to M's Turing table, if the tape contents in a fixed time step in three neighboring cells is b, c, and d, and the input head scans a symbol e, then in the next time step the c will be replaced by an a. Similarly, let $H_1(a, b)$ iff a is the blank symbol and b is the first symbol of the input, let $H_2(a)$ iff a is an arbitrary symbol to which the accepting state is attached, and finally $H_3(a)$ iff a is a marker symbol for the right and left end of the used part of the tape. Now,

$$\begin{aligned} x \in A \iff \bigvee_a \Bigg(\bigwedge_{t=1}^{n^k} \bigwedge_{i=1}^{s(n)} & H\big(a(t,i), a(t-1, i-1), \\ & \quad a(t-1, i), a(t-1, i+1), \\ & \quad x(e(n,t))\big) \\ \wedge \bigwedge_{i=0}^{s(n)} & H_1\big(a(0,i), x(1)\big) \wedge H_2\big(a(n^k, 1)\big) \\ \wedge \bigwedge_{t=0}^{n^k} & \big(H_3\big(a(t,0)\big) \wedge H_3\big(a(t, s(n)+1)\big)\big)\Bigg) \end{aligned}$$

Next, we encode the values $a(t, i)$ by boolean vectors $b(t, i, 1) \cdots b(t, i, m)$ for suitable m (depending on the size of M's alphabet). The above formulae H, H_1, H_2, and H_3 can then be transformed into 3-CNFs of constant size, say with r clauses. Let $b(k)$ be the k-th bit of the binary string b. Define $0^0 = 1^1 = 1$ and $0^1 = 1^0 = 0$. Then there are functions h and a such that

$$x \in A \iff \bigvee_b \bigwedge_{t=0}^{n^k} \bigwedge_{i=0}^{s(n)+1} \bigwedge_{j=1}^{r} \bigvee_{k=1}^{3} b\big(h(t,i,j,k)\big)^{a(t,i,j,k)}$$

Here $h(t, i, j, i)$ is one of the triples $(t, i, \mu), (t-1, i-1, \mu), (t-1, i, \mu), (t-1, i+1, \mu)$ for $1 \le \mu \le m$ and $a(t, i, j, k) \in \{0, 1\}$. Observe that by the assumption of the theorem and by obliviousness, $x(e(n, t))$ is computable in time $\log^2 n \le s(n) \cdot \log n$; hence since the transition function is fixed h and a are computable with the same resources.

The only problem which remains to be fixed now is that in the just given formula the access to b is not type 1. To achieve this we encode b in a special form. We abbreviate $b(t, i, 1) \cdots b(t, i, m)$ by $b(t, i)$ and $b(t, 0) \cdots b(t, s(n) + 1)$ by $b(t)$. The length of every $b(t)$ is $L =_{\text{def}} |b(t)| = m \cdot (s(n) + 2)$. Observe that for every $t > 0, i, j$ the bits $b(h(t, i, j, 1))$, $b(h(t, i, j, 2))$, and $b(h(t, i, j, 3))$ can be found in $b(t-1)$ and $b(t)$. Thus what we have to accomplish is that we are able to ask for $b(t-1)$ and $b(t)$ (for $t > 1$) in a type 1 manner. Let 2^r be the smallest power of 2 greater than n^k. For $n^k < t < 2^r$ we set $b(t) =_{\text{def}} 0 \cdots 00$. Construct an oracle B as follows: Given t, $1 \le t < 2^r$, we want to encode $b(t)$ into B by putting one string into the oracle for every bit in $b(t)$ that is 1. First, let the length r binary representation of t be $u_1 \cdots u_{\ell-1} u_\ell 1 0^k$. Let $b(t) = b_1 b_2 \cdots b_L$. Then for every $1 \le i \le L$, set $0^L u_1 0^L u_2 0^L \cdots 0^L u_\ell 0^i \in B$ iff $b_i = 1$. Then we can recover $b(t)$ by asking the oracle for all $0^L u_1 0^L u_2 0^L \cdots 0^L u_\ell 0^i$, $1 \le i \le L$. Moreover, it can be seen that if we want to recover any $b(t-1)$ and $b(t)$ we only have to ask B in a type 1 manner. The length of the oracle queries is now bounded by $O(\log n \cdot s(n))$. ❑

Defining $SC^k =_{\text{def}}$ DTIME-SPACE($n^{O(1)}, \log^k n$) and $NSC^k =_{\text{def}}$ NTIME-SPACE($n^{O(1)}, \log^k n$) we thus have:

Corollary 8.5.

$$\begin{aligned} NSC^k &\subseteq \exists^1[3] \cdot \forall^{\log} \cdot \mathrm{DTIME}(\log^{k+1}) \\ &\subseteq \exists^1 \cdot \forall^{\log} \cdot SC^{k+1} \\ &\subseteq NSC^{k+1}. \end{aligned}$$

Finally we want to discuss the question about a scaled-down version of (3). From the discussion at the beginning of this section, it is clear that

$$\begin{aligned} NC^1 &\subseteq \exists^1 \cdot \overline{R}^{\log} \cdot \mathrm{DLOGTIME} \\ &\subseteq \exists^1 \cdot BP^{\log} \cdot \mathrm{DLOGTIME} \\ &\subseteq \mathrm{POLYLOGSPACE} \end{aligned}$$

or

$$\begin{aligned} NC^1 &\subseteq \exists^1 \cdot \overline{R}^{\log} \cdot \mathrm{POLYLOGTIME} \\ &\subseteq \exists^1 \cdot BP^{\log} \cdot \mathrm{POLYLOGTIME} \\ &\subseteq \mathrm{POLYLOGSPACE} \end{aligned}$$

implies the IP = PSPACE result (3). For the upper bounds we can prove as before:

Lemma 8.6. $\exists^1 \cdot \mathrm{BP}^{\log} \cdot \mathrm{SC} \subseteq \mathrm{NSC}$.

Proof. Analogous to the proof of Lemma 8.3. ❑

Corollary 8.7.

1. $\exists^1 \cdot \mathrm{BP}^{\log} \cdot \mathrm{DLOGTIME} \subseteq \exists^1 \cdot \mathrm{BP}^{\log} \cdot \mathrm{L} \subseteq \mathrm{NL}$.

2. $\exists^1 \cdot \mathrm{BP}^{\log} \cdot \mathrm{POLYLOGTIME}$
$\subseteq \exists^1 \cdot \mathrm{BP}^{\log} \cdot \mathrm{SC} \subseteq \mathrm{NSC}$.

However the inclusion

$$\mathrm{NC}^1 \subseteq \exists^1 \cdot \forall^{\log} \cdot \mathrm{POLYLOGTIME}$$

is not very likely for the same reasons as explained in Section 7 (see the discussion preceding Theorem 7.2). Again, we will have to allow some encoding of the input. This leads to the following question:

Open Question 3. *Does*

$$\mathrm{NC}^1 \leq_m^{p,s} \exists^1 \cdot \mathrm{BP}^{\log} \cdot \mathrm{POLYLOGTIME}$$

or even

$$\mathrm{NC}^1 \leq_m^{p,s} \exists^1[O(1)] \cdot \overline{\mathrm{R}}^{\log} \cdot \mathrm{POLYLOGTIME}?$$

Is

$$\begin{aligned} \mathrm{NSC} &\simeq \exists^1[O(1)] \cdot \overline{\mathrm{R}}^{\log} \cdot \mathrm{POLYLOGTIME} \\ &\simeq \exists^1 \cdot \mathrm{BP}^{\log} \cdot \mathrm{POLYLOGTIME}? \end{aligned}$$

Again, this statement would imply 2.5 immediately by translation. In fact, we would get even more:

Theorem 8.8. *If*

$$\mathrm{NC}^1 \leq_m^{p,s} \exists^1[O(1)] \cdot \overline{\mathrm{R}}^{\log} \cdot \mathrm{POLYLOGTIME},$$

then

$$\mathrm{PSPACE} = \exists^1[O(1)] \cdot \mathrm{BP}^{p} \cdot \mathrm{P}.$$

The following modified form of Open Question 3 can be considered as the *logspace variant of the PCP theorem.*

Open Question 4. *Is* $\mathrm{NL} = \exists^1 \cdot \mathrm{BP}^{\log} \cdot \mathrm{L}$ *or even* $\mathrm{NL} = \exists^1[O(1)] \cdot \overline{\mathrm{R}}^{\log} \cdot \mathrm{L}$*?*

References

[1] K. Ambos-Spies. Randomness, relativizations, and polynomial reducibilities. In *Proceedings 1st Structure in Complexity Theory*, volume 223 of *Lecture Notes in Computer Science*, pages 200–207. Springer-Verlag, 1986.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the intractability of approximation problems. In *Proceedings 33rd Symposium on the Foundations of Computer Science*, pages 14–23. IEEE Computer Society Press, 1992.

[3] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. In *Proceedings, 33rd Symposium on the Foundations of Computer Science*, pages 2–13. IEEE Computer Society Press, 1992.

[4] L. Babai. Transparent (holographic) proofs. In *Proceedings 10th Symposium on Theoretical Aspects of Computer Science*, volume 665 of *Lecture Notes in Computer Science*, pages 525–534. Springer-Verlag, 1993.

[5] L. Babai, L Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings 23rd Symposium on Theory of Computing*, pages 21–32. ACM Press, 1991.

[6] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. In *Proceedings 31st Symposium on Foundations of Computer Science*, pages 16–25. IEEE Computer Society Press, 1990.

[7] H. Baier. Type-1 probabilistic bounded-error operators. Manuscript, 1997.

[8] H. Baier and K. W. Wagner. The analytic polynomial-time hierarchy. Technical Report 148, Institut für Informatik, Universität Würzburg, 1996.

[9] H. Baier and K. W. Wagner. Restricting the number of queries in the analytical polynomial-time hierarchy. manuscript, 1997.

[10] D. M. Barrington. Bounded-with polynomial size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38:150–164, 1989.

[11] C. Bennett and J. Gill. Relative to a random oracle $\mathrm{P}^A \neq \mathrm{NP}^A \neq \mathrm{coNP}^A$ with probability 1. *SIAM Journal on Computing*, 10:96–113, 1981.

[12] R. V. Book. Some observations on separating complexity classes. *SIAM Journal on Computing*, 20:246–258, 1991.

[13] R. V. Book, J. Lutz, and K. W. Wagner. An observation on probability versus randomness with applications to complexity classes. *Mathematical Systems*, 27:201–209, 1994.

[14] R. V. Book, H. Vollmer, and K. W. Wagner. On type-2 probabilistic quantifiers. In *Proceedings 23rd International Colloquium on Automata, Languages and Programming*, volume 1099 of *Lecture Notes in Computer Science*, pages 369–380. Springer-Verlag, 1996.

[15] R. V. Book, H. Vollmer, and K. W. Wagner. Type 2 operators and ALMOST-classes. Technical Report 96-035, Electronic Colloquium on Computational Complexity, 1996.

[16] J.-Y. Cai and M. Furst. PSPACE survives constand-width bottlenecks. *International Journal of Foundations of Computer Science*, 2:67–76, 1991.

[17] Z. Dang and R. V. Book. On characterizations of Almost-R_{A}. Manuscript, 1996.

[18] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proceedings 32nd Symposium on Foundations of Computer Science*, pages 2–12. IEEE Computer Society Press, 1991.

[19] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. In *Proceedings 3rd Structure in Complexity Theory*, pages 156–161. IEEE Computer Society Press, 1988.

[20] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, and K. W. Wagner. On the power of polynomial time bit-reductions. In *Proceedings 8th Structure in Complexity Theory*, pages 200–207, 1993.

[21] H. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[22] S. Hougardy, H.-J. Prömel, and A. Steger. Probabilistically checkable proofs and their consequences for approximation algorithms. In W. Deubner, H.-J. Prömel, and B. Voigt, editors, *Trends in Discrete Mathematics*, volume 9 of *Topics in Discrete Mathematics*, pages 175–223. North Holland, 1995.

[23] B. Jenner, P. McKenzie, and D. Thérien. Logspace and logtime leaf languages. In *9th Annual Conference Structure in Complexity Theory*, pages 242–254, 1994.

[24] C. Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 117:215–217, 1983.

[25] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proceedings 31st Symposium on Foundations of Computer Science*, pages 2–10. IEEE Computer Society Press, 1990.

[26] W. Merkle and Y. Wang. Separations by random oracles and "Almost" classes for generalized reducibilities. In *Proceedings 20th Symposium on Mathematical Foundations of Computer Science*, volume 969 of *Lecture Notes in Computer Science*, pages 179–190. Springer-Verlag, 1995.

[27] N. Nisan. On read-once vs. multiple access to randomness in logspace. *Theoretical Computer Science*, 107:135–144, 1993.

[28] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[29] P. Orponen. Complexity classes of alternating machines with oracles. In *Proceedings 10th International Colloquium on Automata, Languages and Programming*, volume 154 of *Lecture Notes in Computer Science*, pages 573–584. Springer-Verlag, 1983.

[30] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[31] H. Rogers Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, 1967.

[32] U. Schöning. Probabilistic complexity classes and lowness. *Journal of Computer and System Sciences*, 39:84–100, 1989.

[33] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39:869–877, 1992.

[34] Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th Symposium on Theory of Computing*, pages 330–335, 1983.

[35] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In *Proceedings 5th ACM Symposium on the Theory of Computing*, pages 1–9, 1973.

[36] S. Toda. PP is as hard as the polynomial time hierarchy. *SIAM Journal on Computing*, 20:865–877, 1991.

[37] H. Vollmer and K. W. Wagner. Measure one results in computational complexity theory. In D.-Z. Du and K.-I. Ko, editors, *Advances in Algorithms, Languages, and Complexity*. Kluwer Academic Publishers, 1997.

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