1. INTRODUCTION

The paper contains several results showing advantages of probabilistic automata over deterministic ones. The advantages always are of one kind, namely, the complexity of probabilistic automata turns out to be considerably smaller. The paper contains both new results and results already published in the USSR (though, hardly these results are known outside the USSR).

The economy of computer resources (computation time, memory, etc.) is studied in this paper, comparing computation by probabilistic and deterministic machines. Both from theoretical and applications aspect the most interesting part of the problem is the possibility of such an economy in computation of explicitly defined natural functions with probability as close to 1 as possible. Nevertheless until recently this part of the problem was the most hard and the least explored. The power of probabilistic machines was proved only in two cases: 1) when the probability of the correct result is high but the concept of computation is rather artificial, 2) for recognition of explicitly defined specific languages but for non-isolated cut-points, i.e. with probability exceeding 1/2 only by unboundedly small numbers.

Every theorem on power of probabilistic machines in this survey consists of two parts. The first part asserts that the language under consideration is recognizable by a probabilistic machine. The second part shows that no deterministic machine can do this. We organize the paper according to the methods used in the proofs of the first parts of theorems. The proofs of the second parts use various techniques with no unifying idea.
The types of automata and machines used in the paper are well-known in the literature. Precise definitions can be found in [Hen 77].

2. A LEMMA FOR JOINING COUNTERS

When designing efficient probabilistic algorithms for simple computational devices sometimes the algorithms can be presented as a procedure involving large number of counters, and the number of the counters depends on the input word. This section contains a lemma that can be used to justify the correctness of the probabilistic algorithm.

Let \( N \) denote the set of all non-negative integers, \( Z \) denote the set of all integers and \( X \) denote an arbitrary set.

Let \( n \in \mathbb{N} \). Let \( P \) be a function \( X \rightarrow \{0,1\} \) and \( F \) be a function \( X \times \{1,2,\ldots,n\} \rightarrow \mathbb{Z} \). We call the pair of functions \( <P,F> \) dispersive if for all \( x \in X \) the following holds:

1) \( P(x)=1 \rightarrow (\forall u,v \in \{1,2,\ldots,n\}) \ (F(x,u)=F(x,v)) \),
2) \( P(x)=0 \rightarrow (\forall u,v \in \{1,2,\ldots,n\}) \ (u \neq v \rightarrow (F(x,u) \neq F(x,v))) \).

Let \( n \in \mathbb{N} \), \( k(x) \) be a function \( X \rightarrow \mathbb{N} \), and for every \( i \in \{1,2,\ldots,k(x)\} \) a dispersive pair of functions \( \langle P_i : X \rightarrow \{0,1\} ; F_i : X \times \{1,2,\ldots,n\} \rightarrow \mathbb{Z} \rangle \). We denote the family \( \{F_1,F_2,\ldots,F_n\} \) by \( F \). We consider the following random value \( S_F(x) \). For arbitrary \( i \in \{1,2,\ldots,k(x)\} \) a random number \( y_i \) is taken which is distributed equiprobably in \( \{1,2,\ldots,n\} \) and every \( y_i \) is statistically independent from all the other \( y_j \). Then

\[
S_F(x) = \sum_{i=1}^{k(x)} F_i(x,y_i).
\]

**LEMMA 2.1.** For arbitrary \( x \in X \), if \( \sum_{i=1}^{k(x)} P_i(x)=1 \) then there is a \( z \in \mathbb{Z} \) such that \( S_F(x)=z \) with probability 1, and if \( \sum_{i=1}^{k(x)} P_i(x)=0 \) then there is not a single \( z \) such that the probability of \( S_F(x)=z \) would exceed \( 1/n \).

**PROOF.** The first assertion is evident. To prove the second assertion we assume that for some \( x \), \( i \in \{1,2,\ldots,k(x)\} \) it holds \( P_i(x)=0 \). Then the values \( \{F_i(x,1),F_i(x,2),\ldots,F_i(x,n)\} \) are
pairwise distinct. The total \( S_F(x) = \sum_{j=1}^{k(x)} F_j(x,y_j) \) includes \( F_i(x,y_i) \) and \( y_i \) is independent from all the rest of \( y_j \). Hence the total \( S_F(x) \) can equal \( z \) for no more than one of the \( n \) possible values \( y_i \).

3. THREE-HEAD PROBABILISTIC FINITE AUTOMATA
VERSUS MULTI-HEAD DETERMINISTIC ONES

Some languages can be recognized by multi-head finite automata. For these languages the minimum number of heads is a complexity measure. It was proved in FREIVALDS (1979) that for arbitrary \( k>0 \) there is a language which can be recognized by probabilistic \( 2 \)-head finite automaton with probability \( 1-\varepsilon \) for arbitrary \( \varepsilon>0 \) but which is not recognizable by any deterministic \( k \)-head finite automaton. In this Section another advantage of probabilistic multi-head finite automata is proved. A language is described which can be recognized by a probabilistic \( 3 \)-head finite automaton but no deterministic multi-head finite automaton can recognize this specific language.

On the other hand, I risk to conjecture that \( 2 \)-head of a probabilistic automaton do not suffice for such an advantage:

**CONJECTURE 1.** If a language is recognized by a probabilistic \( 2 \)-head finite automaton with probability \( 1-\varepsilon \) for arbitrary \( \varepsilon>0 \) then the language is recognized by a deterministic multi-head finite automaton as well.

We describe a language \( W \). To describe it we introduce the types of blocks \( A(b) \), \( B(m,k) \), \( C(m,k) \), \( D(n,i) \), \( E_b(i,k) \).

\[
A(b) = \begin{cases} 0^b & \text{if } b \leq 1 \\ 10^2 & \text{if } 2 \leq b \leq 3 \\ 10^{2-1} & \text{if } 3 \leq b \leq 2^b \\ 20 & \text{if } b > 2^b \end{cases}
\]

\[
B(m,k) = \begin{cases} 0^m & \text{if } m \leq k-1 \\ 10^m & \text{if } m \geq k-1 \end{cases}
\]

\[
C(m,k) = \begin{cases} 0^3 & \text{if } 3 \leq m \leq 3 \\ 3D(m,3) & \text{if } 3 \leq m \leq 3 \\ 3B(m,k-1)4B(m,k) & \text{if } m \geq 3 \end{cases}
\]

\[
D(n,i) = \begin{cases} 0^i & \text{if } i \leq 1 \\ 10^i & \text{if } i \leq 2 \\ 10^i & \text{if } i \geq 2 \end{cases}
\]

\[
E_b(i,k) = \begin{cases} C(1,k)4D(1^k,1)4C(2,k)4D(2^k,1) & \text{if } b = 1 \\ \cdots & \text{if } b = 2 \\ \cdots & \text{if } b > 2 \end{cases}
\]
\[ \ldots 4c(2^b-1,k)4D((2^b-1)^k,i)5c(2^b,k)5D((2^b)^k,i); \]
\[ \text{code}_b(i_1,i_2,\ldots,i_b,j_b,\ldots,j_2,j_1): A(b)7E_b(i_1,1)6E_b(i_2,2)6\ldots \]
\[ \ldots 6E_b(i_{b-1},b-1)7E_b(i_b,b)7E_b(j_b,b)6E_b(j_{b-1},b-1)6\ldots \]
\[ \ldots 6E_b(j_2,2)7E_b(j_1,1); \]

\[ W' = \{ x | x \in \{0,1,2,3,4,5,6,7\}^* \land (\exists b \geq 2, i_1, i_2, \ldots, i_b, j_b, \ldots, j_2, j_1) \]
\[ (x = \text{code}_b(i_1, i_2, \ldots, i_b, j_b, \ldots, j_2, j_1) \} \]

\[ W = \{ x | x \in \{0,1,2,3,4,5,6,7\}^* \land (\exists b \geq 2, i_1, i_2, \ldots, i_b) \]
\[ (x = \text{code}_b(i_1, i_2, \ldots, i_b, i_b, \ldots, i_2, i_1) \} \]

**Theorem 1.** (1) For arbitrary \( c > 0 \) there is a probabilistic 3-head finite automaton which recognizes the language \( W \) accepting every word in \( W \) with probability 1 and rejecting carry word in \( W \) with probability \( 1-c \). (2) No deterministic multi-head finite automaton can recognize \( W \).

**Proof.** (1) Let \( b_0 \) be the least value of \( b \) for which \( b/2^b < c \), and let \( d \) be a power of 2 for which \( d > 1/c \). The computation is started by the head \( h_1 \) going through the subword \( 0^b \) of the block \( A(b) \) and deciding whether or not \( b \geq b_0 \). Note that it suffices to read only the first \( b_0+1 \) letters of the subword. The processing of the input word is different for large and small values of \( b \).

If \( b \) is small, i.e. \( b/2^b = c \) then the input word \( x \in W \) can be represented as a concatenation of words \( z_1^0, z_2^0, z_3^0, z_4^0 \ldots z_1^t \), where \( t \) is a constant, \( z_1, z_2, \ldots, z_t \) are fixed words in \( \{0,1,2,3,4,5,6,7\} \) and these words begin with letters differing from 0. Additionally, there is a binary relation \( R_b \) completely defined by the parameter \( b \) such that \((i,j) \in R_b \Rightarrow 1_i = 1_j \).

Two heads \( h_1 \) and \( h_2 \) suffice to process the word for small values of \( b \). The leading head \( h_1 \) does all the reading from the tape. The head \( h_2 \) reads nothing. Its position on the tape is used to simulate a counter. The content of the counter equals the distance between the heads multiplied to a constant factor plus residue modulo this factor kept in the internal memory of the automaton.

The correctness of \( z_1, z_2, \ldots, z_t \) is checked by the finite
automaton while reading the input. The checking of all the equalities $l_i = l_j$ where it is needed is done in a probabilistic way by using the counter. We define the following family of dispersive pairs of functions.

$$P_{ij}(x) = \begin{cases} 1, & \text{if } l_i = l_j \\ 0, & \text{if } l_i \neq l_j \end{cases}$$

$$F_{ij}(x, y) = y(l_i - l_j).$$

The pair $(P_{ij}, F_{ij})$ is in the family iff $(i, j) \in \mathcal{R}_b$. Let $\mathcal{F} = \{F_{ij}\}$. The probabilistic automaton for every $(i, j) \in \mathcal{R}_b$ produces a random number $y_{ij} \in \{1, 2, \ldots, d\}$ being independent of all the other random numbers and adds $F_{ij}(x, y_{ij})$ to the counter. The input word is accepted iff at the end the counter is empty. The correctness of the algorithm follows from Lemma 1.

If $b$ is large, i.e. $b/2^b < c$ then the processing of the input word is more complicated. The processing consists of 3 separate actions:

1) testing whether $x \in \mathcal{W}'$,
2) choice of a random number $a$ ($1 \leq a \leq 2^b$),
3) testing whether $a_1^1 + a_2^1 + \ldots + a^b_1 + a_1^2 + a_2^2 + \ldots + a^b_2$. The leading head $h_1$ reads the input word. The head $h_2$ simulates the counter. This counter is used to perform the actions 1) and 3). The head $h_3$ is used to perform the action 2).

To perform the action 1), first, correctness of the structure of the input word is tested (the structure can be described by means of a special regular language $W'$ such that $W' \subseteq W$), and, second, $S_F(x)$ is accumulated in the counter where $F$ is the following family of functions.

$F_1(x, y) = y(x_1^1 - x_2^1)$ where $x_1^1$ is the number of zeros in the subword $0^b$ in $A(b)$, and $x_2^1$ is the number of subwords consisting of zeros between the symbols 2 in the block $A(b)$.

$F_2(x, y) = y(x_1^2 - x_2^2)$ where $x_1^2$ is the number of zeros in the first subword $0^b$ consisting of zeros after the second symbol 2 in the block $A(b)$, and $x_2^2$ is the number of blocks $C(u, 1)$ in the block $E_b(i_1, 1)$.

$F_{3, u}(x, y) = y(2x_1^3 - x_2^3)$, $(1 \leq u \leq b - 1)$ where $x_1^3$ is the number of zeros in the $u$-th subword $0^b$ of zeros after the first
symbol 2 in the block $A(b)$, and $x_2^{u+1}$ is the number of zeros in the $(u+1)$-th subword of zeros after the first symbol 2 in the block $A(b)$.

$F_{y,u,v,w,z}(x,y) = y(x_1^{y,u,v,w,z} - x_2^{y,u,v,w,z+2})$, (1≤z≤2, 1≤w≤b, 1≤v≤w, 1≤u≤2b) where $x_1^{y,u,v,w,z}$ is the number of zeros in the first substring $0^u$ of zeros in the block $B(u,v)$ in $C(u,w)$ either in $E_b(i_w,w)$, if $z=1$, or in $E_b(j_w,w)$, if $z=2$, and $x_2^{y,u,v,w,z}$ is the number of subwords of zeros in the same block $B(u,v)$ in $C(u,w)$ in $E_b(i_w,w)$, if $z=1$, or in $E_b(j_w,w)$, if $z=2$.

$F_{5,u,v,w,z}(x,y) = y(x_1^{5,u,v,w,z,t} - x_2^{5,u,v,w,z,t})$, (1≤z≤2, 1≤w≤b, 1≤v≤w, 2≤t≤u) where $x_1^{5,u,v,w,z,t}$ is the number of zeros in the $t$-th subword of zeros in the block $D(u,v)$ in $C(u,w)$ either in $E_b(i_w,w)$, if $z=1$, or in $E_b(j_w,w)$, if $z=2$, and $x_2^{5,u,v,w,z,t}$ is the number of zeros in the $(t+1)$-th subword of zeros in the block $B(u,v)$ in $C(u,w)$ either in $E_b(i_w,w)$, if $z=1$, or in $E_b(j_w,w)$, if $z=2$.

$F_{6,u,v,w,z}(x,y) = y(x_1^{6,u,v,w,z} - x_2^{6,u,v,w,z})$, (1≤z≤2, 1≤w≤b, 1≤v≤w, 2≤u≤2b) where $x_1^{6,u,v,w,z}$ is the number of zeros between the first and the third symbol 2 in the block $B(u,v)$ in $C(u,w)$ either in $E_b(i_w,w)$, if $z=1$, or in $E_b(j_w,w)$, if $z=2$, and $x_2^{6,u,v,w,z}$ is the number of zeros in the first subword of zeros after the third symbol 2 in the block $B(u,v)$ in $C(u,w)$ either in $E_b(i_w,w)$, if $z=1$, or in $E_b(j_w,w)$, if $z=2$.

$F_{7,u,v,w,z}(x,y) = y(x_1^{7,u,v,w,z} - x_2^{7,u,v,w,z})$, (1≤z≤2, 1≤w≤b, 1≤v≤w-1, 1≤u≤2b) where $x_1^{7,u,v,w,z}$ is the number of zeros after the third symbol 2 subword of zeros in the block $B(u,v)$ in $C(u,w)$ either in $E_b(i_w,w)$, if $z=1$, or in $E_b(j_w,w)$, if $z=2$, and $x_2^{7,u,v,w,z}$ is the number of zeros in the second subword of zeros in the block $B(u,v+1)$ in $C(u,w)$ either in $E_b(i_w,w)$, if $z=1$, or in $E_b(j_w,w)$, if $z=2$.

$F_{8,u,v,w,z}(x,y) = y(x_1^{8,u,v,w,z} - x_2^{8,u,v,w,z})$, (1≤z≤2, 1≤w≤b, 1≤v≤w-1, 1≤u≤2b) where $x_1^{8,u,v,w,z}$ is the number of zeros in the first subword of zeros in the block $B(u,v)$ in $C(u,w)$ either in $E_b(i_w,w)$, if $z=1$, or in $E_b(j_w,w)$, if $z=2$, and $x_2^{8,u,v,w,z}$ is the number of zeros in the first subword of zeros in the block
$B(u,v+1)$ in $C(u,w)$ either in $E_b(i_w,w)$, if $z=1$, or $E_b(j_w,w)$, if $z=2$.

$F_{9,u,w,z}(x,y) = y(x_1^{9,u,w,z} - x_2^{9,u,w,z})$, $(1 \leq z \leq 2, 1 \leq w \leq b, 1 \leq u \leq 2^b)$

where $x_1^{9,u,w,z}$ is the number of zeros in the last subword of zeros in the block $B(u,w)$ in $C(u,w)$ either in $E_b(i_w,w)$, if $z=1$, or in $E_b(j_w,w)$, if $z=2$, and $x_2^{9,u,w,z}$ is the number of zeros in the first subword of zeros either in the block $D(u^w,i_w)$ in $E_b(i_w,w)$, if $z=1$, or in the block $D(u^w,j_w)$ in $E_b(j_w,w)$, if $z=2$.

$F_{10,u,w,z,t}(x,y) = y(x_1^{10,u,w,z,t} - x_2^{10,u,w,z,t})$, $(1 \leq z \leq 2, 1 \leq w \leq b, 1 \leq u \leq 2^b - 1)$ where $x_1^{10,u,w,z,t}$ is the number of zeros in the $t$-th subword of zeros either in the block $D(u^w,i_w)$ in $E_b(i_w,w)$, if $z=1$, or in $D(u^w,j_w)$ in $E_b(j_w,w)$, if $z=2$, and $x_2^{10,u,w,z,t}$ is the number of zeros in the $(t+1)$-th subword of zeros either in the block $D(u^w,i_w)$ in $E_b(i_w,w)$, if $z=1$, or in the block $D(u^w,j_w)$ in $E_b(j_w,w)$, if $z=2$.

$F_{11,u,w,z}(x,y) = y(x_1^{11,u,w,z} - x_2^{11,u,w,z})$, $(1 \leq z \leq 2, 1 \leq w \leq b, 1 \leq u \leq 2^b - 1)$ where $x_1^{11,u,w,z}$ is the number of zeros in the first subword of zeros either in the block $D(u^w,i_w)$ in $E_b(i_w,w)$, if $z=1$, or in the block $D(u^w,j_w)$ in $E_b(j_w,w)$, if $z=2$, and $x_2^{11,u,w,z}$ is the number of zeros in the first subword of zeros either in the block $D((u+1)^w,i_w)$ in $E_b(i_w,w)$, if $z=1$, or in the block $D((u+1)^w,j_w)$ in $E_b(j_w,w)$, if $z=2$.

$F_{12,u,w,z}(x,y) = y(x_1^{12,u,w,z} - x_2^{12,u,w,z+1})$, $(1 \leq z \leq 2, 1 \leq w \leq b, 1 \leq u \leq 2^b - 1)$ where $x_1^{12,u,w,z}$ is the number of zeros in the second subword of zeros in the block $C(u,w)$ either in $E_b(i_w,w)$, if $z=1$, or in $E_b(j_w,w)$, if $z=2$, and $x_2^{12,u,w,z}$ is the number of zeros in the second subword of zeros in the block $C((u+1),w)$ either in $E_b(i_w,w)$, if $z=1$, or in $E_b(j_w,w)$, if $z=2$.

$F_{13,u,w,z}(x,y) = y(x_1^{13,u,w,z} - x_2^{13,u,w,z})$, $(1 \leq z \leq 2, 1 \leq w \leq b, 1 \leq u \leq 2^b - 1)$ where $x_1^{13,u,w,z}$ is the number of zeros in the first subword of zeros in the block $C(u,w)$ either in $E_b(i_w,w)$, if $z=1$, or in $E_b(j_w,w)$, if $z=2$, and $x_2^{13,u,w,z}$ is the number of zeros in the first subword of zeros in the block $C((u+1),w)$ either in $E_b(i_w,w)$, if $z=1$, or in $E_b(j_w,w)$, if $z=2$. 
\[ F_{14,w,z}(x,y) = y(x_{14,w,z} - x_{214,w,z}), \quad (1 \leq z \leq 2, \ 1 \leq w \leq b - 1) \]

\( x_{14,w,z} \) is the number of zeros in the first subword of zeros either in \( E_b(i_{w},w) \), if \( z = 1 \), or in \( E_b(j_{w},w) \), if \( z = 2 \), and \( x_{214,w,z} \) is the number of zeros in the first subword of zeros either in \( E_b(i_{w+1},w+1) \), if \( z = 1 \), or in \( E_b(j_{w+1},w+1) \), if \( z = 2 \).

\[ F_{15}(x,y) = y(x_{15} - x_{215}), \]

where \( x_{15} \) is the number of zeros in the first subword of zeros in the input word, and \( x_{215} \) is the number of blocks \( E_b(i_{w},w) \), in the input word up to the third symbol 7.

\[ F_{16}(x,y) = y(x_{16} - x_{216}), \]

where \( x_{16} \) is the number of zeros in the first subword of zeros in the input word, and \( x_{216} \) is the number of blocks \( E_b(j_{w},w) \), in the input word after the third symbol 7.

\( P_1(x), P_2(x), P_3(x), \ldots, P_{16}(x) \) equal 1 if \( F_1(x,y), F_2(x), F_3(x), \ldots, F_{16}(x) \), respectively, are equal 0, and they equal 0, otherwise.

We denote the family of functions \( \{F_1, F_2, \ldots, F_{16}\} \) by \( F \).

The head \( h_2 \) simulates a counter accumulating the value \( S_F(x) \). The machine starts with an empty counter. Hence, the head \( h_2 \) coincides with \( h_1 \). To start the computation of an arbitrary function \( F_\delta \in F \), a random number \( y_\delta \in \{1, 2, \ldots, d\} \) is chosen. Suppose, \( F_\delta(x,y) = y(x_1^{\delta} - x_2^{\delta}) \) where \( x_1^{\delta} \) is the number of symbols from a set \( M_1 \), and \( x_2^{\delta} \) is the number of symbols from a set \( M_2 \). Then in response to reading a symbol from \( M_1 \) by the head \( h_1 \), the content of the counter is increased by \( y_\delta \). In response to reading a symbol from \( M_2 \), the content of the counter is decreased by \( y_\delta \). While simulating the counter by the head \( h_2 \), the following relation between the content of the counter and the distance between the heads \( h_1 \) and \( h_2 \) is kept: the content equals 54 \( d \) times the distance plus the residue modulo 54 \( d \) (which is stored in the internal memory of the automaton).

The number of pairs \( \langle P_i, F_\perp \rangle \) can be unboundedly large but the finite memory of the automaton does not prevent it from the counting up the total \( S_F(x) \) since the automaton never needs to compute simultaneously more than two functions from any subfamily \( \{F_3, u(x,y)\}, \quad \{F_5, u, v, w, z, t(x,y)\}, \quad \{F_7, u, v, w, z(x,y)\}, \quad \{F_8, u, v, w, z(x,y)\}, \quad \{F_{10}, u, w, z, t(x,y)\}, \quad \{F_{11}, u, w, z(x,y)\}, \quad \{F_{12}, u, w, z(x,y)\}, \quad \{F_{13}, u, w, z(x,y)\}, \quad \{F_{14}, w, z(x,y)\} \), and it never
needs more than one function from any subfamily \( \{F_4,u,v,w,z(x,y)\} \), \( \{F_6,u,v,w,z(x,y)\} \), \( \{F_9,u,w,z(x,y)\} \). Hence at no moment the automaton keeps more than 26 random numbers \( y_\delta \in \{1,2,\ldots,d\} \). For the functions from \( \{F_3,u(x,y)\} \) sometimes the automaton add 2 \( y_\delta \) to the content of the counter. We see that the content is never changed by a number exceeding 27d.

To perform the action 2) the head \( h_3 \) starts with finding the block \( B(i_1,l) \). While the head \( h_1 \) goes through the block \( A(b) \), at moments when \( h_1 \) reads the first symbol 2 and all the symbol 1 (i.e. before every subword \( 0^2,0^2,\ldots,0^{2^{b-1}} \)) the automaton uses the random number generator and it produces a random bit \( r \). If this bit corresponding to the subword \( 0^{2^i} \) equals 1 then the head \( h_3 \) is moved \( 2^i \) blocks \( D(j,i_1) \) ahead (one block \( D(j,i_1) \) per one zero in \( 0^{2^i} \)). If the bit \( r \) equals 0 then the head \( h_3 \) is not moved. Thus the generation of all these \( r \)'s ends in moving \( h_3 \) to a random block \( D(a,i_1) \) where the values \( a \in \{1,2,3,\ldots,2^b\} \) are equiprobable.

To perform the action 3) the numbers \( i_1,i_2,\ldots,i_b,j_b,\ldots,j_2,j_1 \) are read by the head \( h_3 \). The block \( D(a,i_1) \) contains \( a^1_i \) zeros. The number \( a^1_i \) is added to the counter. Next, while \( h_1 \) traverses the block \( E_b(i_2,2) \) and hence meets \( 2^b \) blocks \( D(1^2,i_2), D(2^2,i_2),\ldots,D((2^b)^2,i_2) \), the head \( h_3 \) goes through \( 2^b \) blocks of type \( D \), reaches the block \( D(a^2,i_1) \) and adds the number of zeros in \( D(a^2,i_1) \) (being equal \( a^2_i \)) to the content of the counter. Then, while \( h_1 \) traverses the block \( E_b(i_3,3) \), thr head \( h_3 \) goes through \( 2^b \) blocks of type \( D \), reaches the block \( D(a^3,i_3) \) and adds \( a^3_i \) to the content of the counter, etc. After reaching the block \( E_b(j_b,b) \) by \( h_1 \), the head \( h_3 \) goes to the block \( D(a^b,b) \) and from this moment on the numbers \( a^b_i, j_b, a^{b-1}_b, j_{b-1}, \ldots, a^1_i \) are subtracted (not added). In total, the action 3) results in adding \( a^1_i + a^2_i + \ldots + a^b_i + a^1_i + a^{b-1}_i + \ldots + a^1_i \) to the content of the counter.

The automaton ends its work after performing the actions 1), 2), 3). If the counter is empty at this moment then the input word is accepted, otherwise it is rejected.
In order to prove the probability of the right result, we first note that random numbers \( y \) and \( a \) are chosen to be statistically independent.

If \( x \in W \) then for all possible choices of random numbers the counter is empty at the end and the input word is accepted. If \( x \in W' \setminus W \) then \( S_F(x) = 0 \) and at the end the counter contains
\[
a^1 i_1 + a^2 i_2 + \ldots + a^b i_b - a^1 j_1 - a^2 j_2 - \ldots - a^b j_b.
\]

Taking into account the properties of Vandermonde determinant it may be proved that no more than \( b-1 \) out of \( 2^b \) possible values of a allow equality
\[
a^1 (i_1 - j_1) + a^2 (i_2 - j_2) + \ldots + a^b (i_b - j_b) = 0
\]
if not all \( i_1 = j_1, i_2 = j_2, \ldots, i_b = j_b \). In the beginning of our proof we made a distinction between large and small values of \( b \), and for large values of \( b \) the probability of the error \( b/2^c < \varepsilon \).

Now we consider \( x \in W' \). Lemma 2.1 asserts that for arbitrary fixed value \( a \) the probability \( P_a \) of \( S_F(x) \) being
\[
(a^1 i_1 + \ldots + a^b i_b - a^1 j_1 - a^2 j_2 - \ldots - a^b j_b)
\]
does not exceed \( \varepsilon \). Since \( a \) is chosen independently of \( y \), the total probability of error equals the mean value of \( P_a \) for all \( a \)'s and it does not exceed \( \varepsilon \).

(2) Let \( g > 2 \). We will prove that if an arbitrary deterministic (and even nondeterministic) \( g \)-head 1-way finite automaton accepts all words in \( W \) with the parameter \( b \) equal \( g^8 \) then it accepts at least one word not in \( W \).

Let \( n \) be sufficiently large integer (we make this restriction precise below in the proof). By the definition of \( W \), the word \( x \in W \) is completely described by the parameters \( b, i_1, i_2, \ldots, i_b \). We fix a finite subset \( S_0 \subseteq W \) consisting at all the words \( x \in W \) for which \( b = g^8 \) and
\[
i_1 + i_2 + \ldots + i_8 = n,
i_8 + 1 + i_8 + 1 + \ldots + i_2 g^8 = n,
i_b - g^6 + 1 + i_b - g^6 + 2 + \ldots + i_b = n.
\]

We divide the word \( x \) into the following nonintersecting subwords:
\[
W_1 = A(b)7E_b(i_1,1)6E_b(i_2,2)6\ldots6E_b(i_8,g^6),
\]
$W_2 = 6E_b(i_{g+1}, g^6+1)6E_b(i_{g+2}, g^6+2)6\ldots 6E_b(i_{2g^6}, 2g^6),$

$W_g^2 = 6E_b(i_{b-g^6+1}, b-g^6+1)6E_b(i_{b-g^6+2}, b-g^6+2)6\ldots 7E_b(i_{b}, b)$,

$W_{g^2+1} = 7E_b(i_{b}, b)6\ldots 6E_b(i_{b-g^6+2}, b-g^6+2)6E_b(i_{b-g^6+1}, b-g^6+1),$

It is easy to see that

$$\text{card}(S_0) = \left(\frac{n+g^6-1}{g-1}\right)^2$$

Configuration of the g-head automaton at a fixed moment is a $(g+1)$-tuple $(\sigma, p_1, \ldots, p_g)$ where $\sigma$ is internal state of the automaton and $p_i$ is the index of the tape square observed by the head $h_i$. The type of the configuration is a $g$-tuple $(q_1, \ldots, q_g)$ where $q_i$ is the index $j$ of the subword $W_j$ observed by the head $h_i$.

If the automaton is nondeterministic then we fix one possible accepting path of computation for every $x \in S_0$. Let $c_1(x), c_2(x), \ldots, c_t(x)$ be the sequence of the configurations. Let $d_1(x), \ldots, d_1(x)$ be the subsequence obtained by taking $d_1(x) = c_1(x)$ and all $c_i(x)$ such that type $(c_i(x)) \neq \text{type}(c_{i-1}(x))$. The subsequence $d_1(x), \ldots, d_1(x)$ is called the schema of the computation.

Let $S$ be the number of the states. Since $1 < g (2g^2-1)+1$, the number of possible schemes do not exceed $O(n^{2g^4})$.

We divide $S_0$ into subsets with respect of the schema. The cardinality of the largest subset (we denote it by $S_1$) is no less than

$$\left(\frac{n+g^6-1}{g-1}\right)^2 / O(n^{2g^4}).$$

We denote the schema corresponding to $S_1$ by $d_1, \ldots, d_1$.

We say that the subword $W_i$ correspond to $W_{2g^2-i+1}$. If one of the $\left(\begin{array}{c} \frac{g^2}{2} \\ \end{array}\right)$ pairs of heads at some moment is placed on a pair of corresponding subwords $W_i$ and $W_{2g^2-i+1}$ then at no moment they are placed on another pair of corresponding subwords $W_j$ and $W_{2g^2-j+1}$, $j \neq i$. Since there are more corresponding pairs of subwords than pairs of heads there is at least one pair of subwords $(W_i, W_{2g^2-i+1})$ which is never simultaneously observed by any pair of
heads. Knowing the schema we can reconstruct such an i. Let l be such an i for the schema $d_1, ..., d_l$.

We divide $S_1$ further into subsets with respect to the values of $W_1, ..., W_{l-1}, W_{l+1}, ..., W_{2g^2-1}, W_{2g^2-1+2}, ..., W_{2g^2}$ (all $W_j$ are taken except $W_1$ and $W_{2g^2-l+1}$). The largest of these subsets is denoted by $S_2$. It contains

$$\left(\frac{n+g^6}{g^8-1}\right)^{g^2} = O\left(\frac{n+g^6}{g^8-1}\right) = O\left(n^{g^6-2g^4-1}\right)$$

words. When we started to prove (2) we announced that $n$ is sufficiently large integer and promised to make this statement precise. Now we have suitable terms for the precision. Namely, we demand that $n$ is large enough such that $S_2$ contains at least two different words. Let these words be, respectively,

$$X = W_1 \cdots W_{l-1} X_1 W_{l+1} \cdots W_{2g^2-1} X_{2g^2-l+1} W_{2g^2-l+2} \cdots W_{2g^2},$$

$$Y = W_1 \cdots W_{l-1} Y_1 W_{l+1} \cdots W_{2g^2-1} Y_{2g^2-l+1} W_{2g^2-l+2} \cdots W_{2g^2},$$

Now we consider the following word:

$$Z = W_1 \cdots W_{l-1} X_1 W_{l+1} \cdots W_{2g^2-1} Y_{2g^2-l+1} W_{2g^2-l+2} \cdots W_{2g^2}.$$

We will prove that the multi-head automaton assumed to accept all words in $W$ accepts the word $Z$ as well. Indeed, we consider the sequences of the configurations $\{c_i(x)\}$, $\{c_i(y)\}$ and construct admissible sequence $\{c_i(z)\}$. To do this, we divide the given sequences into blocks of configurations such that the first (and only the first) configuration in every block is from the schema $d_1, ..., d_l$. To construct $\{c_i(z)\}$ we take the blocks in which no head reads the subword $Y_{2g^2-l+1}$, from $\{c_i(x)\}$ and we take the blocks in which at least one head reads $Y_{2g^2-l+1}$, from $\{c_i(y)\}$. Note that at no moment a pair of heads reads $X_1$ and $Y_{2g^2-l+1}$ simultaneously but all the subwords of $W$ type are in $x$ and $y$ as well. Hence the automaton cannot distinguish $z$ from $x$ or from $y$ and accepts it along with $x$ and $y$. 
4. TWO-COUNTER PROBABILISTIC AUTOMATA VERSUS
MULTI-COUNTER DETERMINISTIC ONES

We considered a language \( W \) in Section 3 and proved that it can be recognized by a 3-head probabilistic finite automaton but cannot be recognized by a multi-head deterministic finite automaton. Counter automata (with one one-way head) differ from multi-head finite automata rather much. Nonetheless the same language \( W \) can distinguish capabilities of multi-counter probabilistic and deterministic automata (Theorem 4.1). Later in Theorem 4.2 we consider a still more complicated language and get a wider gap in complexity.

**Theorem 4.1.** (1) For arbitrary \( \varepsilon > 0 \) there is a probabilistic 1-way 3-counter automaton which recognizes the language \( W \) in real time accepting every word in \( W \) with probability 1 and rejecting every word in \( \bar{W} \) with probability \( 1 - \varepsilon \). (2) No deterministic 1-way multi-counter automaton can recognize \( W \) in real time.

**Proof.** (1) Like in Theorem 3.1 we consider separately the work of the automaton for large values of \( b \) \((b/z^B<\varepsilon)\) and for small values of \( b \).

For small values of \( b \) the proof of Theorem 3.1 contains an algorithm for a probabilistic automaton with 1 counter, and a description how to modify this algorithm to get a 2-head finite automaton. Our proof involves the same 1-counter automaton without any modifications.

For large values of \( b \) the processing of the input word consists of actions 1), 2), 3) described in the proof of Theorem 3.1.

The action 1) is already described in terms of a counter automaton. It remains to note additionally that this action can be performed in real time.

The action 2) and, hence, the action 3) in our case is organized in a different way. While the head \( h_1 \) (now it is the only head) reads the block \( A(b) \), a random number \( a \) (produced in the same way) is recorded by the second counter. When \( h_1 \) traverses \( E_b(i_1,1) \), the content of the second counter is used to find the block \( D(a^1_1,i_1) \). Since the value of \( a \) will be needed many times more, it should not be lost. Therefore whenever a unit is
subtracted from the content of the second counter, it is immediately added to the content of the third counter. When the second counter becomes empty, the third counter contains a, they change the roles and the automaton is ready to search for the block \( D(a^2, i_2) \), etc.

(2) Assume from the contrary that there such an automaton exist. We denote the number of its counters by \( k \) and the number of states by \( S \).

Let \( n \) be a sufficiently large positive integer. We consider a finite set \( S_0 \) of words in \( W \). For all these words the parameter \( b \) equals \( 2k+1 \) but \( i_1, i_2, \ldots, i_b \) take all possible values from the set \( \{1, 2, 3, \ldots, n\} \). Then \( \text{card} \ (S_0) = n^{2k+1} \). The length of the words in \( S_0 \) does not exceed \( c_k n \), where \( c_k \) depends on \( k \) but not on \( n \).

Configuration of a \( k \)-counter automaton at a definite moment is a \((k+1)\)-tuple \((\sigma, p_1, \ldots, p_k)\) where \( \sigma \) is the state of the automaton, and \( p_i \) is the content of the \( i \)-th counter. Note that in \( t \) steps the content of a counter can enlarge at most for \( t \) units. Hence, a real-time automaton can have no more than \( S \ (c_k n)^k \) different configurations on words in the set \( S_0 \).

For arbitrary \( x \in S_0 \) we fix an admissible sequence of instructions performed by the automaton on \( x \) such that the automaton accepts \( x \). (This fixation makes sense only if the automaton is nondeterministic. If it is deterministic then only one such a sequence is possible). For arbitrary \( x \in S_0 \) we consider the configuration of the automaton on \( x \), provided the fixed sequence of performed instructions, at the moment when the head reads the third symbol \( 7 \) of the word (this is the center of the word).

Now we make precise the requirement for \( n \). It should be large enough to ensure \( n^{2k+1} > S \ (c_k n)^k \). Then \( S_0 \) contains at least two different words \( x \) and \( y \) for which the abovementioned configurations coincide. We construct a new word \( z \) from the given \( x \) and \( y \), taking the head of the word \( x \) (up to the third symbol \( 7 \)) and the tail of the word \( y \) (from the third symbol \( 7 \)). The word \( z \) is not in \( W \) but the automaton accepts it along with \( y \). Contradiction.

Modifying the language \( W \), it is possible to strengthen the assertion (1) in Theorem 4.1 replacing the probabilistic 3-counter automaton by a 2-counter one, keeping (2) untouched at
the same time. We present here only the essence of the improvement.

For the new language the blocks A(b), B(m,k), C(m,k), D(n,i) are defined precisely as for W.

\[
E_b(i,k) : C(1,k)5D(i^k,1)5C(2,k)4D(2^k,1)4... \nonumber \\
...4C(2^b-1,k)4D((2^b-1)^k,1)5C(2^b,k)5D((2^b)^k,1) \nonumber \\
F_b(i,k) : C(2^b,k)5D((2^b)^k,1)5C(2^b-1,k)4D((2^b-1)^k,1)4... \nonumber \\
...4C(2,k)4D(2^k,1)5C(1,k)5D(1^k,1) \nonumber \\
\text{code}_b(i_1,i_2,...,i_{2b},j_2b,...,j_{2},j_1) : A(b)7E_b(i_1,1)6F_b(i_2,2)6 \nonumber \\
E_b(i_3,3)6F_b(i_4,4)6...6E_b(i_2b-1,2b-1)7F_b(i_{2b},2b)7 \nonumber \\
E_b(j_{2b},2b)6F_b(j_{2b-1},2b-1)6...6E_b(j_4,4)6F_b(j_3,3)6 \nonumber \\
E_b(j_{2},2)7F_b(j_1,1). \nonumber \\
\]

\[V=\{x / x\in\{0,1,2,3,4,5,6,7\}^* \& (3b>2,i_1,i_2,...,i_{2b})
\]

\[ (x=\text{code}_b(i_1,i_2,...,i_{2b},i_{2b},...,i_2,i_1)\}. \nonumber \\
\]

**THEOREM 4.2.** (1) For arbitrary \(c>0\) there is a probabilistic 1-way 2-counter automaton which recognizes the language \(V\) in real time accepting every word in \(V\) with probability 1 and rejecting every word in \(\overline{V}\) with probability \(1-c\). (2) No deterministic 1-way multi-counter automaton can recognize \(V\) in real time.

## 5. OTHER APPLICATIONS OF LEMMA 2.1.

Lemma 2.1 is a useful tool that allows us to use multiple random choices provided they are statistically independent. Some more profound results on probabilistic algorithms are proved as well which are a bit outside our topic. We note here two results of this kind.

Undecidability of the emptiness problem for languages recognizable by probabilistic 2-head 1-way finite automata was proved by the author in 1980 (English version see in [Fre 83]).

**INPUT:** an infinite sequence of probabilistic 2-tape 1-way finite automata all of which recognize the same language \(L\); the automata accept every pair of words in \(L\) with probability 1 and
reject every pair in $\overline{L}$ with probabilistic $2/3$, $3/4$, $4/5$, ..., respectively.

PROPERTY: $L$ is empty.

It is known that a projection of a language recognized by a deterministic multihead 1-way finite automaton to one of its tapes is a regular language. A similar result holds also for nondeterministic automata but not for probabilistic ones. In the latter case the projection, of course, is a recursively enumerable language as a projection of a recursive language. It turns out that one can say no more.

THEOREM 5.1. Given arbitrary recursively enumerable language $L$ of strings in a finite alphabet, there is a language $K$ of triples of words such that: 1) $L$ is the projection of $K$ to the first tape, 2) given arbitrary $\epsilon > 0$, there is a probabilistic 3-tape 1-way finite automaton which accepts every triple of words in $K$ with probability 1 and rejects every triple of strings in $\overline{K}$ with probability $1 - \epsilon$.

Only for recursive languages $L$ it may be possible to replace the language $K$ of triples of words by a language $K'$ of pairs of words.

6. PROBABILISTIC RECOGNITION OF PALINDROMES

The methods considered above could prove advantages of probabilistic machines over their deterministic counterparts only for 1-way machines. Now we consider a new method. Suppose, it is needed to compare two objects (strings, matrices, polynomials) whether or not they are identical. Instead of full scale comparison we propose to consider a large set of simple functions defined on these objects, to pick one function at random and compute its value on the two given objects. If the objects are really identical then the obtained values coincide. The set of functions should be chosen to ensure that for every pair of nonidentical objects most of these functions expose their distinctness.

The author proved his first theorem by this method in 1975. It was known [Bar 65] that palindromes cannot be recognized by deterministic 1-head off-line Turing machines in less time than $\text{const } n^2$. It turned out [Fre 75] that probabilistic machines can
have less running time (see Theorem 6.1 below). This theorem and
the proof have been published in several modifications (see also
[Fre 77], [Fre 83]). For the sake of brevity, we include only a
brief sketch of proof here. On the other hand, the first lower
bound for probabilistic Turing machines ([Fre 75], [Fre 79])
seems never having been published in English. Hence it is
included here in full detail (Theorem 6.2 below).

**THEOREM 6.1.** For arbitrary $\epsilon > 0$, there is a probabilistic
1-head off-line Turing machine recognizing palindromes with
probability $1-\epsilon$ in const $n \log n$ time.

**SKETCH OF PROOF.** The input word $x$ itself and its reversion
are interpreted as binary notations of numbers $\overline{x}, \overline{y}$. They are
compared modulo a small random prime number. For this, a random
string $m \in \{0,1\}^d$, $d = \lceil \log_2 c \cdot |x| \rceil$ where $c$ is an absolute constant
obtained from theory-of-numbers considerations. If the string
turns out to represent a prime number $\overline{m}$, it is tested whether $x \equiv y$
(mod $\overline{m}$) holds. The result of this test is the output of the
probabilistic machine. If $\overline{m}$ is not a prime number, a new string $m$
is generated, and so on. The proof of the estimate of the running
time and the probability of the correct result involve Cebiáev
theorem on density of prime numbers.

**THEOREM 6.2.** Let $\epsilon < 1/2$ and a probabilistic 1-head off-line
Turing machine recognize palindromes in $\{0,1\}^*$ with probability
$1-\epsilon$ in time $t(x)$. Then there is a $c > 0$ such that
$t(x) > c \cdot |x| \cdot \log_2 |x|$ for infinitely many words $x$.

**PROOF.** Let the machine $M'$ recognize palindromes. We can
assume that $M'$ always starts on the leftmost symbol of the
input word.

We modify $M'$ to get an additional property. The new
machine $\overline{M}$ at first marks the leftmost and the rightmost
nonempty symbols of the input word. Then $\overline{M}$ simulates the work
of $M'$ keeping precise records of where the most extreme ever
visited squares of the tape are. When the simulation ends because
$M'$ has produced the result, the new machine ends its work by
walking through all the used part of the tape in a special state
$q(0)$ or $q(1)'$, respectively. This way, the running time has
increased at most by $O(|x|)$ but the machine has acquired a new
important property: every nonempty crossing sequence allows to
reconstruct the result. It suffices to prove the Theorem for the
machine $\overline{M}$ only.
We call a palindrome special if its length equals 0 modulo 3, and the central third part consists of zeros only. The set of squares on the tape corresponding to this central third part is called the central zone.

Let \( n \) be an arbitrary integer. We consider the work of \( \mathbb{M} \) on the special palindromes of length \( 3n \). The set of all these special palindromes is denoted by \( S_n \).

Let \( \mathbb{M} \) recognize palindromes with probability \( 1-\varepsilon \) in time \( t(x) \). We define the function \( \sigma(n) \) such that

\[
\frac{n (\sigma(n)-1)}{n} < \max_{|x|=3n} t(x) < n \sigma(n),
\]

where the maximum is taken over all the special palindromes of length \( 3n \). To prove the Theorem, it suffices to show that \( \sigma(n) > O(\log^2 n) \).

We consider admissible sequences of instructions corresponding to the given machine \( \mathbb{M} \) and the given input word \( x \). All possible admissible sequences of instructions of \( \mathbb{M} \) on \( x \in S_n \) can be divided into two subsets:

a) "good", i.e. producing the result 1 in no more than \( n \sigma(n) \) steps,

b) "bad", i.e. all the other possible sequences.

Note that there can be only finitely many "good" sequences, while infinitely many "bad" ones are possible.

Knowing the program of the machine and the probabilities of the random number generator (which is equiprobable and Bernoulli type), it is easy to compute the probability of each admissible sequence of instructions. The total probability of the "good" sequences for every \( x \in S_n \) exceeds \( 1-\varepsilon \).

Consider

\[
\sum_{\rho \text{ being "good" sequence}} p_{\rho} \sum_{t=1}^{n \sigma(n)} \sum_{i \in \text{central zone}} X(\rho, i, t) \quad (6.2.1)
\]

where \( p_{\rho} \) is the probability of the sequence \( \rho \) of instructions of \( \mathbb{M} \) on the given \( x \in S_n \), and
\[ X(\rho, i, t) = \begin{cases} 
1, & \text{if, when performing the } t\text{-th instruction in the} \\
& \text{sequence } \rho, \text{ the head crosses the point } i; \\
0, & \text{if otherwise.} 
\end{cases} \]

On the one hand, the innermost sum does not exceed 1, and hence the total \((6.2.1)\) does not exceed \(n \sigma(n)\). On the other hand, all sums in \((6.2.1)\) are finite. Hence

\[
\sum_{\rho \text{ being } "good" \text{ sequence}} \left( \sum_{\text{is central zone}} \left( \sum_{t=1}^{n \sigma(n)} p_{\rho} X(\rho, i, t) \right) \right) < n \sigma(n)
\]

and for arbitrary \(x \in S_n\) it is possible to fix a point \(i_0(x)\) in the central zone such that

\[
\sum_{\rho \text{ being } "good" \text{ sequence}} \left( \sum_{t=1}^{n \sigma(n)} p_{\rho} X(\rho, i_0(x), t) \right) < \sigma(n).
\]

We will refer this \(i_0(x)\) as the checkpoint of \(x\).

The formula above contains the internal sum

\[
\sum_{t=1}^{n \sigma(n)} X(\rho, i_0(x), t)
\]

expressing the number of times when the head crosses \(i_0(x)\) at the sequence of instructions \(\rho\). All the members in the sum are nonnegative. Hence for arbitrary \(\delta > 0\) it is true that the probability, for the given \(\mathcal{M}\) and \(x\), of the event when simultaneously 1) the sequence \(\rho\) is "good", and 2) the length of the crossing sequence in the checkpoint exceeds \(1/\delta \sigma(n)\), does not exceed \(\delta\).

Hence, for arbitrary \(\delta > 0\) it is true that the probability of the event, for the given \(\mathcal{M}\) and \(x\), when simultaneously 1) the result is correct, and 2) the length of the crossing sequence in the checkpoint does not exceed \(1/\delta \sigma(n)\), is no less than \(1-\varepsilon-\delta\). In particular, the abovementioned is true for \(\delta = 1-2\varepsilon/4\).
To define the notions "the leftside probability of the crossing sequence", "the rightside probability of the crossing sequence", we consider the following procedure $\gamma$ for the given $x \in S_n$, checkpoint $i_o(x)$ and crossing sequence $\tau=q(1) q(2) \ldots q(2r-1)$ where $q(2r-1)=q(1)$.

The procedure starts with simulation of the work of $M$ on $x$. The simulation continues uninterrupted until the head crosses the checkpoint. If at the given sequence of instructions $M$ comes to the checkpoint in the state $q(1)$ then the simulation is temporarily interrupted, the state replaced by $q(2)$, the head returned through the checkpoint into the square of the tape bordering the checkpoint on the left side, and then the simulation is continued. If $M$ comes to the checkpoint for the first time in a state different from $q(1)$ then the procedure stops without result.

If the head comes to the checkpoint (from the left) for the second time in the state $q(3)$ then the head is returned automatically in the state $q(4)$ (if the head comes to the checkpoint in a state different from $q(3)$ then the procedure stops without result), etc.

If this way the procedure reaches a moment when the head is going to cross the checkpoint from the left to the right for the $r$-th time in the state $q(2r-1)$ then the procedure comes to an accepting end.

The probability of the abovementioned procedure with the given $M$, $x$, $i_o(x)$ and $\tau$ is called the leftside probability of the crossing sequence $\tau$ in $i_o(x)$ for $M$ and $x$.

The rightside probability is defined in a similar way (only the simulation is performed on the part of the tape to the right from the checkpoint).

It is easy to see that the leftside and rightside probabilities are nonnegative numbers not exceeding 1 and their product equals the probability of the event "given $M$, $x$, $i_o(x)$, the crossing sequence of $M$ and $x$ at $i_o(x)$ equals $\tau$". (Note that we use here the assumption $q(2r-1)=q(1)$, in the general case the product expresses the event "the crossing sequence has an initial fragment $\tau".

Let $x'$ and $x''$ be two different words from $S_n$. Let $x'''$ be a word consisting of the head (up to the checkpoint $i_o(x')$) of the word $x'$ and the tail (after the checkpoint $i_o(x'')$) of the word
The word $x'''$ may be of length different from $3n$ but, any case, $x'''$ is not a palindrome.

We denote by $\tau_1, \tau_2, \tau_3, \ldots$ all possible crossing sequences allowing acception of the input word. We assume that the crossing sequences are ordered in increasing length.

We denote by $\xi'_1, \xi'_2, \xi'_3, \ldots$ the leftside probabilities of the crossing sequences $\tau_1, \tau_2, \tau_3, \ldots$ for $x'$ in $i_0(x')$. We denote by $\eta'_1, \eta'_2, \eta'_3, \ldots$ the rightside probabilities of the crossing sequences $\tau_1, \tau_2, \tau_3, \ldots$ for $x'$ in $i_0(x')$. We denote by $\xi''_1, \xi''_2, \xi''_3, \ldots$ the rightside probabilities of $\tau_1, \tau_2, \tau_3, \ldots$ for $x''$ in $i_0(x'')$.

We denote by $s$ the index of the crossing sequence $\tau_s$ such that all the crossing sequences $\tau_1, \tau_2, \ldots, \tau_s$ and only these crossing sequences have the properties: 1) they lead to the acception of the input word, and 2) their length does not exceed $4/1-2e\sigma(n)$. It is easy to see that

$$s < 2^O(\sigma(n)),$$

$$\xi'_1 \eta'_1 + \xi'_2 \eta'_2 + \ldots > 1-e,$$

$$\xi''_1 \eta''_1 + \xi''_2 \eta''_2 + \ldots + \xi''_s \eta''_s > 1-e - 1-2e/4 = 3-2e/4 \quad (6.2.2)$$

The probabilities of the crossing sequences in $i_0(x')$ for $x'$ and $x'''$ being $\tau_1, \tau_2, \tau_3, \ldots$, respectively, are $\xi'_1 \eta'_1, \xi'_2 \eta'_2, \xi'_3 \eta'_3, \ldots$. The input word $x'''$ is not a palindrome. Hence

$$\xi''_1 \eta''_1 + \xi''_2 \eta''_2 + \ldots + \xi''_s \eta''_s + \ldots < e .$$

All the terms in the left part of the inequality are nonnegative. Hence

$$\xi''_1 \eta''_1 + \xi''_2 \eta''_2 + \ldots + \xi''_s \eta''_s < e$$

Subtracting this from (6.2.2) we get

$$(\xi'_1 - \xi''_1) \eta'_1 + (\xi'_2 - \xi''_2) \eta'_2 + \ldots + (\xi'_s - \xi''_s) \eta'_s > 3-6e/4$$

Hence

$$\xi'_1 - \xi''_1 + \xi'_2 - \xi''_2 + \ldots + \xi'_s - \xi''_s > 3-6e/4 \quad (6.2.3)$$

The $S$-tuple

$$\xi(x) = (\xi'_1, \xi'_2, \ldots, \xi'_s)$$
of the leftside probabilities of the crossing sequences \( \tau_1, \tau_2, \ldots, \tau_3 \) of \( x \) in its checkpoint can be understood as a point in an \( s \)-dimensional unit cube. The inequality (6.2.3) shows that the points corresponding to distinct special palindromes from \( S_n \) should be distant in the metrics

\[
\rho(x', x'') = \xi_1' - \xi_1'' + \xi_2' - \xi_2'' + \ldots + \xi_s' - \xi_s''.
\]

Around arbitrary point \((\xi_1^0, \xi_2^0, \ldots, \xi_s^0)\) corresponding a special palindrome from \( S_n \) we circumscribe a body

\[
\xi_1 - \xi_1^0 + \xi_2 - \xi_2^0 + \ldots + \xi_s - \xi_s^0 < 1 - 2\varepsilon/4.
\]

It follows from (6.2.3) that these bodies do not intersect. The volume of every such body equals

\[
\left(\frac{1 - 2\varepsilon}{4}\right)^s = \frac{S^s}{s!} 2^S.
\]

All they are situated in an \( S \)-dimensional cube with the side length

\[
1 + 2 \frac{1 - 2\varepsilon}{4} = \frac{3 - 2\varepsilon}{2}.
\]

Hence, the number of the distinct special palindromes in \( S_n \) does not exceed

\[
\left(\frac{3 - 2\varepsilon}{2}\right)^s (s!) 2^{s/2} = 2^{s \log_2 s}.
\]

Hence

\[
2^n < 2^{s \log_2 s}.
\]

On the other hand,

\[
S > 0 \left(\frac{n}{\log_2 n}\right)
\]

Hence

\[
0 \left(\frac{n}{\log_2 n}\right) < 2^{\sigma(n)}
\]

and

\[
\sigma(n) > 0(\log_2 n).
\]
7. BOOLEAN CIRCUITS WITH MEMORY

M.O. Rabin [Rab 63] considered language recognition by probabilistic finite automata and found an essential property of such recognition which influences the practical meaningfulness of this process. This property is cut-point's being isolated.

The most natural way to define what does it mean when a probabilistic automaton \( \mathfrak{A} \) recognizes a language \( L \) with cut-point \( \lambda \), is as follows. The automaton \( \mathfrak{A} \) accepts arbitrary input word \( x \) with the probability \( p(x) \). We say that \( \mathfrak{A} \) accepts \( x \), if \( p(x) > \lambda \) (a version of the definition: if \( p(x) > \lambda \)), and \( \mathfrak{A} \) rejects \( x \), if otherwise.

For practical purposes such a "recognition" is somewhat dubious because it may be hard to distinguish by a statistical experiment between \( x' \) and \( x'' \) such that \( p(x') = \lambda + \delta' \) and \( p(x'') = \lambda - \delta'' \) where \( \delta' \) and \( \delta'' \) are very small. We have a different case when for the given automaton there is a positive constant \( \delta_0 \) (called isolation radius) such that for all \( x \), either \( p(x) > \lambda + \delta_0 \) or \( p(x) < \lambda - \delta_0 \). Then it is easy to calculate how many times the experiment should be repeated to get the result with the needed probability.

M.O. Rabin [Rab 63] proved that languages recognized with isolated cut-point by probabilistic finite automata are recognized by deterministic finite automata as well. On the other hand, the complexity of the deterministic automaton can be higher.

More precisely, M.O. Rabin proved that a deterministic finite 1-way automaton needs no more than \((1+r/\delta)^{m-1}\) states to recognize a language recognized by a probabilistic finite 1-way automaton with \( m \)-states and isolation radius \( \delta \). As another result in [Rab 63], a probabilistic automaton with 2 states and cut-points \( \{\lambda_t\} \) was constructed such that the corresponding deterministic finite automaton needs no less than \( t \) states.

This way, it was shown that complexity of a probabilistic finite automaton cannot be characterized merely by the number of states. At least the isolation radius is to be taken into consideration.

We return once more to the last mentioned theorem by Rabin. A sequence \( L = \{L_t\} \) of languages was constructed such that the corresponding deterministic automata are to have no less than
$D_L(t) = t$ states. Unfortunately, the first mentioned theorem by Rabin gives us an exaggerated estimate $R_L(t) = (1 + r/\delta)^m - 1 = 3^t$.

It was an open problem: either to lower $R_L(t)$ or to construct a better sequence of languages for which

$$R_L(t) = O\left(3^{D_L(t)}\right)$$

The estimate $R_L(t)$ was somewhat lowered in [Paz 66] and [GM 78]. Unfortunately, these improvements did not influence even the exponent $3$ in $3^{D_L(t)}$. For the first time $3$ was replaced by a function growing less rapidly in [Fre 82]. Namely, it was replaced by a function growing less rapidly than any exponent but growing more rapidly than any polynomial.

This Section contains further improvements of the abovementioned theorem. A.A.Lorenc [Lor 86] in his invited talk at the USSR National Symposium on probabilistic automata, Kazan, 1983 turned everybody's attention to the fact that in all the results of the considered type either very small isolation radius are used or very complicated probabilities of transition among the states of the automaton are used. Any case, if the probabilistic finite automata are represented by Boolean circuits with 2-argument Boolean gates, 1-bit memory elements and the simplest random number generators producing zeros and ones equiprobably by Bernoulli scheme, i.e. independently, then the probabilistic circuit is no less complicating than the deterministic one.

A.A.Lorenc proposed to construct a sequence of languages $B = \{B_t\}$ such that in the probabilistic Boolean circuit with memory: 1) the cut-point is $1/2$, 2) the isolation radius does not depend on $t$ and is reasonably large (for instance, $1/4$), 3) the complexity of the probabilistic circuit turns out to be essentially smaller than the complexity of every deterministic Boolean circuit with memory representing the same language. In fact, a more realistic complexity measure for probabilistic finite automata is considered.

In our results in this Section the Boolean circuit with memory has one input and one output. The input and output alphabets are $\{0, 1\}$. At the first step the first input symbol is read, at the second step the second input symbol is read, etc. At any step the automaton claims to output the result of whether or
not the input word read up to this moment in the language represented by the automaton.

We will compare complexity of circuits over basis $\mathfrak{A}$ versus basis $\mathfrak{B}$. Basis $\mathfrak{A}$ consists of the 2-argument conjunction, the 2-argument disjunction, the 1-argument negation (these elements have no delay in time), and 1-argument memory element which outputs at any moment the value of its input read at the previous moment. Basis $\mathfrak{B}$ contains all the mentioned elements, and it contains additionally a special 0-argument element the output of which equals 1 with probability $1/2$, and 0 with probability with $1/2$, and outputs at any moment are statistically independent of the outputs at other moments. Hence, circuits over $\mathfrak{A}$ are deterministic and circuits over $\mathfrak{B}$ are probabilistic.

We say that the circuit recognizes the language $L$ with isolated cut-point $\lambda$ and isolation radius $\delta$ if the circuit accepts every $x \in L$ with probability no less than $\lambda + \delta$, and the circuit accepts every $x \not\in L$ with probability not exceeding $\lambda - \delta$.

We consider the sequence of languages $D = \{D_t\}$. The language $D_t$ consists of one word only, namely, $0^2t^1$.

**Assertion 7.1.** Arbitrary (deterministic) Boolean circuit with memory over basis $\mathfrak{A}$, which recognizes the language $D_t$ has at least $t+1$ elements of memory.

**Proof.** Immediate.

**Theorem 7.2.** For arbitrary $t$ there is a (probabilistic) Boolean circuit with memory over basis $\mathfrak{B}$ which recognizes the language $D_t$ with the cut-point $1/2$, and isolation radius $1/4$. The circuit has 1 simplest random element, $O(\log t)$ memory elements and $O(t/\log t)$ Boolean elements.

**Proof.** Following the traditional notation in number theory textbooks (e.g. [Bu 60]) we denote the increasing sequence of all prime numbers by $p_1, p_2, p_3, \ldots$ ($p_1 = 2$, $p_2 = 3$, $p_3 = 5, \ldots$) Čebišev function $\Phi(x)$ is the sum of natural logarithms of all prime numbers not exceeding $x$. $\Phi(x) = \sum_{p \leq x} \ln p$.

We introduce auxiliary function $F(t)$ equal to $p_1 \cdot p_2 \cdot p_3 \cdots p_t = (p_t)$. Note that if a positive integer $x$ does not exceed $F(t)$ then $x$ has no more than $t$ distinct prime divisors. The function
\( \Pi(x) \) denotes the number of distinct primes not exceeding \( x \).

P.L. Čebiáev [Čeb 44] proved that there are positive numbers \( a, b, c, d \) such that:
\[
ax < \Pi(x) < bx,
\]
\[
c \frac{x}{\ln x} < \Pi(x) < d \frac{x}{\ln x},
\]
\[
\frac{1}{d} t \ln t < p_t < \frac{2}{c} t \ln t.
\]
Hence \( F(t) \) has the order of magnitude
\[
(\text{const}) t \ln t
\]

It is well-known that
\[
\lim_{k \to +\infty} \left( 1 - \frac{1}{k} \right)^k = \frac{1}{e}.
\]

Let \( f \) and \( g \) be large positive integers such that
\[
\lim_{k \to +\infty} \left( 1 - \frac{1}{k} \right)^{fk} < \frac{1}{64}, \quad \lim_{k \to +\infty} \left( 1 - \frac{1}{k} \right)^{k/g} > \frac{7}{8}.
\]
Hence there is a \( k_o \) such that for arbitrary \( k > k_o \)
\[
\left( 1 - \frac{1}{k} \right)^{fk} < \frac{1}{64} \quad (7.2.1)
\]
\[
\left( 1 - \frac{1}{k} \right)^{k/g} > \frac{7}{8} \quad (7.2.2)
\]

Let \( l \) be a large positive integer such that \( F(l) > 2^t \). Surely, \( F(l) < 2^{t+1} \). Hence
\[
t = O(l \log l)
\]
By \( r \) we denote the number \( \left\lfloor \log_2 p_{41+5} \right\rfloor \), and by \( S \) we denote \( 2^{r+1} - 1 \). It follows from Čebiáev theorems that \( p_{41+5} \) and \( S \) have the order of magnitude \( t \), and \( r = \log_2 t + O(1) \).

By \( u \) we denote the nearest complete power of 2 exceeding \( 2t \). Surely, \( u < 4t \).

Let \( w \) be a positive solution of the equation
\[
g \left( 2^t - u \right) / w = 2^w
\]
and \( v = \lfloor w \rfloor \). Then
\[
(2^{t-u})/v < (2^{t-u})/w = 2^w/g < 2^v/g \quad (7.2.3)
\]
Hence \( w = t - o(t) \) and for large \( t \)
\[
\frac{2^{t-u}}{w} < \frac{3}{2} \left( \frac{2^{t-u}}{v} \right).
\]

Since \( v < w+1 \), we have \( 2^v < 2^{w+1} \). Hence for large \( t \)
\[
\frac{2^v}{g} < 3 \frac{2^{t-u}}{v}
\]  \hspace{1cm} (7.2.4)

It follows from (7.2.2) that
\[
\left( 1 - \frac{1}{2^v} \right) 2^v/g > \frac{7}{8}.
\]

Taking into account (7.2.3), we have
\[
\frac{2^{t-u}}{v} \left( 1 - \frac{1}{2^v} \right) > \frac{7}{8}
\]  \hspace{1cm} (7.2.5)

It follows from (7.2.1) that
\[
\left( 1 - \frac{1}{2^v} \right) f 2^v < \frac{1}{64}.
\]

Taking into account (7.2.4), we have for large \( t \)
\[
\left( 1 - \frac{1}{2^v} \right) f g 3(2^{t-u})/v < \left( 1 - \frac{1}{2^v} \right) f g 2^v/g < \frac{1}{64},
\]
\[
\left( 1 - \frac{1}{2^v} \right) f g (2^{t-u})/v < \frac{1}{4}
\]  \hspace{1cm} (7.2.6)

The formulation of the assertion in our Theorem is only asymptotic. Hence it suffices to prove it only for large \( t \). We prove the Theorem for large \( t \) such that: 1) \( \frac{f}{c} \ln s < t-1 \); 2) \( F(l+1) > fgF(l) \); 3) \( \ln s > c k_o \); 4) \( u < 2^t \); 5) (7.2.6) holds.

The circuit in demand consists of 6 blocks.

The block 1 is a counter up to the number \( u \). One memory element is fixed in this block. This element is supposed to have as its output for the first \( u \) moments the value 0 and after that the value 1. The block ends its work after producing the first 1. Such a block can be constructed using no more than \( \left\lceil \log_2 u \right\rceil = \log_2 t + O(1) \) memory elements and \( O(\log t) \) Boolean instantaneous elements.
The blocks 2 and 3 work in parallel with 1 but completely independent from 1. First 2 starts, and after 2 has ended, 3 starts.

The block 2 is designed to choose a random prime number \( m < s \). The block has \( r + 1 \) memory elements. One of these elements is used to signal the end of the work of the block 2. All the remaining \( r \) memory elements are used to memorize outputs of the only random elements during \( r \) moments in row. After that a Boolean circuit consisting of \( \frac{2^r}{r} + O \left( \frac{2^r}{r} \right) \) instantaneous elements tests whether or not the obtained \( r \)-digit number is prime. The abovementioned number of elements in the circuit suffices for arbitrary Boolean function. (It follows from a theorem by O.B. Lupanov [Lup 63]). If prime then the block 2 ends and produces 1 in the special memory element. If not prime then a new \( r \)-digit random number is generated in \( r \) steps, the primality tested, etc. We remind that \( r = \log t + O(1) \), \( 2^r/r = O(t/\log t) \).

The block 3 starts only after the block 2 has ended. It is designed to find the residue modulo \( m \) (produced by the block 2) of \( 2^t - u \). More precisely, the residues of \( 2^t \) and of \( u \) are found separately and then their difference is found. To compute these residues, first the residues of \( 2^1, 2^2, 2^3, 2^4, \ldots \) modulo \( m \) are computed. For this the block has 4 sets of memory elements: two sets each consisting of \( r \) elements are used to record the residues \( (r = \log_2 t + O(1)) \), one set of \( \lceil \log_2 t \rceil \) elements is to record first \( t \), then \( t-1 \), \( t-2 \), \ldots, and one set of \( \log_2 u \) elements is to record \( u, u/2, u/4, \ldots \). In total, the block 3 has \( O(\log t) \) memory elements and \( O(\log t) \) instantaneous Boolean elements.

The block 4 starts immediately after the block 1 has ended. This block does not check whether the blocks 2 and 3 have ended but the parameters of the circuit are chosen such that with high probability blocks 2 and 3 end before the block 1. The block 4 is designed to compute the residue modulo \( m \) of the number of zeros read from the input during the work of this block, provided that no other symbols have been read from the input. The block 4 ends when the first symbol 1 is read from the input. The block has \( r = \log_2 t + O(1) \) memory elements. These elements store an \( r \)-digit binary number expressing the residue modulo \( m \) of the number of zeros read from the input up to the moment. Whenever this number reaches \( m \), i.e. it equals \( m \) recorded
by memory elements in the block 2, the residue is automatically returned to 0.

The block 5 also starts immediately after the block 1 has ended. This block is designed to enable the circuit at special moments (namely, at moments equal 0 modulo v) to enter with small probability $2^{-v}$ a special rejecting state in which the circuit remains forever and rejects all the continuations of the considered input word. For this purpose, the block 5 has $2+[\log_2 v]<\log_2 t+O(1)$ memory elements. One memory element serves to signal of transition to the special rejecting state. $[\log_2 v]$ elements make a counter modulo v. At every moment the block considers the symbol received from the random element of the circuit. If during one cycle between two adjacent moments of empty counter all the outputs of the random element equal 0 (to check this, one more memory element is needed) then the circuit enters the special rejecting state. If not, the normal work of the circuit continues.

The block 6 works after the first symbol 1 has been read from the input. If the block 5 has produced the signal of the special rejecting state then the circuit rejects all the continuations of the word. If at the moment when the first 1 is read from the input the block 1 has ended its work and the residue of the number of zeros read from the input during the work of 4 coincides with the result of the block 3 then the input word is accepted and the circuit is prepared to reject any continuation of this word. If the results of 3 and 4 differ then the circuit rejects the input word. The block 6 does not contain new memory elements and it has $O(\log t)$ instantaneous Boolean elements.

We have followed the number of elements in the circuit. Now we will prove the correctness.

The block 3 works no more than $t+1$ steps. Now we estimate the probability of the event "the block 2 completes its work in no more than $t-1$ steps". The work of this block consists of cycles the length of which equals r. Cycle is resultative if a prime number m is generated. Out of all s possible and equiprobable numbers the number of primes is $\Pi(s)$. Hence the probability to generate a prime number is $\Pi(s)/s$. By Çebiáev’s theorem, $\Pi(s)>c s/\ln s$. The probability to have generated, only compound numbers in the first $(f \ln s)/c$ cycles equals
which, due to (7.2.1), does not exceed $1/64$. Since we prove the Theorem only for large $n$ such that $r \ln s < t-1$, we may predict that with probability $63/64$ the block 2 will work no more than $t-1$ steps, and the blocks 2 and 3 in total no more than $2^t$ steps, and, hence, the block 1 will end first.

If the input word $0^n1 \in D_t$, i.e. if $n=2^t$ then the word will be accepted, provided two conditions: 1) the blocks 2 and 3 end the work before the block 1 (the probability of this event is no less that $63/64$), and 2) the block 5 does not enter the special rejecting state. To calculate the probability of 2), we note that the work of the block 5 consists of $2^t/v$ cycles and the probability of entering the special state equals $2^{-v}$. The probability to avoid this state equals

$$\left(1 - \frac{1}{2^v}\right)^{2^t/v}$$

which, by (7.2.5), exceeds $7/8$. Hence, the input word is accepted with probability

$$\frac{63}{64} \cdot \frac{7}{8} > \frac{3}{4}$$

Let $n \neq 2^t$ and $n > f \phi 2^t$. It follows from (7.2.6) that the block 5 enters the special rejecting state with probability exceeding $3/4$.

Let $n \neq 2^t$ and $n < f \phi 2^t$. Then $n < f \phi F(1)$ and $n < G(1+1)$. We only increase the estimate for the probability of error if we pay no attention to the possibility for the block 5 to enter the special rejecting state and if we pay no attention to the possibility that the block 1 end the work before the blocks 2 and 3. We will show that no less than $3/4$ of all possible values of $m$ are such that $n \equiv 2^t \pmod{m}$ and $n-u \equiv 2^t-u \pmod{m}$. Indeed, the congruence $n \equiv 2^t \pmod{m}$ holds only if $m$ divide $|n-2^t|$. The number $|n-2^t|$ does not exceed $F(1+1)$, and, hence, it has no more than $1+1$ distinct prime divisors. The number $m$ is chosen equiprobably among $P_1, P_2, P_3, \ldots, P_{41+5}$. Hence, no more than $3/4$ choices for $m$ will show that $0^n1 \notin D^t$. 

\[
\left(1 - \frac{\Pi(s)}{s}\right) < \left(1 - \frac{c}{\ln s}\right)
\]

\[
\left(1 - \frac{\Pi(s)}{s}\right) < \left(1 - \frac{c}{\ln s}\right)
\]
THEOREM 7.3. For arbitrary \( t \) there is a (probabilistic) Boolean circuit with memory over basis \( S \) which recognizes the language \( D_t \) with the cut-point \( 1/2 \), and isolation radius \( 1/4 \). The circuit has 1 simplest random element, \( O((\log t)^2) \) memory elements and \( O((\log t)^2) \) Boolean elements.

PROOF. Like the circuit in the proof of Theorem 7.2, this circuit consists of 6 blocks. The purpose of the blocks is the same. What the circuits differ in is the following. The circuit in Theorem 7.2 produced one prime modulo \( m \) and compared the number of zeros with the standard modulo this \( m \) but the new circuit does not test primality. It takes a bundle of random modulos and compares the number of zeros with the standard modulo all these \( m \). The bundle is taken large enough (namely, \( \lceil (\log s)/c \rceil = O(\log t) \) random modulos) to ensure at least one prime modulo in this bundle with high probability.

The blocks 1 and 5 completely equal their counterparts in the proof of Theorem 7.2.

The block 2 in \( r \lceil (\log s)/c \rceil \) steps generate \( \lceil (\log s)/c \rceil \) random \( r \)-digit numbers \( m \).

The blocks 3 and 4 consist of \( \lceil (\log s)/c \rceil \) copies of the corresponding block in the proof of Theorem 7.2, one per a value of \( m \).

The block 6 perform the same functions as its counterpart in the proof of Theorem 7.2.

Theorems 7.2 and 7.3 try to minimize distinct complexity measures for Boolean circuits with memory. In Theorem 7.3 the total number of elements is much less than in Theorem 7.2 but in Theorem 7.2 the number of memory elements is smaller than in Theorem 7.3. The trade-off between these two complexity measures in this context is still an open problem.
8. SPACE COMPLEXITY OF 1-WAY TURING MACHINES

Finite automata are too much restricted to be a realistic model of computers. Unrestricted Turing machines and their generalizations are hard for proving lower bounds of complexity. A nice compromise is the notion of 1-way Turing machine. They are reasonably powerful, and, on the other hand, allow nontrivial lower bounds of complexity.

This survey, of course, cannot include all the results on advantages of probabilistic machines over their deterministic counterparts. Nevertheless, there was a recent advancement for space complexity of 1-way Turing machines. In [KF 90] an open problem was solved.

If a language \( L \) is recognized by a deterministic 1-way Turing machine in \( O(\log n) \) space then \( L \) is regular [SHL 65]. In [Fre 83] it was shown that nonregular languages can be recognized with arbitrarily high probability by a \( \log \log n \)-bounded 1-way Turing machines as well. (This theorem is repeated as our Theorem 8.2).

All the results in this Section on advantages of probabilistic machines are based on the following lemma. It shows that for any two nonequal natural numbers \( N' \) and \( N'' \), there are many reasonably small prime modulos \( m \) such that \( N' \equiv N'' \mod m \). Probabilistic algorithms in this Section are based on the possibility to take such a prime modulo \( m \) at random.

Let \( P_1(1) \) be the number of primes not exceeding \( 2^{\left\lfloor \log_2 1 \right\rfloor} \), \( P_3(1, N', N'') \) be the number of primes not exceeding \( 2^{\left\lfloor \log_2 1 \right\rfloor} \) and not dividing \( |N' - N''| \), and \( P_3(1, n) \) be the maximum of \( P_2(1, N', N'') \) over all \( N' < 2^n, N'' < 2^n, N' \equiv N'' \).

**Lemma 8.1.** ([3]) Given any \( c > 0 \), there is a natural number \( c \) such that

\[
\lim_{n \to +\infty} \frac{P_3(cn, c)}{P_1(cn)} < c.
\]

**Proof.** Let \( p_1, p_2, \ldots, p_k \) be all the primes that divide \( Z = |N' - N''| < 2^n \). Since \( Z > k! \), we have \( k < O(n/\log n) \). By Chebyshev's theorem on the density of primes (see [Bu 60]), the first \( \lfloor k/c \rfloor \) primes do not exceed \( cn \) for a suitable constant \( c \).
We define a language $S \subseteq \{0,1,2,3,4,5\}^*$. Let $\text{bin}(i)$ denote a string representing $i$ in the binary notation (the first symbol of $\text{bin}(i)$ being 1).

We start to describe how the strings in $S$ can be generated. An arbitrary integer $k$ is taken and the following string is considered

$\text{bin}(1) \ 2 \ \text{bin}(2) \ 2 \ \text{bin}(3) \ 2 \ \ldots \ \ 2 \ \text{bin}(2k)$. 

Next, every symbol in the substrings $\text{bin}(k+1), \text{bin}(k+2), \ldots, \text{bin}(2k)$ is preceded by one arbitrary symbol from $\{3,4\}$. If the obtained string in $\{0,1,2,3,4\}^*$ is denoted by $x$ then the language $S$ contains the string $x5x$. Every string in $S$ is obtained by this procedure.

**THEOREM 8.1.** (1) Given any $\epsilon > 0$, there is a log $n$-space bounded probabilistic 1-way Turing machine which accepts every string in $S$ with probability 1 and rejects every string in $\overline{S}$ with probability $1-\epsilon$.

(2) Every deterministic 1-way Turing Machine recognizing $S$ uses at least const. $n$ space.

**PROOF.** (1) The head of string $w$ is its initial fragment up to the symbol 5. The tail of $w$ is the rest of the string.

The probabilistic machine performs the following 11 actions to recognize whether or not the given string $w$ is in $S$:

1) it checks whether the projection of the head of $w$ to the subalphabet $\{0,1,2\}$ is a string of the form

$\text{bin}(1) \ 2 \ \text{bin}(2) \ 2 \ \ldots \ \ 2 \ \text{bin}(2k_1)$

for an integer $k_1$;

2) it checks whether the projection of the tail of $w$ to the subalphabet $\{0,1,2\}$ is a string of the form

$\text{bin}(1) \ 2 \ \text{bin}(2) \ 2 \ \ldots \ \ 2 \ \text{bin}(2k_2)$

for an integer $k_2$;

3) it checks whether $k_1=k_2$;

4) it counts the number $k_3$ of the substrings $\text{bin}(1), \text{bin}(2), \ldots, \text{bin}(k_3)$ in the head of $w$ where no symbol from $\{3,4\}$ are inserted;

5) it checks whether $k_1=k_3$;

6) it counts the number $k_4$ of the substrings $\text{bin}(1), \text{bin}(2), \ldots, \text{bin}(k_4)$ in the tail of $w$ where no symbols from $\{3,4\}$ are inserted;
7) it checks whether \( k_2 = k_4 \);

8) using generator of random numbers it generates a string in \( \{0,1\}^c \) where \( c \) is the constant from Lemma 8.1, and \( 1 \) is the length of \( \text{bin}(2k_3) \). The generated string \( m \) \((1 < m < 2^c)\) is tested for primality. If \( m \) is not prime, a new string \( \text{bin}(m) \) is generated, tested for primality, and so on;

9) it regards the projection \( y \) of the head of \( w \) to the subalphabet \( \{3,4\} \) as binary notation of a number \( N_1 \) \((0 < n_1 < 2 |Y_1| - 1)\) and calculates the remainder \( m_1 \) obtained by dividing \( N_1 \) to \( m \);

10) it regards the projection \( y \) of the tail of \( w \) to the subalphabet \( \{3,4\} \) as binary notation of a number \( N_2 \) \((0 < n_2 < 2 |Y_2| - 1)\) and calculates the remainder \( m_2 \) obtained by dividing \( N_2 \) to \( m \);

11) it checks whether \( m_1 = m_2 \).

The string \( w \) is accepted if all the checks result positively. Otherwise \( w \) is rejected.

The actions 1)-11) can be performed in \( \log w \) space. Lemma 8.1 implies the correctness of the result with the needed probability.

2) If \( w \in S \) then its projection to the subalphabet \( \{3,4,5\} \) is of the form \( y_5 y \), where \( y \in \{3,4\}^* \) and \(|y| > \frac{|w|}{6} - 2 \log_2 |w| \). Hence a linear space bound linear in \(|w|\) for deterministic one-way Turing machines is evident.

The language \( S \) is similar to \( S \) but it is defined in a more complicate way.

Let the strings \( x = x_1 x_2 \ldots x_u \) and \( y = y_1 y_2 \ldots y_v \) in \( \{0,1\}^u \) and \( \{0,1\}^v \), respectively, be considered, and either \( u = v \) or \( u + 1 = v \). We use \( \text{join} (x,y) \) to denote the string \( x_1 y_1 x_2 y_2 \ldots \).

We define a map \( Z : \{0,1\}^* \rightarrow \{0,1,2,3,4,5\}^* \). At first, the given \( w \in \{0,1\}^* \) is transformed into \( w' \), substituting every symbol 0 in \( w \) by 3 and every 1 by 4. We denote the total number of symbols in the strings \( \text{bin}(t+1), \text{bin}(t+2), \ldots, \text{bin}(2t) \) by \( s(t) \). Let \( l \) be an integer such that \( s(l-1) < |w| \leq s(l) \). Then \( Z(w) \) can be obtained from the string

\[ \text{bin}(1) 2 \text{bin}(2) 2 \text{bin}(3) 2 \ldots 2 \text{bin}(2l) \]

inserting symbols from \( \{3,4,5\} \) so that every symbol in the substrings \( \text{bin}(l+1), \text{bin}(l+2), \ldots, \text{bin}(2l) \) is preceded by one symbol from \( \{3,4,5\} \) and the projection of the obtained string to
the subalphabet \{3,4,5\} equals the w'555 ....

The language S consists of all possible strings of the form \(z(\text{join}(\text{bin}(1),\text{bin}(2))) \ 6 \ z(\text{join}(\text{bin}(2),\text{bin}(3))) \ 6 \ z(\text{join}(\text{bin}(3),\text{bin}(4))) \ 6 \ ... \ 6 \ z(\text{join}(\text{bin}(2k-1),\text{bin}(2k)))\).

**Theorem 8.2.** (1) Given any \(\epsilon>0\), there is a loglogn-space bounded probabilistic 1-way Turing machine which accepts every string in S with probability 1 and rejects every string in S with probability 1-\(\epsilon\). (2) Every deterministic 1-way Turing machine recognizing S uses at least const \(\log n\) space.

**Proof** is similar to the proof of Theorem 8.1. The main additional idea in the proof of (1) is to perform all the needed (probabilistic) comparisons whether the substrings bin(i) correspond one to another in the fragment
\[\ldots \ 6 \ z(\text{join}(\text{bin}(i-1),\text{bin}(i))) \ 6 \ z(\text{join}(\text{bin}(i),\text{bin}(i+1))) \ 6 \ ...\]
independently, i.e. by using another choice of a random modulo. If the given string is in S then all the many comparisons end in positive with probability 1. If there is at least one discrepancy then the comparisons end in negative with probability 1-\(\epsilon\).

It is possible to extend Theorems 8.1 and 8.2 for "natural" space complexities \(f(n)\) between \(\log n\) and loglogn. On the other hand, the method used above does not permit to construct nonregular languages recognizable by probabilistic 1-way Turing machines in \(o(\log \log n)\) time. Now we proceed to prove Theorem 8.3 which shows that if a language L is recognized by probabilistic 1-way Turing machine in \(o(\log \log n)\) space then L is regular. The result may seem trivial since Trakhtenbrot [Tra 74] and Gill [Gi 74] have proved theorems showing that determinization of probabilistic Turing machines increase space complexity no more than exponentially, and it is known from [SHL 65] that if a language L is recognized by a deterministic 1-way Turing machine in \(o(\log n)\) space then L is regular. Unfortunately, the situation is more complicated since no function \(o(\log n)\) is space constructible by deterministic 1-way Turing machines. Hence the argument by Trakhtenbrot and Gill is not applicable. Our proof is nonconstructive. We do not present an algorithm for determinization of probabilistic 1-way Turing machines.

Theorem 8.2 gave an example of a nonregular language recognizable in \(o(\log \log n)\) space by a probabilistic one-way Turing machine.
On the other hand, as proved in [SHL 65], no deterministic one-way Turing machine can recognize nonregular languages in \( o(\log n) \) space. R. Freivalds [Fre 83] proved that recognition of a language in \( o(\log \log n) \) space by a probabilistic one-way Turing machine with probability \( 2/3 \) implies regularity of the language.

A modification of this theorem says that regularity of a language is implied as well its recognition with any probability \( p>1/2 \) by a probabilistic one-way Turing machine which never exceeds a space bound \( S(n)=o(\log \log n) \) whatever random options are taken by the probabilistic Turing machine.

It was formulated explicitly in [Fre 83] as an open problem, to eliminate the abovementioned restriction (either \( p=2/3 \) or space bound \( S(n)=o(\log \log n) \) for all random options). In spite of many attempts, this problem turned out to be very hard. We solve it only [KF 90] by considering a notion of \( n \)-similar pairs of words and proving the crucial Lemma 2 which, we believe, may be of some interest itself.

It is interesting to note that for two-way machines there is no minimal nontrivial space complexity such that capabilities of probabilistic and deterministic machines differ only starting from this complexity. Theorem 9.1 below shows that there is a nonregular language which can be recognized by probabilistic two-way finite automata with arbitrary probability \( 1-c \) (\( c>0 \)).

We remind the reduction theorem by M.O. Rabin [Rab 63]. Let \( X \) be a finite set, and \( L\subseteq X^* \) be a language. The words \( w', w^*\in X^* \) are called equivalent with respect to the language \( L \) if

\[
(\forall w^*\in X^*) (w'w\in L) \iff (w^*w\in L).
\]

By weight \( (L) \) we denote the number of the classes of equivalence with respect to the language \( L \) (language \( L \) is regular if weight \( (L)\leq \infty \)).

M.O. Rabin proved the following theorem. If a language \( L \) is recognized by a finite probabilistic 1-way automaton with \( k \) states with probability \( 1/2+\delta \) then weight \( (L)\leq (1+1/\delta)^{k-1} \).

(In fact, M.O. Rabin formulated his theorem in a slightly more general form. Any case, it follows from this theorem that finite probabilistic automata with isolated cut-point recognize only regular languages).

Let \( X \) be a finite set, \( L\subseteq X^* \) be a language and \( n=0 \) be an
integer. The words \( w', w'' \in X^* \) are called \( n \)-similar with respect to the language \( L \) if

\[
(\forall w \in X^*)(w' \in X^* \land w'' \in X^*) \Rightarrow (w' \in L \leftrightarrow w'' \in L).
\]

To denote this relation we use

\[
w' \w''(L, X^*).
\]

Rank \( r_{\text{sim}}(L, X^*) \) is the cardinality of the maximal subset of \( X^* \) such that all words in the subset are pairwise non-\( n \)-similar with respect to \( L \).

We say that a probabilistic automaton recognizes the initial \( n \)-fragment of \( L \) with probability \( 1/2 + \delta \) if the automaton accepts every word in \( X^* \cap L \) with probability no less than \( 1/2 + \delta \) and accepts every word in \( X^* \backslash L \) with probability not exceeding \( 1/2 - \delta \).

The proof of Rabin's theorem proves the following lemma as well.

**Lemma 8.2.** If a finite probabilistic automaton with \( k \) states recognizes the initial \( n \)-fragment of \( L \cap X^* \) with probability \( 1/2 + \delta \) \((\delta > 0)\) then

\[
r_{\text{sim}}(L, X^*) \leq (1 + 1/\delta)^k - 1.
\]

Below we prove the crucial technical lemma.

**Lemma 8.3.** If a language \( L \subseteq X^* \) is nonregular then for infinitely many \( n \)

\[
r_{\text{sim}}(L, X^*) \geq \left\lceil (n+3)/2 \right\rceil.
\]

Rather many textbooks on automata and formal language theory contain the following definition.

Let \( X \) be a finite set and \( L \subseteq X^* \) be a language. The words \( w', w'' \in X^* \) are called equivalent with respect to \( L \) if

\[
(\forall w \in X^*)(w' \in L \leftrightarrow w'' \in L).
\]

We denote this equivalence by \( w' \w''(L) \). For every \( n \in \mathbb{N} \) the relation \( (L) \) divides \( X^* \) into a certain number of classes of the equivalence. The number of these classes is denoted by \( r_{\text{reach}}(L, X^*) \) and called the rank of \( n \)-reachability of the language \( L \).

For arbitrary \( n \in \mathbb{N} \) the words \( w', w'' \in X^* \) are called \( n \)-indistinguishable with respect to \( L \) if

\[
(\forall w \in X^*)(w' \in L \leftrightarrow w'' \in L).
\]

This property is denoted by \( w' \w''(L, X^*) \).

The relation \( (L, X^*) \) again is an equivalence type relation, and the rank of \( n \)-indistinguishability \( r_{\text{indist}}(L, X^*) \) is the
number of the equivalence classes of $X^*$ for this relation.

Note that the relation of $n$-similarity (see Section 1 for the definition) is not an equivalence type relation. Nevertheless this relation has useful properties rather close to equivalence type relations:

\[(\forall w \in X^\leq n)(w w(L,X^\leq n)), \quad (8.1)\]

\[(\forall w', w'' \in X^\leq n)(w' w''(L,X^\leq n) \Rightarrow w'' w'(L,X^\leq n)) \quad (8.2)\]

\[<n \leq n \leq n (\forall w', w'' \in X^\leq n)(|w| \leq \max\{|w'|, |w''\}\& w w'(L,X^\leq n)\&
\& w'' w'(L,X^\leq n) \Rightarrow (w' w''(L,X^\leq n)), \quad (8.3)\]

\[(\forall w, w', w'' \in X^\leq n)(w' w''(L,X^\leq n)\& |w' w''| \leq n\&\]
\& |w'' w w''(L,X^\leq n)) \Rightarrow (w' w''(L,X^\leq n)). \quad (8.4)\]

We consider the functions $n \rightarrow r_{reach}(L,X^\leq n), n \rightarrow r_{indist}(L,X^\leq n), n \rightarrow r_{sim}(L,X^\leq n)$ in this Section. Note that these functions are nondecreasing.

**EXAMPLE 8.1.** Prefix of a word is a subword containing the first symbol of the given word (or it is the empty subword). We consider a language $L \subseteq \{0,1\}^*$ defined by the following property. All words in $L$ contain a prefix in which the number of zeros strictly exceeds the number of ones. Fig.1 shows the diagram of a finite deterministic one-way automaton recognizing the initial $n$-fragment of $L$. The initial state of the automaton is $q_1$, the only accepting state is $q_{\text{accept}}$. The total number of the states is $1+\lfloor(n+2)/2\rfloor=\lfloor(n+4)/2\rfloor$.

![Diagram](image)

Fig. 1.

The subsequent Lemma shows that for the considered language $L$ the inequality $r_{sim}(L,\{0,1\}^\leq n) \leq \lfloor(n+4)/2\rfloor$ holds for arbitrary $n$. 

LEMMA 8.4. Let $X$ be a finite set, $L \subseteq X^*$ be a language, $n \in \mathbb{N}$. The rank $r_{\text{sim}}(L, X^n)$ equals the number of states of the minimal finite deterministic one-way automaton recognizing the initial $n$-fragment of $L$.

PROOF. We consider the case $n=0$ separately. Evidently, $r_{\text{sim}}(L, X^n) = 1$. On the other hand, the minimal automaton recognizing the initial 0-fragment (either containing $e$ or not containing it) needs no more than one state. Hence, Lemma holds for $n=0$.

Let $n>0$. Since $L$ and $n$ are fixed for the rest of the proof, we replace $w' w''(L, X^n)$ by $w' w''$. We call a word $w \in X^n$, being short if there exists no word $w' \in X^n$ such that $|w'| < |w|$ and $w' w$. We denote $r_{\text{sim}}(L, X^n)$ by $r$. We take $r$ pairwise non-$n$-similar words $w_1, w_2, \ldots, w_r$ from $X^n$ (Their existence is implied by the definition of $r$). For every $i \in \{1, \ldots, r\}$ we fix a word $w_i'$ as the shortest word with the property $w_i' w_i$.

(If several words of the same minimum length exist, we take one of them). Clearly $|w_i'| \leq |w_i|$.

Now we prove that $w_i'$ is short. (Had been equivalence type relation, this would be trivial). Assume the contrary.

Then for some $w'' \in X^n$ there holds $|w''| < |w_i'|$ and $w'' w_i'$. Since $|w_i'| \leq \max\{|w_i'|, |w''|\}$, $w_i' w_i$, $w_i' w''$, it follows from (3) that $w_i' w_i'$. Contradiction, since $w_i'$ is the shortest among the words $n$-similar to $w_i$. Hence, all the words $w_i'$ ($i=1, 2, \ldots, r$) are short.

Now we prove that for distinct $i, j$ the words $w_i'$ and $w_j'$ are non-$n$-similar. Assume the contrary, namely, assume $w_i' = w_j'$. From this and $|w_i'| \leq \max\{|w_i'|, |w_j'|\}$, and (3) it follows $w_i' w_i'$. Now we make the same type conclusion from $w_j' w_i$, $|w_j'| \leq \max\{|w_i'|, |w_j'|\}$, and (3). We get $w_i' w_j$. Contradiction.

Thus all the words $w_1', w_2', \ldots, w_r'$ are short and pairwise non-$n$-similar with respect to $L$. Assume from the contrary that there is a finite deterministic one-way automaton with less than $r$ states recognizing the initial $n$-fragment of $L$. Then there are distinct $i, j$ such that $w_i'$ and $w_j'$ move the automaton from the initial state to the same current state. But then the automaton cannot distinguish between $w_i'$ and $w_j'$. Hence for arbitrary $w \in X^*$ either both $w_i' w$ and $w_j' w$ are accepted or the two words are both rejected. We have

$$(\forall w \in X^*)(w_i' w \in X^n \& w_j' w \in X^n) \Rightarrow (w_i' w \in L \lor w_j' w \in L).$$
Hence $w_i' = w_j'$. Contradiction.

To conclude the proof it remains to construct a finite deterministic one-way automaton with $r$ states for recognition of the initial $n$-fragment of $L$. Since $\{w_1', w_2', \ldots, w_r'\}$ is a maximal set of pairwise non-$n$-similar words, for arbitrary $w \in X^{\leq n}$ there is an $i$ such that $w = w_i'$. Hence, for some $i$, it holds $e = w_i'$.

From the shortness of $w'$ if follows $w_i' = e$. For the sake of brevity, we assume $w_i' = e$.

We define the map $\varphi: \{q_1, q_2, \ldots, q_r\} \times X \to \{q_1, q_2, \ldots, q_r\}$ as follows. If $i \in \{1, 2, \ldots, r\}$, $|w_i'| \leq n-1$, $v \in X$ then $\varphi(q_i, v)$ is defined to be $q_j$ such that $w_i'v = w_j'$. (Such a value $j$ exists because $w_i'v \in X^{\leq n}$). If several distinct $j$ with this property are found, we take one of them. For those pairs $(i, v)$ where $|w_i'| = n$ the value $\varphi(q_i, v)$ is not really needed and we define, for example, $\varphi(q_i, v) = q_i$.

We consider the automaton $A$ with the set of states $Q = \{q_1', \ldots, q_r'\}$, the initial state $q_1'$, the set of accepting states $S = \{q_i | w_i' \in L\}$ and the map $\varphi$.

We extend the map $\varphi$ to the domain $Q \times X^*$ in the standard way. Let $\varphi(q_i, w)$ be used to denote the state of $A$ into which $A$ is moved after reading $w$, provided it has been in $q_i$ initially. By induction over the length of $w$ we prove

$$\forall w \in X^{\leq n}, \varphi(q_1, w) = q_1 \Leftrightarrow w = w_i'.$$ (8.5)

Since $\varphi(q_1, e) = q_1$, $e = w_i'$, then for $w = e$ the property (8.5) holds. Let (8.5) hold for the words of length $k$ ($k < n$). We will prove (8.5) for arbitrary word $w \in X^{\leq n}$ where $w \in X^k$, $v \in X$. Let $\varphi(q_1, w) = q_i$. By the assertion of induction, $w = w_i'$. Since the word $w_i'$ is short, $|w_i'| \leq |w| = k < n$. Hence, $w_i'v \in X^{\leq n}$. It follows from (8.4) that $wv = w_i'v$. Let $\varphi(q_i, w) = q_j$, i.e. $w_i'v = w_j'$. Since $w_i'v = wv$, $w_i'v = w_j'v$, $|w_i'v| \leq \max\{|wv|, |w_j'|\}$, it follows from (8.3) that $wv = w_j'$. Taking into consideration $\varphi(q_1, wv) = \varphi(\varphi(q_1, w), v) = \varphi(q_i, v) = q_j$ we have the property (8.5) for the word $wv$. This concludes the proof of (8.5).

Let $w \in X^{\leq n}$ and $\varphi(q_1, w) = q_j$. It follows from (8.5) that $w = w_j'$ and this implies $(w \in L) \Leftrightarrow (w_j' \in L)$. Hence, $(q_j \in S = \{q_i | w_i' \in L\}) \Leftrightarrow (w_j' \in L) \Leftrightarrow (w \in L)$. Hence, the automaton $A$ recognizes the initial $n$-fragment of the language $L$.

**Lemma 8.5.** If a language $L \subseteq X^*$ is nonregular then

$$\lim_{n \to \infty} \sim_r (L, X^{\leq n}) = \infty.$$
PROOF. If \( L \) is nonregular then there is an infinite set of words pairwise nonequivalent with respect to \( L \). Let \( c \) be an arbitrary positive integer. Let \( w_1, w_2, \ldots, w_c \in X^* \) be pairwise nonequivalent, i.e. \( 1 \leq i < j \leq c \Rightarrow w_i \neq w_j(L) \). For every pair \( w', w'' \in X^* \) such that \( w' \neq w''(L) \) there is a \( n_0 \) such that \(( \forall n \geq n_0 ) w' \neq w''(L, X^n) \). We take \( n_0 \) to be large enough to ensure for all \( n > n_0 \), \( 1 \leq i < j \leq c \) the property \( w_i \neq w_j(L, X^n) \). Then for \( n > n_0 \) it holds \( r_{sim}(L, X^n) \geq c \). It implies \( \lim \limits_{n \to \infty} r_{sim}(L, X^n) = \infty \).

Now we are ready to prove the crucial Lemma 8.3.

PROOF OF LEMMA 8.3. Assume the contrary. Then there is \( n_0 > 0 \) such that for all integers \( n > n_0 \) it holds \( r_{sim}(L, X^n) \leq \lceil (n+3)/2 \rceil - 1 = \lceil (n+1)/2 \rceil \). By Lemma 8.5 \( \lim \limits_{n \to \infty} r_{sim}(L, X^n) = \infty \).

Hence there is a \( m > n_0 \) such that \( r_{sim}(L, X^{m+1}) > r_{sim}(L, X^m) \).

Let \( A_m \) and \( A_{m+1} \) be the minimal finite deterministic one-way automata recognizing, respectively, the initial \( m \)- and \( m+1 \)-fragments of the language \( L \). By Lemma 8.4 they have, respectively \( r_{sim}(L, X^m) \) and \( r_{sim}(L, X^{m+1}) \) states. Since \( r_{sim}(L, X^m) < r_{sim}(L, X^{m+1}) \), it follows that \( A_m \) recognizes the initial \( m \)-fragment of \( L \), but not the initial \( m+1 \)-fragment. Hence, the shortest word accepted by one of the automata \( A_m, A_{m+1} \) but rejected by the other one, is of the length \( m+1 \). In another words, \( A_m \) and \( A_{m+1} \) are not equivalent but they are indistinguishable on the \( X^m \). Two nonequivalent automata can be distinguished by a word of length no more than the sum of the numbers of the states minus one ([KAP 85], Theorem 2.10). Hence, \( m+1 \leq r_{sim}(L, X^m) + r_{sim}(L, X^{m+1}), \) i.e.

\[
r_{sim}(L, X^m) + r_{sim}(L, X^{m+1}) \geq m+2.
\]

This contradicts our assumption.

\[
r_{sim}(L, X^m) \leq \lceil (m+1)/2 \rceil, \quad r_{sim}(L, X^{m+1}) \leq \lceil (m+2)/2 \rceil
\]

Example 1 shows that the bound \( \lceil (n+3)/2 \rceil \) is nearly optimal. Note that for \( r_{reach}(L, X^n) \) and \( r_{indist}(L, X^n) \) bounds are quite different: if \( L \) is nonregular then for all \( n \), \( r_{reach}(L, X^n) \geq n+1 \), and \( r_{indist}(L, X^n) \geq n+1 \) (Corollaries of Theorem 2.13 in [TB 72]).
THEOREM 8.3. If a language $L$ is recognized by a probabilistic one-way Turing machine with probability $1/2+\delta$ ($\delta>0$) in $g(n)=o(\log \log n)$ space then $L$ is regular.

PROOF. Let the input alphabet of the probabilistic machine $M$ be $X$, the work-tape alphabet be $Y$ and the set of states be $Q=\{q_1, \ldots, q_k\} \cup \{q_{\text{accept}}, q_{\text{reject}}\}$, where $q_1$ is the initial state. Without loss of generality we assume that $M$ can enter the states $q_{\text{accept}}$ or $q_{\text{reject}}$ only when the head on the input tape observes the symbol $\#\in X$.

Configuration of $M$ at any moment (but the final moment) is triple $(q,u,l)$ where $q\in Q\setminus\{q_{\text{accept}}, q_{\text{reject}}\}$ is the current state, $u\in Y^*$ is the content of the work-tape, $l\in\{1,2,\ldots,|u|\}$ is the current position of the head on the work-tape. Configuration of $M$ at the final moment consists of the state $1_{\text{accept}}$ or $q_{\text{reject}}$ only.

For arbitrary $n\in\mathbb{N}$ we denote by $C_n$ the set of all configurations of the type $(Q\setminus\{q_{\text{accept}}, q_{\text{reject}}\}) \times Y^{g(n)} \times \{1,\ldots,g(n)\}$.

It follows from the properties of $M$ that for arbitrary $w\in X^n \setminus L$ (respectively, for arbitrary $w\in X^n \setminus L$) the probability of the following event is no less than $1/2+\delta$ (respectively, does not exceed $1/2-\delta$): all the configurations during the processing of $w$ by $M$ (but the final moment) belong to $C_n$ and $M$ ends in $q_{\text{accept}}$. We denote the probability of this event by $h_n,w$.

Let the head on the input tape observe arbitrary symbol $x\in X$ and the current configuration be $(q',u',l') \in C_n$. We associate with this moment a period of work of $M$ onwards from this moment until one of the two events take place:

1) the head on the input tape moves right;
2) $M$ enters a configuration outside $C_n$.

We denote by $p_n(x,(q',u',l'),\infty)$ the probability of the associated period lasting infinitely long time. By

$p_n(x,(q',u',l'),\text{full})$

we denote the probability of the associated period ending in a configuration outside $C_n$ (i.e. in a configuration $(q'',u'',l'')$ where $|u''|>g(n)$). For arbitrary triple $(q'',u'',l'') \in C_n$ we denote by

$p_n(x,(q',u',l'),(q'',u'',l''))$
the probability of the associated period ending in moving the head on the input tape, and in this moment \( M \) finding itself in the configuration \((q'',u'',l'')\).

Now we consider the associated period for arbitrary configuration \((q',u',l')\in C_n\) where the head on the input tape observes \#. In this case we associate a period of work of \( M \) until one of the events take place:

1) the input word is accepted or rejected;
2) \( M \) enters a configuration \((q'',u'',l'')\) where \(|u''|>g(n)\).

We denote by

\[ p_n(\#, (q',u',l'), \infty) \]

the probability of the associated period lasting infinitely long time. By

\[ p_n(\#, (q',u',l'), \text{full}) \]

we denote the probability of the associated period ending in a triple \((q'',u'',l'')\) where \(|u''|>g(n)\). By

\[ p_n(\#, (q',u',l'), q_{\text{accept}}), \]
\[ p_n(\#, (q',u',l'), q_{\text{reject}}) \]

we denote the probabilities of the acceptance and rejection, respectively, at the end of the associated period.

Additionally for arbitrary \( z\in X\cup\{\#\} \) we define

\[ p_n(z, q_{\text{accept}}, \text{stop}) = p_n(z, q_{\text{reject}}, \text{stop}) = 1, \]
\[ p_n(z, \infty, \infty) = p_n(z, \text{full}, \text{full}) = p_n(z, \text{stop}, \text{stop}) = 1 \]

Let \( n \) be arbitrary positive integer. Consider finite probabilistic one-way automaton \( \mathcal{A}_n \) in alphabet \( X\cup\{\#\} \) with the set of states

\[ S_n = C_n \cup \{\infty, \text{full}, q_{\text{accept}}, q_{\text{reject}}, \text{stop}\}, \]

the initial state \((q_1, \Lambda, 1)\) where \( \Lambda \) is the empty symbol \( q_{\text{accept}} \) is the only accepting state. The automaton \( \mathcal{A}_n \) works as follows. When \( \mathcal{A}_n \) in the state \( s'\in S_n \) reads from the input an arbitrary symbol \( z\in X\cup\{\#\} \), the automaton \( \mathcal{A}_n \) moves to the state \( s''\in S_n \) with the probability \( p_n(z, s', s'') \). It is easy to see that the automaton \( \mathcal{A}_n \) accepts arbitrary word \( w\# \), where \( w\in X^{s_n} \), with probability \( h_{n,w} \). Every word not of the form \( w\# \), where \( w\in X^* \), is rejected by \( \mathcal{A}_n \).

Since

\[ (w\in X^{s_n}\setminus L) \Rightarrow h_{n,w} \geq 1/2+\delta, \]
\[ (w\in X^{s_n}\setminus \{\#\}) \Rightarrow h_{n,w} \leq 1/2-\delta, \]

the automaton \( \mathcal{A}_n \) recognizes the initial \((n+1)\)-fragment of the language \( L\# = \{w\# | w\in L\} \) with probability \( 1/2+\delta \). Hence by Lemma 8.2,
for arbitrary \( n \in \mathbb{N} \) it holds
\[
\text{rsim}(L_\#, (X \cup \{\#\})^{n+1}) \leq (1 + 1/\delta) S_n.
\]
On the other hand, when \( n \to \infty \) we have \( g(n) = o(\log \log n) \),
\[
|S_n| \leq |Q| \cdot |\{q_{\text{accept}}, q_{\text{reject}}\}| \cdot |Y^{g(n)}| g(n) + 5 = o(\log n),
\]
\[
(1 + 1/\delta) S_n = o(n).\)
Hence, for sufficiently large \( n \)
\[
\text{rsim}(L_\#, (X \cup \{\#\})^{n+1}) \leq (1 + 1/\delta) S_n < n/2.
\]
It follows from Lemma 8.3 that the language \( L_\# \) is regular.
Hence \( L \) is as well regular.

**9. SPACE COMPLEXITY OF 2-WAY TURING MACHINES**

M.O. Rabin [Rab 57] and J.C. Shepherdson [She 59] considered deterministic 2-way finite automata (2-FA) and proved that they recognize only regular languages. Nondeterministic and even alternating 2-FA also accept only regular languages [LLS 78]. However we prove below that for arbitrary \( \varepsilon > 0 \) the nonregular language \( \{0^n1^n\} \) can be recognized by a probabilistic 2-FA with probability \( 1 - \varepsilon \). Had the recognizability of this language been proved by the method of invariants, we would have also a nondeterministic 2-FA recognizing \( \{0^n1^n\} \) but nondeterministic 2-FA recognize only regular languages. The same reason causes a positive probability of error for strings in the complement of the language.

**THEOREM 9.1.** For arbitrary \( \varepsilon > 0 \) there is a probabilistic 2-way finite automaton recognizing the language \( A = \{0^n1^n\} \) with probability \( 1 - \varepsilon \).

**PROOF.** Let \( c(\varepsilon) \) and \( d(\varepsilon) \) be large natural numbers such that
\[
2^{\left(\frac{1}{2}\right)^d(\varepsilon)} < c, \quad \left(\frac{2^{c(\varepsilon)}}{1 + 2^{c(\varepsilon)}}\right)^{d(\varepsilon)} > 1 - \varepsilon.
\]
Let the input string \( x \) be of the form \( 0^n1^m \). The automaton processes alternately the block of zeros and the block of ones. One processing of a block is a series of options when \( c(\varepsilon) \) random symbols 0 or 1 are produced per every letter in the block. We call the processing to be positive if all the results are 1, and
If the length of the block is \( n \), then the probability of a positive processing of it is \( 2^{-n} c(c) \).

We interpret a processing of an ordered pair of blocks as a competition. A competition where one processing is positive and the other is negative, is interpreted as a win of the block processed positively.

To recognize the language the automaton holds competitions until the total number of wins reaches \( d(c) \). If at this moment the two blocks have at least one win each, then \( x \) is accepted, otherwise it is rejected.

If the competitions are held unrestrictedly, then one of the blocks wins with probability 1. If \( n=m \) then the probability of the win by the shortest block relates to the probability of the win by the longest block at least as \( 2^{c(c)} : 1 \). Our choice of \( c(c) \) and \( d(c) \) ensures that the probability of error does not exceed \( c \) both in the case \( n=m \) and in the case \( n \neq m \).

Now we consider a more complicate language being the Kleene star of the language \( \{0^n 1^n\} \).

\[
A^* = \{0^n 1^n 0^n 1^n \ldots 0^n 1^n | k=0,1,2,\ldots; n_1,\ldots,n_k=1,2,\ldots \}
\]

**THEOREM 9.2.** For arbitrary \( c>0 \) there is a probabilistic 2-way finite automaton recognizing the language \( A^* \) with probability \( 1-c \).

**PROOF.** The basic difficulty arises from the following obstacle. The algorithm in the proof of Theorem 9.1 yields the right answer with a guaranteed probability 1 neither for strings in \( A \) nor for strings in \( \overline{A} \). The number \( k \) can be large, and therefore recognition of each fragment \( 0^n 1^n \) with a high fixed probability does not suffice to obtain the right answer about whether the string is in \( A^* \) or not with a high enough probability.

Let \( \delta \) be a real number \((0<\delta<1)\), and \( d(c) \) a natural number such that

\[
2 \left( \frac{1}{2} \right)^{d(c)} < \epsilon, \\
(1-\delta)^{d(c)} > 1-\epsilon.
\]

Let \( A(\delta) \) be the automaton recognizing the language \( A \) from
the proof of Theorem 9.1. We shall use $\mathcal{A}(\delta)$ as a part of the automaton to be constructed. We describe the performance of our new automaton on a string of the form
\[ n_1 m_1 n_2 m_2 \ldots n_k m_k \]  
(9.1)

The main idea of the proof reminds the idea of the proof of Theorem 9.1, and consists in organizing "competitions" (in the sense of that proof) between the string (9.1) and the string
\[ n_1 n_1 n_2 n_2 \ldots n_k n_k \]  
(9.2)

Macroprocessing of the string (9.1) (or 9.2) is a series of applications of the algorithm $\mathcal{A}(\delta)$ to each fragment $0^i 1^m$ of the string (or to fragment $0^i n_i$). The macroprocessing is positive if $\mathcal{A}(\delta)$ has accepted all the fragments $0^i m_i$.

Macrocompetition is a pair of macroprocessings: each of the strings (9.1) and (9.2) is processed once. A macrocompetition where one string is processed positively and the other is processed negatively is counted as a win for the positively processed string. The macrocompetitions are repeated until the total number of wins reaches $d(c)$.

Let $a_i$ denote (for $i=1,2,\ldots,k$) the probability with which the automaton $\mathcal{A}(\delta)$ accepts the string $0^i n_i m_i$. The similar probability for $0^i n_i n_i$ is denoted by $b_i$. The probability of a positive macroprocessing of strings (9.1) and (9.2), is $a_1 a_2 \ldots a_k$ and $b_1 b_2 \ldots b_k$, respectively. It is important for us that, if $n_i = m_i$, then $a_i = b_i \geq 1 - \delta$, and if $n_i \neq m_i$, then $a_i < \delta$ and $b_i \geq 1 - \delta$. Hence, if $n_i = m_i$ for all $1 \leq i \leq k$, the probabilities of positive macroprocessing for both strings are equal. If $n_i \neq m_i$ for at least one $i$, then
\[ \frac{a_1 a_2 \ldots a_k}{b_1 b_2 \ldots b_k} < \frac{\delta}{1 - \delta} \]

10. AUTOMATA ON TREES

The method of the proof of Theorem 9.2 is rather powerful. It can be used for more complicated types of automata where capabilities of probabilistic automata proved by this method are a bit unexpected.
We consider 3-way automata walking on binary trees. In contrast to other definitions of automata on trees, there is only one copy of the automaton, and no duplication of the automaton is allowed. The automaton may choose among 3 possible direction: up, down-to-the-left, down-to-the-right. The automaton starts at the root of the binary tree (which is the uppermost vertice of the tree). The tree is finite. Every leaf is at a finite distance from the root. The automaton can walk as much time as it wishes. When the automaton decides to stop, it produces the output value "yes" or "no". Such an automaton can be used to decide properties of finite binary trees (languages of trees).

We denote by $D$ the following language. It consists of all uniform finite binary trees, i.e. all the leaves are at the same distance from the root. It is easy to see that no such deterministic finite automaton can decide uniformity of trees. In contrast to this we have the following theorem.

**THEOREM 10.1.** For arbitrary $\varepsilon > 0$ there is a probabilistic finite automaton walking on finite binary trees which recognizes the language $B$ with probability $1-\varepsilon$.

**PROOF.** Finite binary tree is uniform if and only if for arbitrary vertice $a$ of the tree the lengths of the two following paths are equal: 1) the path starting from $a$ and going, first, down-to-the-right, and then down-to-the-left, down-to-the-left, down-to-the-left,..., 2) the path starting from $a$ and going, first, down-to-the-left, and then down-to-the-right, down-to-the-right,...

The essence of the probabilistic algorithm completely coincides with that in the proof of Theorem 9.2.

**REFERENCES**


