Electronic Colloquium on Computational Complexity - Reports Series 1997 - available via:
ECCC
TR97-029
FTP: ftp.eccc.uni-trier.de:/pub/eccc/
WWW: http://www.eccc.uni-trier.de/eccc/
Email: ftpmail@ftp.eccc.uni-trier.de with subject 'help eccc'

On the Power of Las Vegas for One-way Communication Complexity, Finite Automata, and Polynomial-time Computations*

Pavol Ďuriš[1] and Juraj Hromkovič[2†] and José D. P. Rolim[3] and Georg Schnitger[4]

August 20, 1997

[1]Department of Computer Science, Comenius University,
Mlynská dolina, 842 15 Bratislava, Slovakia

[2]Lehrstuhl für Informatik I, RWTH Aachen,
Ahornstr. 55, 52 074 Aachen, Germany

[3]Centre Universitaire d'Informatique, Université de Genève,
1211 Genéve 4, Switzerland

[4]Fachbereich Informatik, Johann Wolfgang Goethe–Universität Frankfurt,
Robert Mayer Strasse 11–15, 60054 Frankfurt am Main, Germany

*Extended abstract of this paper has been presented at STACS'97.

†The work of this author has been supported by DFG Project HR 14/3-1.

Abstract

The study of the computational power of randomized computations is one of the central tasks of complexity theory. The main goal of this paper is the comparison of the power of Las Vegas computation and deterministic respectively nondeterministic computation. We investigate the power of Las Vegas computation for the complexity measures of one-way communication, finite automata and polynomial-time relativized Turing machine computation.

(i) For the one-way communication complexity of two-party protocols we show that Las Vegas communication can save at most one half of the deterministic one-way communication complexity.

We also present a language for which this gap is tight.

(ii) For the size (i.e., the number of states) of finite automata we show that the size of Las Vegas finite automata recognizing a language L is at least the root of the size of the minimal deterministic finite automaton recognizing L. Using a specific language we verify the optimality of this lower bound.

Note, that this result establishes for the first time an at most polynomial gap between Las Vegas and determinism for a uniform computing model.

(iii) For relativized polynomial computations we show that Las Vegas can be even more powerful than nondeterminism with a polynomial restriction on the number of nondeterministic guesses.

On the other hand superlogarithmic many advice bits in nondeterministic computations can be more powerful than Las Vegas (even Monte Carlo) computations in a relativized word.

Keywords: computational and structural complexity, Las Vegas, determinism, nondeterminism, communication complexity, automata

1 Introduction and Definitions

The comparative study of the computational power of nondeterministic, deterministic, and randomized computations is one of the central tasks of complexity theory. In this paper we focus on the relationship between Las Vegas and determinism and between Las Vegas and nondeterminism.

The relationship between the complexity classes P and ZPP, the class of languages accepted by polynomial-time Las Vegas Turing machines, is unresolved. The two classes coincide for two-party communication [12] and for the combinational complexity of non-uniform circuits and PRAMs [5].

Generally, for fundamental complexity measures one believes that the costs of Las Vegas computations are closer to the costs of deterministic computations than to the costs of nondeterministic ones. For time complexity this hypothesis suggests that P is a proper subset of NP, but ZPP is equal to P. We do not solve this central problem of complexity theory here, but we prove this hypothesis for two further complexity measures. An exponential gap between determinism (Las Vegas) and nondeterminism, and a quadratic gap between Las Vegas and determinism have been proved for the communication complexity of two-party protocols in [14, 12]. An at most polynomial gap between Las Vegas and determinism is known for the combinational complexity of circuits too.

We consider the P *versus* ZPP problem, respectively the "ZPP *versus* NP" problem for the following three complexity measures:

(i) message length in one-way communication complexity,

(ii) the size (i.e., the number of states) of finite automata, and

(iii) the time complexity of relativized polynomial-time Turing machine computations.

To define Las Vegas computations we follow [3]. In particular we consider *self-verifying nondeterminism* (introduced for communication complexity in [7]).

A self-verifying nondeterministic machine M is allowed to give three possible answers *yes, no, I do not know.* M is not allowed to make mistakes: If the answer is *yes*, then the input must be in $L(M)$. If the answer is *no*, then the input *cannot* be in $L(M)$. For every input there is at least one computation that does not finish with the answer "*I do not know.*"

We say that M is a Las Vegas machine recognizing a language L if and only if M is a self-verifying nondeterministic machine recognizing L and for each input the answer "*I do not know*" is given with probability at most $\frac{1}{2}$.

Obviously, the difference between self-verifying nondeterminism and nondeterminism is that the negative answer of a nondeterministic machine gives no information about the relationship of the input in $L(M)$. The answer *no* only means that the machine has not succeeded to prove that the input is in $L(M)$. For self-verifying nondeterminism the answer *no* means that M has proved in this computation that $x \notin L(M)$.

We observe that the concept of selfverification is general and can be applied to almost all computing models. Selfverification allows a natural definition of Las Vegas computation and even the difference between Las Vegas- and Monte-Carlo computation is is comparable to the difference between self-verifying nondeterminism and standard nondeterminism.

Self-verifying nondeterminism is of independent interest, since it provides a natural concept for a comparative study of the complexities of solution verification, search for a solution and proving the nonexistence of solutions:

1. The complexity of deterministic computations is the maximum of {complexity of the search for a solution, complexity of the proof of the nonexistence of any solution}.

2. The complexity of nondeterministic computations is the complexity of verifying that a guessed candidate for a solution is a correct solution.

3. The complexity of self-verifying nondeterminism is the maximum of {complexity of verification of a guessed candidate, complexity of a proof of the nonexistence of any solution}.

As a byproduct we also compare deterministic and nondeterministic computations with not just Las Vegas computations, but also general self-verifying nondeterministic computations. The main results of this paper are as follows:

(i) **One-way communication complexity**

We consider the one-way version of two-party protocols as introduced by Yao [15] for a fixed partition of the input. Computer C_I receives the first half of the input and computer C_{II} receives the rest. Informally, a deterministic one-way protocol P determines first the message sent from computer C_I to computer C_{II} and then decides, whether C_{II} accepts or rejects.

The **one-way communication complexity of P, $\mathbf{cc_1}(P)$**, is the length of the longest message sent by C_I. Finally for a a Boolean function f, $\mathbf{cc_1}(f)$ denotes the one-way communication complexity of the best protocol computing f. Let $\mathbf{ncc_1}(f)$ [resp. $\mathbf{svncc_1}(f)$, $\mathbf{lvcc_1}(f)$] denote the one-way nondeterministic [resp. self-verifying, Las Vegas] communication complexity of f[1].

In our main result of this part we show that

$$\mathrm{lvcc}_1(f) \geq \mathrm{cc}_1(f)/2$$

for every Boolean function f. This result is quite surprising, because there is a quadratic gap between Las Vegas and determinism for the general (two-way) model of communication complexity [12]. Moreover, for a specific language $L \subseteq \{0,1\}^*$, we show that $\mathrm{cc}_1(h_n(L)) = 2 \cdot \mathrm{lvcc}_1(h_n(L))$, where $h_n(L)$ is the Boolean function of n variables defined by $h_n(L)(\alpha) = 1$ iff $\alpha \in$

[1] For formal definitions and details see the monographs on communication complexity [9, 10].

$L \cap \{0,1\}^n$ and hence our relationship between Las Vegas and deterministic one-way communication is best possible.

It is well known that there exist languages A with an exponential gap between $\mathrm{cc}_1(h_n(L))$ and $\mathrm{ncc}_1(h_n(L))$. Here, we show that there is a language L with an exponential gap between $\mathrm{cc}_1(h_n(L))$ and $\mathrm{svncc}_1(h_n(L))$. Thus, self-verifying nondeterminism may be much more powerful than determinism. As a consequence we have found another substantial difference between one-way and two-way communication complexity, where self-verifying nondeterminism is polynomially related to determinism [1, 7].

(ii) **Finite automata**

We consider the model of one-way finite automata. In the following $L(A)$ denotes the language accepted by the computing device A. We also consider self-verifying nondeterministic finite automata (*SNFA*) as nondeterministic automata whose states are partitioned into three disjoint groups: accepting states, rejecting states, and neutral states. An input word is accepted (rejected) by an $SNFA$ if there exists a computation finishing in an accepting (rejecting) state. Moreover, for no input there exist computations finishing in both accepting and rejecting states and for each input at least one computation is accepting or rejecting.

We introduce a Las Vegas finite automaton (*LVFA*) as a *SNFA* A which for any $x \in L(A)$ reaches an accepting state with probability at least $\frac{1}{2}$ and which for any $x \notin L(A)$ reaches a rejecting state with probability at least $\frac{1}{2}$. (The probability of a computation of a $LVFA$ is defined through the transition probabilities of the automaton.)

For any regular language L we define $\mathbf{s}(\boldsymbol{L})$, $\mathbf{ns}(\boldsymbol{L})$, $\mathbf{svns}(\boldsymbol{L})$ and $\mathbf{lvs}(\boldsymbol{L})$ respectively as the size of a minimal deterministic, nondeterministic, self-verifying nondeterministic and Las Vegas finite automaton for L.

The main result of this part shows that

$$\mathrm{lvs}(L) \geq \sqrt{\mathrm{s}(L)}$$

for every regular language L. The optimality of this lower bound on $\mathrm{lvs}(L)$ is verified by constructing a language L' with $\mathrm{s}(L') = \Omega((\mathrm{lvs}(L'))^2)$. Note, that this is the first result showing a polynomial relation between Las Vegas and determinism for a uniform computing model.

It is well known that there are regular languages with $\mathrm{s}(L) \sim 2^{\mathrm{ns}(L)}$. Here, we show that for some regular languages A,B there are exponential gaps between $\mathrm{s}(A)$ and $\mathrm{svns}(A)$, and between $\mathrm{svns}(B)$ and $\mathrm{ns}(B)$.

(iii) **Relativized polynomial-time computations**

It is well known that Las Vegas may be more powerful than determinism in a relativized world. We strengthen this result by showing that Las Vegas computations may be even more powerful than nondeterministic computations with at most $f(n)$ advice bits (nondeterministic decisions) for any f bounded by a polynomial. This shows the existence of a relativization for which any polynomial restriction on the number of nondeterministic guesses (advice bits) essentially decreases the power of polynomial-time nondeterministic computations.

On the other hand, for any polynomial-time constructable function h growing asymptotically faster than $\log_2 n$, there exists an oracle B such that polynomial-time nondeterministic computations with oracle B and at most $h(n)$ advice bits are more powerful than polynomial-time two-sided error Monte Carlo computations with oracle B. Thus, there exists a relativization for which a small (i.e., superlogarithmic) number of nondeterministic guesses gives more power than Monte Carlo (Las Vegas) computations with any number of random bits. The last result solves an open question stated in [6].

This paper is organized as follows. Section 2 is devoted to the formal presentation of the results and proofs that are not too technical. In Section 3 we give the technical proofs of the four main results (i.e., $\mathrm{lvcc}_1(f) \geq \mathrm{cc}_1(f)/2$ for every f, $\mathrm{lvs}(\mathrm{h}_n(L)) \geq \sqrt{\mathrm{s}(\mathrm{h}_n(L))}$ for every regular language L, and the two relativization results).

2 Results

We present the results in the sequence (i), (ii), (iii) as in Introduction.

2.1 One-way Communication Complexity

In this section we consider one-way Las Vegas protocols with a public[1] random source [13]. Note that this strengthens the lower bound results because the lower bounds on randomized protocols with public random sources are also lower bounds on randomized protocols with private random sources. Let, for any Boolean function $f : \{0,1\}^{2n} \to \{0,1\}$, $\boldsymbol{M(f)} = [a_{u,v}]_{u,v \in \{0,1\}^n}$ with $a_{u,v} = f(uv)$ be the **communication matrix representation of $\boldsymbol{f}$**.

First, we present our main result whose proof is given in Section 3.1.

Theorem 2.1.1 *For every Boolean function f*

$$\mathrm{lvcc}_1(f) \geq \mathrm{cc}_1(f)/2.$$

To show that the lower bound of Theorem 2.1.1 cannot be improved we consider the language

$$L = \{xy \in \{0,1\}^* \mid |x| = |y| \text{ and if } y = 0^i 1 z, \text{ then } x_{i+1} = 1\}.$$

Theorem 2.1.2 *For every positive integer n,*

[1] called also common random source in some papers

(i) $\mathrm{cc}_1(h_{4n}(L)) = 2n$,

(ii) $\mathrm{lvcc}_1(h_{4n}(L)) = n$, *and*

(iii) $\lceil \log_2 2n \rceil \leq \mathrm{svncc}_1(h_{4n}(L)) \leq \lceil \log_2 2n \rceil + 1$.

Proof: It is well-known, that for every Boolean function f $\mathrm{cc}_1(f)$ is equal to the logarithm of the number of different rows in $M(f)$ (see [1, 9]). Since there are no two identical rows in $M(h_{4n}(L))$ and $M(h_{4n}(L))$ has 2^{2n} rows, the result (i) follows.

We obtain a Las Vegas protocol for $h_{4n}(L)$ exchanging n bits as follows. The first computer flips an unbiased coin and sends accordingly the first respectively the second half of its input. Obviously the second computer is now able to determine the result with probability $\frac{1}{2}$. Because of (i) and Theorem 2.1.1 there is no better protocol.

The fact $\mathrm{svncc}_1(h_{4n}(L)) \geq \lceil \log_2 2n \rceil$ is obvious because $\mathrm{svncc}_1(h_{4n}(L)) \geq \lceil \log_2(\mathrm{cc}_1(f)) \rceil$ for every f [14]. On the other hand, $\lceil \log_2 2n \rceil + 1$ bits suffice for a self-verifying protocol, if processor C_I guesses a position $j \in \{1, \ldots, 2n\}$ and sends the binary code of j and the bit x_j to C_{II}. C_{II} knows the crucial bit position and gives a binding answer if position j matches. 2

Thus, Theorem 2.1.2 shows not only the optimality of the lower bound of Theorem 2.1.1, but also a surprising exponential gap between determinism and self-verifying nondeterminism. Note, that this exponential gap cannot be improved because $\mathrm{ncc}_1(f) \leq 2^{\mathrm{cc}_1(f)}$ for every Boolean function f. To show also an exponential gap between nondeterminism and self-verifying nondeterminism, it suffices to consider the language $ID^C = \{xy \in \{0,1\}^* \mid x \neq y, |x| = |y|\}$.

Observation 2.1.3 *For every positive integer n,*

(i) $\mathrm{ncc}_1(h_{2n}(ID^C)) \leq \lceil \log_2 n \rceil + 1$, *and*

(ii) $\mathrm{svncc}_1(h_{2n}(ID^C n)) = n$.

Proof: (i) is obvious because C_I can guess a position j in which the inputs of C_I and C_{II} differ. So, the message consists of the binary code of j and of the j-th bit of the input of C_I.

Obviously, for even input lengths the identity language $ID = \{xx \mid x \in \{0,1\}^*\}$ is the complement of ID^C. Since it is well-known that $\mathrm{ncc}_1(h_{2n}(ID)) = n$ (see, for instance [9, 10]) and

$$\mathrm{svncc}_1(f) \geq \max\{\mathrm{ncc}_1(f), \mathrm{ncc}_1(f^C)\}$$

for every function f, the equality (ii) follows. 2

2.2 Finite Automata

First we give our main result showing a quadratic gap between Las Vegas and determinism. The proof of this theorem is given in Section 3.2.

Theorem 2.2.1 *For every regular language L*

$$\mathrm{lvs}(L) \geq \sqrt{\mathrm{s}(L)}.$$

The language $L_k = \{w \in \{0,1\}^* \mid w = u1v \text{ and } |v| = k-1\}$ is a well known example of a language producing an exponential gap between $\mathrm{s}(L)$ and $\mathrm{ns}(L)$ [11]. We use L_k to show that Theorem 2.2.1 cannot be improved. But first we give the following useful observation expressing the typical property of self-verifying nondeterminism.

Observation 2.2.2 *For any regular language L:*

$$\max\{\mathrm{ns}(L), \mathrm{ns}(L^C)\} \leq \mathrm{svns}(L) \leq 1 + \mathrm{ns}(L) + \mathrm{ns}(L^C).$$

Proof: Every SNFA A can be changed to an NFA B by adding the neutral states to the set of rejecting states. Obviously $L(A) = L(B)$ and $s(A) = s(B)$. If one takes the rejecting states of A as accepting ones of an NFA C, and the

accepting and neutral states as the rejecting ones of C, then $L(C) = (L(A))^C$. So, $\text{svns}(L) \geq \max\{\text{ns}(L), \text{ns}(L^C)\}$.

Let E, and F be NFAs such that $L(E) = (L(F))^C$. A SNFA D can be constructed as follows. D connects a new initial state via ε-moves to the initial states of E and F. Finally D chooses as accepting states the set of accepting states of E, as rejecting states the set of accepting states of F, and makes the remaining states of E and F neutral. Then obviously $L(E) = L(D)$. 2

Theorem 2.2.3 *For every positive integer k,*

(i) $\text{s}(L_k) = 2^k$,

(ii) $\text{lvs}(L_k) \leq 4 \cdot 2^{k/2} = O(\sqrt{\text{s}(L_k)})$,

(iii) $\text{svns}(L_k) \leq 2k + 3$, *and*

(iv) $\text{ns}(L_k) = k + 1 = \text{ns}(L_k^C)$.

Proof: (i) and (iv) are well-known facts. (iii) follows immediately from Observation 2.2.2 and from (iv). To show (ii) we consider the following strategy of a LVFA A. A randomly guesses whether the important bit (the k-th bit from the end) is on an even or odd bit position. Now, one can easily construct a deterministic FA of $2^{k/2+1}$ states accepting $L_k^{odd} = \{w \in \{0,1\}^* \mid w = u1v, |v| = k - 1, \text{ and } |u| \text{ is odd}\}$ or $L_k^{even} = \{w \in \{0,1\}^* \mid w = u1v, |v| = k-1, \text{ and } |u| \text{ is even}\}$. 2

Again we see that self-verifying nondeterminism may be much more powerful than determinism (resp. Las Vegas). If one wishes to demonstrate a large difference between self-verifying nondeterminism and nondeterminism, one has to choose a regular language with a large difference between $\text{sn}(L)$ and $\text{sn}(L^C)$. We consider for every $m \in N$ the language

$$U_m = \{u0v1w, u1v0w \mid |v| = m - 1, uvw \in \{0,1\}^*\}.$$

Observation 2.2.4 *For every* $m \in N$

(i) $\mathrm{ns}(U_m) \leq 2m + 2$,

(ii) $\mathrm{ns}({U_m}^C) \geq 2^m$, *and*

(iii) $\mathrm{svns}((U_m)) \geq 2^m$.

Proof: (i) is obvious. Since $\{xx \mid x \in \{0,1\}^m\} = U_m^C \cap \{0,1\}^{2m}$ every NFA accepting U_m^C must be in different states after reading two different words $y, z \in \{0,1\}^m$. Since $|\{0,1\}^m| = 2^m$ we obtain (ii). (iii) is a direct consequence of (ii) and Observation 2.2.2. 2

2.3 Polynomial-time Turing Machines

Here we consider nondeterministic and probabilistic polynomial-time bounded complexity classes as defined in [2]. To be exact in the formulation of the results we start with some formal definitions [3].

Definition 2.3.1 *A* **probabilistic Turing machine** *is a nondeterministic Turing machine such that*

(i) the local degree of nondeterminism is bounded by two (for each argument at most two different actions are possible), and in every step the machine takes a choice from exactly two possibilities,

(ii) the machine is clocked by some constructible function, and the number of steps in each computation is exactly the number of steps allowed by the clock, and

(iii) every computation ends in a final state, which can be either accepting, rejecting or the I do not know *state.*

Thus, the computation tree of a probabilistic Turing machine on any given input is a complete binary tree. In particular, all paths of the computation tree leading from the root to a leaf have the same probability to be a computation on the given input.

Definition 2.3.2 *A probabilistic Turing machine M is a* **Las Vegas Turing machine** *if*

(i) for every $x \in L$ the computation tree of M for x contains only accepting and I do not know *states and at least half of the leaves contain accepting states, and*

(ii) for every $x \notin L$ the computation tree of M for x contains only rejecting and I do not know *states and at least half of the leaves contain rejecting states.*

A **one-sided error Monte Carlo Turing machine** *M is a probabilistic Turing machine such that*

(i) for every $x \in L$ at least half of the leaves of the computation tree of M for x contain accepting states and

(ii) for every $x \notin L$ all leaves of the computation tree of M on x contain rejecting states.

A **two-sided error Monte Carlo Turing machine** *M is a probabilistic Turing machine such that*

there exists $\epsilon > 0$ such that for every $x \in L$ ($x \notin L$), at least $\frac{1}{2} + \epsilon$ of all leaves contain accepting (rejecting) states (i.e. for any input the probability to give the right answer is at least $\frac{1}{2} + \epsilon$).

In what follows we consider the following complexity classes:

- $ZPP = \{L \mid L = L(M)$ for a polynomial-time Las Vegas Turing machine $M\}$

- $R = \{L \mid L = L(M)$ for a polynomial-time one-sided error Monte Carlo Turing machine $M\}$ and
- $BPP = \{L \mid L = L(M)$ for a polynomial-time two-sided error Monte Carlo Turing machine $M\}$.

For any function $f : N \to N$, we define the class $\beta_f = \{L \mid L = L(M)$ for a polynomial-time nondeterministic Turing machine using at most $O(f(n))$ nondeterministic guesses of inputs of length $n\}$.

For any language A and any complexity class X, X^A is the complexity class X relativized by the oracle A. For any language class X, $co\text{–}X = \{L \mid L^C \in X\}$. The following inclusion between complexity classes are well-known:

$$P \subseteq ZPP = R \cap co - R \subseteq R \subseteq NP.$$

Diaz and Torán [6] have asked for the relation between β_f and the classes between P and NP in the relativized world, i.e., whether limited nondeterminism can be more powerful than Las Vegas (resp. Monte Carlo) or whether Las Vegas (resp. Monte Carlo) can be more powerful than limited nondeterminism. The answer for this question is given in the following two theorems.

Theorem 2.3.3 *For every function $f : N \to N$ bounded by a polynomial, there exists a language L and an oracle B such that*

$$L \in ZPP^B - {\beta_f}^B.$$

Corollary 2.3.4 *For every function $f : N \to N$ bounded by a polynomial, there exists a language L and an oracle B such that*

$$L \in BPP^B - {\beta_f}^B \text{ and } L \in R^B - {\beta_f}^B.$$

Theorem 2.3.3 shows the existence of a relativization in which any polynomial restriction on the number of nondeterministic guesses strongly decreases the power of polynomial-time nondeterministic Turing machines (even Las Vegas may be more powerful than limited nondeterminism).

Theorem 2.3.5 *Let $f : N \to N$ be a polynomial-time constructible function bounded by a polynomial such that the function $2^{f(n)}$ majorizes each polynomial for almost all n. Then there exist a language L and an oracle B such that*

$$L \in {\beta_f}^B - BPP^B.$$

Corollary 2.3.6 *For every f fulfilling the assumptions of Theorem 2.3.5, there exist languages L and B such that*

$$L \in {\beta_f}^B - R^B \text{ and } L \in {\beta_f}^B - ZPP^B.$$

Thus, there exists a relativization for which a relatively small (but superlogarithmic) number of nondeterministic guesses gives more power than Monte Carlo (resp. Las Vegas) computations with any number of random bits.

3 Proofs

3.1 The Proof of Theorem 2.1.1

First, we give an informal idea of the proof. Let $f : \{0,1\}^{2n} \to \{0,1\}$ be a Boolean function. We represent f by a $2^n \times 2^n$ Boolean matrix $M(f) = [a_{u,v}]_{u,v \in \{0,1\}^n}$ with $a_{u,v} = f(uv)$. Then the number of different messages of an optimal one-way protocol P computing f is exactly the same as the number $\mathrm{r}(M(f))$ of different rows of $M(f)$, i.e. $\mathrm{cc}_1(f) = \lceil \log_2(r(M(f))) \rceil$ [1].

Any one-way Las Vegas protocol P' may be considered as a collection of (say) m deterministic one-way protocols $P_1, \ldots, P_m$ with probabilities $p_1, \ldots p_m$[1]. For any input α, P_i may compute the results 0, 1 or 2 (i.e., I do not know). Since P' is a Las Vegas protocol, no protocol P_i ever errs and for every $(u,v) \in \{0,1\}^n \times \{0,1\}^n$, the protocols $P_1, \ldots, P_m$ produce the output 2 with probability at most $\frac{1}{2}$. To any protocol P_i $(i = 1, 2, \ldots, m)$, one can assign its 0/1/2 communication

[1]This follows, since we consider Las Vegas protocols with a common random source.

matrix $M(P_i) = [b^i_{uv}]_{u,v \in \{0,1\}^n}$, where $b^i_{uv} = a_{uv}$ if P_i does not give output 2 and $b^i_{uv} = 2$ otherwise.

Our goal is to find one protocol P_i such that $M(P_i)$ has at least $\sqrt{r(M(f))}$ different rows. In order to reduce the number of different rows a deterministic protocol will smartly replace certain entries of $M(f)$ by a 2. Obviously, "twoing" certain entries of $M(f)$ will help reduce the number of different rows far more than twoing other entries: For the identity matrix the diagonal entries play this helper role. For instance we can reduce the number of different rows to two by setting the upper left and the lower right quarter to 2. Observe that this radical reduction in the number of different rows is obtained after twoing only one half of the entries! On the other hand, any significant reduction in the number of different rows has to involve the diagonal entries and any such entry has to stay untouched with probability at least one half. Hence one deterministic protocol exists with at least $N/2$ different rows (if we consider the $N \times N$ identity matrix).

In the above example the diagonal entries form a fooling set and any Las Vegas computation has to send at least $\frac{|F|}{2}$ messages for a fooling set F. However we cannot expect to find large fooling sets in general. In particular, the $n \times \log_2 n$ communication matrix M^*, whose ith row contains the binary representation of i, possesses only fooling sets of logarithmic size, but it can be shown that any Las Vegas one-way protocol has to exchange $\sqrt{n}$ mesages.

Our proof will introduce a new notion of fooling sets. Set $M(f)$=M and assume that M has r pairwise different rows and c pairwise different columns. Our new notion of fooling sets is based on a real-valued weight assignment

$$\text{weight} : \{1, \ldots, r\} \times \{1, \ldots, c\} \to R$$

for M. Let $I = \{1, \ldots, r\}$. We define the function weight iteratively, processing column after column. We begin with column 1.

Case 1: Column 1 is monochromatic for all rows in I. Then set

$$\text{weight}(i, 1) = 0$$

for all rows i.

Case 2: Column 1 is not monochromatic for the rows in I. In particular assume that $c \cdot |I|$ rows have a 0 in column 1 (and $(1-c) \cdot |I|$ rows have a 1 in column 1). We set

$$\text{weight}(i,1) = \begin{cases} \log_2(\frac{1}{c}) & \text{if } M[i,1] = 0 \\ \log_2(\frac{1}{1-c}) & \text{otherwise.} \end{cases}$$

We repeat this procedure for column 2, but now with the subsets $I_0 = \{i \in I \mid M[i,1] = 0\}$ and $I_1 = \{i \in I \mid M[i,1] = 1\}$ replacing the set I. The procedure stops if the row sets are singletons (since then all columns will be monochromatic).

We begin our analysis with the following technical fact.

Fact 3.1.1 *For any $x, y \geq 0$ and $c \in (0,1)$,*

$$x \cdot \log_2 \frac{x}{c} + y \cdot \log_2 \frac{y}{1-c} \geq (x+y) \cdot \log_2(x+y).$$

Proof of the Fact 3.1.1 We start by equivalently transforming the claimed inequality. The first transformation

$$\begin{aligned} & x \cdot \log_2 \frac{x}{c} + y \cdot \log_2 \frac{y}{1-c} \geq (x+y) \cdot \log_2(x+y) \Leftrightarrow \\ & \frac{x}{x+y} \cdot \log_2 \frac{x}{x+y}\frac{x+y}{c} + \frac{y}{x+y} \cdot \log_2 \frac{y}{x+y}\frac{x+y}{1-c} \geq \log_2(x+y) \end{aligned}$$

is trivial. Moreover observe that $\frac{x}{x+y} \cdot \log_2 \frac{x}{x+y}\frac{x+y}{c} = \frac{x}{x+y} \cdot \log_2 \frac{x}{x+y} \cdot \frac{1}{c} + \frac{x}{x+y} \log_2(x+y)$. Thus the term $\log_2(x+y)$ on the right hand side can be cancelled, since we also obtain the term $\frac{y}{x+y} \log_2(x+y)$ on the left hand side. And we get the equivalent inequality

$$\frac{x}{x+y} \cdot \log_2 \frac{x}{x+y}\frac{1}{c} + \frac{y}{x+y} \cdot \log_2 \frac{y}{x+y}\frac{1}{1-c} \geq 0.$$

But this inequality is correct, since the *informational divergence*

$$\sum_{i=1}^{n} p_i \cdot \log_2 \frac{p_i}{q_i}$$

(for any two probability distributions p and q) is always nonnegative [4]. 2

For a subset $R \subseteq \{1, \ldots, r\}$ set

$$\text{differ}(R) = \{j \mid \exists i_1, i_2 \in R : M[i_1, j] \neq M[i_2, j]\}.$$

Now, we are ready to analyse the properties of our weight assignment.

Lemma 3.1.2 **(a)** *For each* $(i, j) \in \{1, \ldots, r\} \times \{1, \ldots, c\}$,

$$\text{weight}(i, j) \geq 0.$$

(b) *For each* i *(*$1 \leq i \leq r$*)*

$$\sum_{j=1}^{c} \text{weight}(i, j) = \log_2 r.$$

(c) For any $R \subseteq \{1, \ldots, r\}$,

$$\sum_{j \in \text{differ}(R)} \sum_{i \in R} \text{weight}(i, j) \geq |R| \cdot \log_2 |R|.$$

Proof of Lemma 3.1.2 **(a)** is immediate by construction. We verify **(b)** by induction on r. The basis for $r = 1$ is trivial. For the inductive step we can assume without loss of generality that column 1 is not monochromatic. Let I_0 (resp. I_1) be the set of those rows with a zero (resp. one) in column 1 and assume that $|I_0| = c \cdot r$.

We apply the induction hypothesis to the rows in I_0 and I_1. For a row $i \in I_0$ we obtain

$$\sum_{j=2}^{c} \text{weight}(i, j) = \log_2(c \cdot r).$$

But $\text{weight}(i, 1) = \log_2(\frac{1}{c})$ and

$$\sum_{j=1}^{c} \text{weight}(i, j) = \log_2(\frac{1}{c}) + \log_2(c \cdot r) = \log_2 r.$$

The claim follows with a symmetric argument for the rows in I_1.

We apply induction on the size of R to verify part **(c)**. The basis for $|R| = 1$ is again trivial. We assume for the inductive step that column 1 is not monochromatic for the rows in R. Hence R splits into the subsets R_0 and R_1 of those rows

in R with value zero (resp. one) in column 1. Since we can apply the induction hypothesis to R_0 and R_1, we obtain

$$\begin{aligned} & \sum_{j \in \text{differ}(R)} \sum_{i \in R} \text{weight}(i,j) \\ = & \sum_{i \in R} \text{weight}(i,1) + \sum_{j \in \text{differ}(R), j \neq 1} \sum_{i \in R} \text{weight}(i,j) \\ \geq & |R_0| \cdot \log_2(\frac{1}{c}) + |R_1| \cdot \log_2(\frac{1}{1-c}) + |R_0| \cdot \log_2(|R_0|) + |R_1| \cdot \log_2(|R_1|) \end{aligned}$$

Thus the claim follows from Fact 3.1.1. 2

Assume we have a one-way Las Vegas protocol P' for a Boolean function f represented by the matrix $M(f)$ with r pairwise different rows. Let the function weight be defined for $M(f)$ with the above three properties. Then we obtain a deterministic one-way protocol $P \in \{P_1, P_2, \ldots, P_m\}$ such that

(*) the sum of all weights of entries of $M(P)$ with value 2 is at most one half of the sum of all weights (i.e., at most $\frac{1}{2} \cdot \sum_{i=1}^{r} \sum_{j=1}^{c} \text{weight}(i,j) = \frac{r}{2} \cdot \log_2 r$ with property (b) of the weight assignment).

This follows, since for every input the output of P' is equal to 2 with probability at most one half. The deterministic protocol partitions the set of all rows of $M(f)$ into classes $R_1, \ldots, R_k$ of identical rows (after twoing). By property (c) of the function weight we obtain for any class R_s

$$\sum_{j \in \text{differ}(R_s)} \sum_{i \in R_s} \text{weight}(i,j) \geq |R_s| \cdot \log_2 |R_s|.$$

The quantity on the left hand side is a lower bound for the weight of all entries of $M(R_s)$ with value 2. Moreover observe that, for $x = \sum_{i=1}^{k} x_i$,

$$\sum_{i=1}^{k} x_i \log_2 x_i \geq x \log_2 x - x \log_2 k.$$

(The convex function $y \log_2 y$ is minimized for $x_1 = \cdots = x_k = \frac{x}{k}$, assuming $x = \sum_{i=1}^{k} x_i$.) And hence the sum of weights of entries of $M(P)$ with value 2 is

at least

$$\sum_{i=1}^{k} |R_i| \log_2 |R_i| \geq r \log_2 r - r \cdot log_2 k.$$

But with (*)

$$\frac{r \log_2 r}{2} \geq \sum_{i=1}^{k} |R_i| \log_2 |R_i| \geq r \log_2 r - r \cdot log_2 k$$

and hence $k \geq \sqrt{r}$. In other words, $M(P)$ has at least $\sqrt{r}$ different rows (resp. the deterministic protocol P has to consist of at least $\sqrt{r}$ messages).

3.2 Proof of Theorem 2.2.1

The idea of the proof is to find a strong connection between the number of messages of one-way protocols and the number of states of finite automata in such a way that Theorem 2.1.1 can be applied. This kind of connection for one-way protocols and finite automata has been observed already in [8] in the form

$$\mathrm{cc}_1(h_n(L)) \leq \lceil \log_2(\mathrm{s}(L)) \rceil$$

for every regular language $L \subseteq \{0,1\}^*$ and every $n \in N$. Obviously this relation holds in the nondeterministic and randomized cases too. More precisely, the number of different messages of the best one-way protocol is a lower bound on the number of states of finite automata. Unfortunaly the difference between these two complexity measure may be arbitrarily large because communication complexity is a non-uniform computing model, whereas automata form a uniform computing model: Consider, for instance unary languages L where we always have $\mathrm{cc}_1(h_n(L)) \leq 1$. To overcome this difficulty we introduce one-way uniform protocols:

Definition 3.2.1 *Let Σ be an alphabet and let $L \subseteq \Sigma^*$. A* **one-way uniform protocol over Σ** *is a pair $D = \langle \Phi, \varphi \rangle$, where:*

(i) $\Phi : \Sigma^* \to \{0,1\}^*$ *is a function with the prefix freeness property, and*

(ii) $\varphi^* : \{0,1\}^* \times \{0,1\}^* \rightarrow \{accept, reject\}$ *is a function.*

We say that $\boldsymbol{D = \langle \Phi, \varphi \rangle}$ **accepts** $\boldsymbol{L}$, $\boldsymbol{L(D) = L}$, *if for all* $x, y \in \Sigma^*$:

$$\varphi(y, \Phi(x)) = accept \Longleftrightarrow xy \in L.$$

The **message complexity of the protocol** $\boldsymbol{D}$ *is*

$$\mathbf{mc}(\boldsymbol{D}) = |\{\Phi(x) \mid x \in \Sigma^*\}|$$

and we define the **message complexity of** $\boldsymbol{L}$ *as*

$$\mathbf{mc}(\boldsymbol{L}) = \min\{\mathrm{mc}(D) \mid D \text{ is a one-way uniform protocol accepting } L\}.$$

Finally we introduce communication matrices:

Definition 3.2.2 *Let* Σ *be an alphabet and let* $L \subseteq \Sigma^*$. *We define the infinite Boolean matrix* $M_L = (a_{uv})_{u,v \in \Sigma^*}$ *so that*

$$a_{uv} = 1 \Longleftrightarrow uv \in L.$$

Let $\mathbf{row_L}$ *be the number of different rows of* M_L.

Now, we can formulate the crucial observation.

Lemma 3.2.3 *For every regular language* L *over an alphabet* Σ,

$$\mathrm{s}(L) = \mathrm{mc}(L) = \mathrm{row}_L.$$

Proof: The equality $\mathrm{s}(L) = \mathrm{row}_L$ is just the Myhill-Nerode theorem, because row_L is exactly the index of the right invariant relation on Σ^* according to L.

The fact that the number of different rows of a communication matrix is equal to the number of different messages used by the best one-way protocol computing this matrix is well-known (see [1, 9]), and thus $\mathrm{mc}(L) = \mathrm{row}_L$. 2

Now, we are ready to complete the proof of Theorem 2.2.1. Let L be a regular language over an alphabet Σ. Since row_L is finite, one can easily find a finite submatrix M of M_L with $\mathrm{row}_L = \mathrm{s}(L)$ different rows. In the proof of Theorem 2.1.1 we have shown that every Las Vegas one-way protocol computing M uses at least $\sqrt{\mathrm{row}_L} = \sqrt{\mathrm{s}(L)}$ different messages. Since M is a submatrix of M_L, computing M_L cannot be any easier. So, every Las Vegas one-way uniform protocol[1] D accepting L satisfies $\mathrm{mc}(D) \geq \sqrt{\mathrm{s}(L)}$. Since for every LVFA A with $\mathrm{s}(A)$ states one can easily construct an equivalent Las Vegas one-way uniform protocol using exactly $\mathrm{s}(A)$ different messages we obtain that $\mathrm{lvs}(L) \geq \sqrt{\mathrm{s}(L)}$.

3.3 Proofs of Theorems 2.3.3 and 2.3.5

Here we give full proofs of our two theorems about relativized Turing machine computations.

Proof of Theorem 2.3.3 Let $M_1, M_2, M_3, \ldots$ be a sequence of all oracle Turing machines such that the running time of machine M_i on each input of length n is bounded by $p_i(n) = n^i + i$, and M_i on each input of length n can make at most $if(n)$ binary nondeterministic choices. Let $f(n)$ be bounded by the polynomial $n^m + m$ for some $m > 0$. Choose a sufficiently large $k > 0$ so that

$$2^{ik} > \log i + i(2^{im} + m) + i^2 + 3 \qquad (1)$$

for each $i \geq 1$. Let $p(n) = n^k$.

To construct the desired language L and the oracle B, first we set a natural number n_i and define B_i for each $i \geq 0$. Let $B_0 = \emptyset$ and let $n_i = 2^i$ for each $i \geq 0$. Having defined B_{i-1}, we introduce B_i as follows. Let

$$U_i = \{0^{n_i}u | u \in \{0,1\}^{p(n_i)}, 0^{n_i}u \text{ is not queried}$$

[1] We omit the formal definition of Las Vegas uniform protocols here because it is a straightforward extention of Definition 3.2.1. One requires for every $x, y \in \Sigma^*$ that the membership of xy in L is decided with probability at least $1/2$.

by $M_j^{\bigcup_{l<j} B_l}$ on 0^{n_j} for each $j \leq i\}$,

and, let

$$V_i = \{0^{n_i}v | v \in \{0,1\}^{p(n_i)+1}, 0^{n_i}v \text{ is not queried by } M_j^{\bigcup_{l<j} B_l} \text{ on } 0^{n_j} \text{ for each } j \leq i\}.$$

If 0^{n_i} is accepted by $M_i^{\bigcup_{j<i} B_j}$ then set $B_i = V_i$ else set $B_i = U_i$.

Let $B = \bigcup_i B_i$ and let $L = \{0^{n_i} | i \geq 1, 0^{n_i} \text{ is not accepted by } M_i^B\}$. Hence, $L \notin \beta_f^B$.

Now, let us show that $L \in R^B \cap \text{co-}R^B$. To do so, we need the following claim and we also need to bound the cardinality of the U_i's and the V_i's.

Claim 1. *0^{n_i} is accepted by $M_i^{\bigcup_{j<i} B_j}$ if and only if 0^{n_i} is accepted by M_i^B for each $i \geq 1$.*

Proof: Since each B_l is either U_l or V_l, there is no computational path of $M_i^{\bigcup_{j<i} B_j}$ on 0^{n_i} with a query in B_l for $l \geq i$ (see the definition of U_l and V_l above). 2

Let X be any oracle. One can easily observe that the number of queries of M_j^X on input 0^n is at most $2^{jf(n)}p_j(n)$. Thus, we have by (1) for each $i \geq 1$ (recall $p(n) = n^k$, $n_j = 2^j$, $p_j(n) = n^j + j$ and $f(n) \leq n^m + m$),

$$\begin{aligned}
|U_i| &\geq 2^{p(n_i)} - \sum_{j=1}^{i} 2^{jf(n_j)} p_j(n_j) \\
&\geq 2^{2^{ik}} - \sum_{j=1}^{i} 2^{j(2^{jm}+m)}(2^{j^2} + j) \\
&\geq 2^{2^{ik}} - i2^{i(2^{im}+m)}(2^{i^2} + i) \\
&\geq 2^{2^{ik}} - 2^{\log i} 2^{i(2^{im}+m)} 2^{i^2+1} \\
&\geq 3.2^{2^{ik}-2} \\
&= (3/4)2^{p(n_i)}.
\end{aligned}$$

Similarly, one can show that $|V_i| \geq (3/4)2^{p(n_i)+1}$ for each $i \geq 1$.

Now we are ready to prove that $L \in R^B$. Let us consider a probabilistic oracle Turing machine M operating on input x as follows. If x is not of the form 0^{n_i}, then M rejects x. If $x = 0^{n_i}$ for some $i \geq 1$, then M guesses a string $y \in \{0,1\}^{p(n_i)}$ and asks the oracle whether $0^{n_i}y \in B$. M accepts x if and only if $0^{n_i}y \in B$. Now our goal is to show that M^B accepts inputs in L with probability at least $3/4$, and that there is no accepting path of M^B on inputs not in L. Choose an arbitrary input x. There are three cases to be considered.

Case 1. $x = 0^{n_i} \in L$. By the definition of L above, 0^{n_i} is not accepted by M_i^B. By Claim 1, 0^{n_i} is not accepted by $M_i^{\bigcup_{j<i} B_j}$. By the definition of B_i above, $B_i = U_i$. Each string $0^{n_i}y$ queried by M^B on input 0^{n_i} belongs to $U_i = B_i \subseteq B$ with probability at least $3/4$, since $|U_i| \geq (3/4)2^{p(n_i)}$ and y is arbitrarily chosen string in $\{0,1\}^{p(n_i)}$ (see above). Thus, each input 0^{n_i} is accepted by M^B with probability at least $3/4$.

Case 2. $x = 0^{n_i} \notin L$. One can prove as in Case 1, that $B_i = V_i$. Any string $0^{n_i}y$ queried by M^B on input x cannot belong to B_i $(= V_i)$, since the length of $0^{n_i}y$ is $n_i + p(n_i)$, but V_i contains only strings of length $n_i + p(n_i) + 1$ (see the definition of V_i above). Similarly, $0^{n_i}y$ cannot belong to any B_j with $j \neq i$, since the length of $0^{n_i}y$ is $n_i + p(n_i) = 2^i + 2^{ik}$, but each B_j with $j \neq i$ is either U_j or V_j, and hence it contains only strings of length either $n_j + p(n_j) = 2^j + 2^{jk}$ or $n_j + p(n_j) + 1 = 2^j + 2^{jk} + 1$. Thus, $0^{n_i}y$ cannot belong to $B = \bigcup_j B_j$. Hence, there is no accepting computational path of M^B on x.

Case 3. x is not of the form 0^{n_i}. In such a case, M^B rejects x (see above).

This completes the proof that $L \in R^B$.

The proof that $L \in$ co-R^B is similar and we omit details here. Note that instead of M one can use a machine M' that accepts any input not of the form 0^{n_i}, and M' guesses a string $y \in \{0,1\}^{p(n_i)+1}$ on input 0^{n_i} and accepts it if and only if $0^{n_i}y \in V_i$.

This completes the proof of the theorem. 2

Proof of Theorem 2.3.5 Let $d_1, d_2, d_3, \ldots$ be an enumeration of all possible triples (M, p, ϵ), where M is a probabilistic oracle Turing machine with polynomial time clock p and $0 < \epsilon < 1/2$ is a rational number. To construct the desired language L and the oracle B, first we define a natural number n_i and a set B_i for each integer $i \geq 0$.

Let $n_0 = 0$ and $B_0 = \emptyset$. Having defined n_{i-1} and B_{i-1}, we define n_i and B_i as follows. Let $d_i = (M_i, p_i, \epsilon_i)$. Choose a sufficiently large n_i so that

(i) $n_i > n_{i-1}$,

(ii) $n_i > \max_{j<i}\{p_j(n_j)\}$,

(iii) $(\epsilon_i/2)2^{f(n_i)} > p_i(n_i)$.

To construct B_i, there are two cases to be considered.

Case 1. The number of accepting computations of $M_i^{\bigcup_{j<i} B_j}$ on input 0_i^n is less than $2^{p_i(n_i)-1}$.

Claim 2. *There is a string $y \in \{0,1\}^{f(n_i)}$ such that the string $0^{n_i}y$ occurs (as a query) in at most $(\epsilon_i/2)2^{p_i(n_i)}$ computations of $M_i^{\bigcup_{j<i} B_j}$ on input 0^{n_i}.*

Proof: Recall that there are exactly $2^{p_i(n_i)}$ computations of $M_i^{\bigcup_{j<i} B_j}$ on 0^{n_i}. For each $l = 1, 2, \ldots, 2^{p_i(n_i)}$, let S_l be the set of all queries occuring in the l-th computation of $M_i^{\bigcup_{j<i} B_j}$ on 0^{n_i}. Clearly, $|S_l| \leq p_i(n_i)$ for each l. Hence,

$$\sum_{l=1}^{2^{p_i(n_i)}} |S_l| \leq 2^{p_i(n_i)} p_i(n_i). \qquad (2)$$

If each string of the form $0^{n_i}y$ with $y \in \{0,1\}^{f(n_i)}$ belongs to at least $(\epsilon_i/2)2^{p_i(n_i)}$ sets S_l's, then $\sum_{l=1}^{2^{p_i(n_i)}} |S_l| \geq (\epsilon_i/2)2^{p_i(n_i)}2^{f(n_i)} > 2^{p_i(n_i)}p_i(n_i)$, by (iii), but this contradicts (2). 2

Choose any y satisfying Claim 2 and set $B_i = \{0^{n_i}y\}$.

Case 2. The number of accepting computations of $M_i^{\bigcup_{j<i} B_j}$ on 0^{n_i} is at least $2^{p_i(n_i)-1}$. Set $B_i = \emptyset$.

Let $B = \bigcup_i B_i$ and let L be the set of all inputs 0^{n_i}, $i \geq 1$, such that the number of accepting computations of $M_i^{\bigcup_{j<i} B_j}$ on 0^{n_i} is less than $2^{p_i(n_i)-1}$.

L can be recognized by a machine M^B in polynomial time with $f(n)$ nondeterministic steps on inputs of length n as follows. M rejects each input not in 0^*. On input 0^n, M first guesses a string $y \in \{0,1\}^{f(n)}$ and then asks the oracle whether $0^n y \in B$. M^B accepts 0^n if and only if $0^n y \in B$.

To show that $L \notin BPP^B$, we proceed as follows. Let us choose any $i \geq 1$. Let $d_i = (M_i, p_i, \epsilon_i)$. Our goal is to prove that L cannot be recognized by M_i^B with error probability ϵ_i. Suppose that the input 0^{n_i} is accepted [rejected] by M_i^B with probability $1/2 + \epsilon_i$. Each computation of $M_i^{\bigcup_{j\leq i} B_j}$ on input 0^{n_i} remains unchanged after replacing the oracle $\bigcup_{j\leq i} B_j$ by the oracle B, since each query produced by $M_i^{\bigcup_{j\leq i} B_j}$ on input 0^{n_i} is of length at most $p_i(n_i) < n_l \leq f(n_l) + n_l$ for $l > i$ (see (ii) above), and hence it cannot belong to $\bigcup_{l>i} B_l$. Thus, the input 0^{n_i} is accepted [rejected] with probability $1/2+\epsilon_i$ by $M_i^{\bigcup_{j\leq i} B_j}$, too. By Definition 2.3.1, the number of accepting [rejecting] computations of $M_i^{\bigcup_{j\leq i} B_j}$ on 0^{n_i} is at least $(1/2+\ \epsilon_i)2^{p_i(n_i)}$. Since each computation of $M_i^{\bigcup_{j<i} B_j}$ on 0^{n_i}, that does not contain any query in B_i, remains unchanged after replacing the oracle $\bigcup_{j<i} B_j$ by the oracle $\bigcup_{j\leq i} B_j$, we have by Claim 2 and by the fact that $B_i = \{0^{n_i} y\}$ (if i meets the assumption of Case 1 above) and by the fact that $B_i = \emptyset$ (if i meets the assumption of Case 2 above), that the number of accepting [rejecting] computations of $M_i^{\bigcup_{j<i} B_j}$ on 0^{n_i} is at least $(1/2 + \epsilon_i)2^{p_i(n_i)} - (\epsilon_i/2)2^{p_i(n_i)} > 2^{p_i(n_i)-1}$. Thus, by the definition of L above, $0^{n_i} \notin L$ [$0^{n_i} \in L$]. But we have assumed above, that 0^{n_i} is accepted [rejected] by M_i^B with probability $1/2 + \epsilon_i$. Thus, L cannot be recognized by M_i^B with error probability ϵ. 2

References

[1] Aho, A.V., Hopcroft, J.E., Yannakakis, M.: On notions of information transfer in VLSI circuits.*In: Proc. 15th Annual ACM STOC, ACM* 1983, 133–139.

[2] Abelson, H.: Lower bounds on information transfer in distributed computations. *In: Proc. 29th Annual IEEE FOCS*, IEEE 1978, 151–158.

[3] Bovet,D.P., Crescenzi,P.: Introduction to the Theory of Complexity. Prentice Hall 1994.

[4] Csiszar, I., Körner, J.: Information theory: coding theorems for discrete memeoryless systems, *Academic Press*, 1986.

[5] Dietzfelbinger, M., Kutylowski, M., Reischuk, R.: Exact lower bounds for computing Boolean functions on CREW PRAMs. *J. Computer System Sciences* 48, 231-254.

[6] Diaz, J. Torán, J.: Classes of bounded nondeterminism. *Mathematical Systems Theory,* **23** (1990), 21-32.

[7] Hromkovič, J., Schnitger, G.: On the power of the number of advice bits in nondeterministic computations. *Proc. ACM STOC'96*,ACM 1996, pp. 551-560.

[8] Hromkovič, J., Relation between Chomsky hierarchy and communication complexity hierarchy. *Acta Math. Univ. Com.* 48-49 (1986), 311-317.

[9] Hromkovič, J.: Communication Complexity and Parallel Computing. Springer-Verlag 1997.

[10] Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press 1997.

[11] Meyer, A.R., Fischer, M.J.: Economies of description by automata, grammars and formal systems. *In: Proceedings 12th SWAT Symp.* 1971, 188–191

[12] Mehlhorn,K., Schmidt,E.: Las Vegas is better than determinism in VLSI and distributed computing. *Proc. 14th ACM STOC'82*, ACM 1982, pp. 330-337.

[13] Newman, I., Private us. Common random bits in communication complexity. *Information Processing Letters* 39 (1991), 301-315.

[14] Papadimitriou, Ch., Sipser, M.: Communication complexity. *J.Comput. Syst. Sci.* **28** (1984), 260–269.

[15] Yao, A.C.: Some complexity questions related to distributed computing. *In: Proc. 11th Annual ACM STOC, ACM* 1981, 308–311.