SIAM J. COMPUT.
Vol. 6, No. 4, December 1977

COMPUTATIONAL COMPLEXITY OF PROBABILISTIC TURING MACHINES*

JOHN GILL†

Abstract. A probabilistic Turing machine is a Turing machine with the ability to make decisions based on the outcomes of unbiased coin tosses. The partial function computed by a probabilistic machine is defined by assigning to each input the output which occurs with probability greater than $\frac{1}{2}$. With this definition, only partial recursive functions are probabilistically computable. The run time and tape of probabilistic machines are defined. A palindrome-like language is described that can be recognized faster by one-tape probabilistic Turing machines than by one-tape deterministic Turing machines. It is shown that every nondeterministic machine can be simulated in the same space by a probabilistic machine with small error probability. Several classes of languages recognized probabilistically in polynomial time are defined and compared with *NP*.

Key words. probabilistic Turing machines, probabilistic algorithms, probabilistic computation, computational complexity, Turing machines, Markov chains, crossing sequences, polynomial reducibility, polynomial-complete languages

1. Introduction.

Probabilistic methods in computation have usually been used for problems arising from probabilistic considerations [5]. Monte Carlo methods are often used to obtain qualitative information about the behavior of large systems, especially for systems subject to random disturbances. As another example, Monte Carlo integration techniques are most applicable to multidimensional integrals, which typically are the probabilities of events that are complex combinations of simpler events.

Recently certain problems have been shown to be solvable by probabilistic algorithms that are faster than the known deterministic algorithms for solving these problems. Rabin [20] describes a probabilistic algorithm for finding the nearest pair of a set of n points in k-space that executes in average time $O(n)$, which is faster than the $n \log n$ steps required by the best known deterministic algorithms. Solovay and Strassen [26] and Rabin [20] present probabilistic algorithms for recognizing prime numbers in polynomial time with a small probability of error. These results suggest that probabilistic algorithms may be useful for solving other deterministically intractable problems.

In this paper we study a formal model for probabilistic algorithms, the probabilistic Turing machine. A probabilistic Turing machine is a computer with the ability to make random decisions. The output of a probabilistic machine is not in general uniquely determined by the input, but we can define in a natural way the partial function computed by a probabilistic machine (Definition 2.1). It has been shown that the ability to make random decisions does not increase the ultimate computational power of Turing machines [4], [23], [8]. However, it is natural to ask if probabilistic machines can be proven to require less time or tape than deterministic machines. This paper provides a partial answer to this question.

In § 2 we define probabilistic Turing machines (PTMs) and the functions computed by probabilistic machines. We show that every probabilistically computable function is partial recursive, and in fact that any standard enumeration of

* Received by the editors June 1, 1976, and in revised form February 18, 1977.

† Department of Electrical Engineering, Stanford University, Stanford, California 94305. This research was supported in part by National Science Foundation under Grant GK-43121.

PTMs yields an acceptable Gödel numbering of the partial recursive functions. We discuss briefly the relationship between our definitions of PTMs and the more general notion of Santos [23].

In § 3 we show that maximum and average run time are unsuitable as complexity measures for the entire class of PTMs. We introduce a more convenient concept of probabilistic run time, and show that every probabilistic machine can be simulated deterministically in at most exponentially more time.

In § 4 we exhibit a palindrome-like language that can be recognized by a fixed one-tape probabilistic Turing machine faster for infinitely many inputs than by any one-tape deterministic Turing machine.

In § 5 we prove elementary facts about several classes of languages recognizable probabilistically in polynomial time. Simon [25] has shown that the largest of these classes, *PP*, contains *NP* and is identical with the class of languages accepted by polynomial bounded threshold machines. The set of formulas of propositional calculus satisfied by a majority of interpretations is proven to be a polynomial complete language for *PP*.

In § 6 we define tape for probabilistic machines and prove that it is a Blum complexity measure. We show that the output length of a PTM is at most an exponential of the tape; that the run time is at most a double exponential of the tape; that every PTM can be simulated deterministically in at most exponentially more tape; and that every language accepted by a nondeterministic machine can be recognized probabilistically using the same amount of tape.

2. Probabilistic Turing machines.

The model of probabilistic computation studied in this paper, the probabilistic Turing machine, is obtained from the usual Turing machine model by allowing machines access to the simplest type of randomness. A probabilistic Turing machine is a coin-tossing computer which can make decisions based on the outcomes of fair coin tosses.

We assume a standard Turing machine model [11]. A *multitape Turing machine* consists of a finite control unit equipped with a read-only input tape, a write-only output tape, and a finite number of read-write worktapes. A Turing machine is *deterministic* if the current state of the machine uniquely determines the next action of the machine; otherwise the machine is *nondeterministic*.

DEFINITION 2.1. A *probabilistic Turing machine* (PTM) is a Turing machine with distinguished states called coin-tossing states. For each coin-tossing state, the finite control unit specifies two possible next states. The computation of a probabilistic Turing machine is deterministic except that in coin-tossing states the machine tosses an unbiased coin to decide between the two possible next states.

Formal definitions of PTMs in terms of state-transition functions are given in [8] and [23] and are omitted here.

The probabilistic Turing machine model can be extended by relaxing the requirement that unbiased random decisions are made. It can be shown that the resulting model has the same computational power as the PTM defined by Santos [23]. At the end of this section, we state a result characterizing the functions computable by PTMs with unrestricted coin biases.

The computation of a PTM is determined by its input and the outcomes of the coin tosses performed by the machine. It will be convenient to describe probabilistic machines in terms of partial functions of two variables. One variable is the usual input to the machine. The other variable is the random input, a binary sequence representing the outcomes of the coin tosses. For simplicity we suppose that the machine tosses a coin at each step of its computation, although the result of the coin toss is ignored except when the machine is in a coin-tossing state. The possible computations of a probabilistic machine M_i with tape alphabet Σ are represented by a partial function $\phi_i: \Sigma^* \times \{0, 1\}^* \to \Sigma^*$. The value of $\phi_i(x; \alpha)$ is the output of the computation of M_i on input x with coin tosses specified by the binary sequence α. By convention, $\phi_i(x; \alpha)$ is undefined if the computation requires more than $|\alpha|$ steps, where $|\alpha|$ denotes the length of α.

In the remainder of this paper $M_i(x)$ will denote either the possible computations of M_i with input x (a random process) or the output of the computation (a random variable). We assume a standard enumeration $\{M_i\}$ of the probabilistic Turing machines.

In general, a PTM computes a random function [23]; for each input x, the machine M_i produces output y with probability $\Pr\{M_i(x)=y\}$. We define the deterministic partial function computed by any PTM as follows.

Definition 2.2. The partial function ϕ_i computed by probabilistic Turing machine M_i is defined by

$$\phi_i(x)=\begin{cases} y & \text{if } \Pr\{M_i(x)=y\}>\frac{1}{2}, \\ \text{undefined} & \text{if no such } y \text{ exists.}\end{cases}$$

A partial function is *probabilistically computable* if it is computed by some PTM.

Proposition 2.3. *Every function computed by a probabilistic Turing machine is partial recursive. Therefore the class of probabilistically computable functions is the same as the class of partial recursive functions.*

Proof. Every partial recursive function is probabilistically computable, since deterministic Turing machines are special cases of probabilistic Turing machines. Conversely, let M_i be a PTM. To compute $\phi_i(x)$ it suffices to find an output y such that $\Pr\{M_i(x)=y\}>\frac{1}{2}$. For any n we can compute $\Pr\{M_i(x)=y \text{ in time } n\}$, the probability that $M_i(x)$ outputs y in some computation of at most n steps. This probability equals $m2^{-n}$, where m is the number of coin toss sequences α of length n such that $\phi_i(x; \alpha)=y$. If $\Pr\{M_i(x)=y\}>\frac{1}{2}$ for some y, then $\Pr\{M_i(x)=y \text{ in time } n\}>\frac{1}{2}$ for all large enough n. Therefore to compute $\phi_i(x)$, we systematically evaluate $\Pr\{M_i(x)=y \text{ in time } n\}$ until we find some y and n such that this probability exceeds $\frac{1}{2}$. We then set $\phi_i(x)=y$. □

The next two propositions imply that $\{\phi_i\}$ is an acceptable Gödel numbering [21] of the partial recursive functions. We assume a standard computable pairing function $\langle i, x\rangle$.

Proposition 2.4 (Universal PTM). *There is a universal probabilistic Turing machine M_u such that $\phi_u(\langle i, x\rangle; \alpha)=\phi_i(x; \alpha)$ for every binary string α. In particular,* $\Pr\{M_u\langle i, x\rangle=y\}=\Pr\{M_i(x)=y\}$ *for every i, x, and y, and therefore $\phi_u\langle i, x\rangle=\phi_i(x)$.*

Proposition 2.5 (S-m-n Theorem). *There is a recursive function s such that $\phi_{s(e,i)}(x; \alpha)=\phi_e(\langle i, x\rangle; \alpha)$ for every binary string α. In particular,* $\Pr\{M_{s(e,i)}(x)=y\}=\Pr\{M_e\langle i, x\rangle=y\}$ *for every e, i, x, and y, and therefore $\phi_{s(e,i)}(x)=\phi_e\langle i, x\rangle$.*

The proofs of Propositions 2.4 and 2.5 use simulations essentially identical to those used for deterministic Turing machines. As a corollary of these two propositions, the recursion theorem holds for $\{\phi_i\}$. In fact, the standard proof of the recursion theorem [22, p. 180] can be modified to yield the following stronger result.

PROPOSITION 2.6 (Recursion Theorem). *For every total recursive function f there is an index e such that* $\phi_e(x;\alpha)=\phi_{f(e)}(x;\alpha)$ *for every binary string* α. *In particular,* $\Pr\{M_e(x)=y\}=\Pr\{M_{f(e)}(x)=y\}$ *for every x and y, and therefore* $\phi_e=\phi_{f(e)}$.

In the later sections of this paper we shall be interested in the time and tape used by probabilistic machines. Another important aspect of a PTM is its reliability.

DEFINITION 2.7. The *error probability* of M_i is the function e_i defined by

$$e_i(x)=\begin{cases}\Pr\{M_i(x)\neq\phi_i(x)\} & \text{if } \phi_i(x) \text{ is defined,}\\ \text{undefined} & \text{if } \phi_i(x) \text{ is undefined.}\end{cases}$$

Clearly $e_i(x)<\frac{1}{2}$ whenever $e_i(x)$ is defined. In general $e_i(x)$ is not computable, as can be shown using the recursion theorem of Proposition 2.6. However, $e_i(x)$ can be effectively approximated from above.

A useful probabilistic algorithm should have small probability of error. At the very least, the error probability should be uniformly bounded below $\frac{1}{2}$ for all inputs.

DEFINITION 2.8. M_i computes ϕ_i with *bounded error probability* if there is a constant $\varepsilon<\frac{1}{2}$ such that $e_i(x)\leqq\varepsilon$ for every ε in the domain of ϕ_i.

A PTM which computes with bounded error probability corresponds to a probabilistic automaton with an isolated cutpoint at $\frac{1}{2}$ [16], [18].

The probabilistic Turing machine model of Definition 2.1 restricts the randomness accessible by the computer to be of a very simple type—independent equiprobable bits. The more general probabilistic Turing machines of Santos [23] can be simulated by coin-tossing machines, provided that coins with arbitrary biases are allowed. The computational power of coin-tossing machines with biased coins is characterized by the following proposition, part of which appears implicitly in [24].

PROPOSITION 2.9. *Let* $p_1, p_2, \cdots, p_k$ *be real numbers such that* $0\leqq p_i\leqq 1$. *The partial functions computable probabilistically by PTMs which make random decisions with biases drawn from* $\{p_1, p_2, \cdots, p_k\}$ *are exactly the functions partial recursive in the binary representations of* $p_1, p_2, \cdots, p_k$.

3. Time.

In the last section we showed that only partial recursive functions are probabilistically computable, and so no new functions can be computed by probabilistic algorithms. The natural question is whether probabilistic machines can compute more efficiently than deterministic machines, that is, using less time or tape. We study time in this and the following two sections and tape in the final section.

First we must agree on the definition of run time for PTMs. The obvious measure of probabilistic run time is the average time. (In some situations the maximum running time is more appropriate.) Although average run time is the first choice as measure of computation time for PTMs, it has theoretical drawbacks. Average run time is not a Blum complexity measure for the entire class of PTMs.

We recall that $\{\phi_i, \Phi_i\}$ is a Blum measure of computational complexity if $\{\phi_i\}$ is an acceptable Gödel numbering and $\{\Phi_i\}$ is a sequence of partial recursive functions satisfying the two axioms [1]:

(B1) $\phi_i(x)$ is defined iff $\Phi_i(x)$ is defined;

(B2) $\Phi_i(x) \leqq n$ is a decidable predicate of i, x, and n.

Suppose that ϕ_i denotes the partial recursive function computed by the probabilistic Turing machine M_i. Let us write $\bar{T}_i(x)$ for the least integer greater than or equal to the average run time of $M_i(x)$. Then $\{\phi_i, \bar{T}_i\}$ is not a complexity measure for several reasons:

(i) $\bar{T}_i$ is not in general a computable function. However, $\bar{T}_i$ is approximable from below; that is, there is a recursive function $g(i, x, n)$, nondecreasing in n, such that $\bar{T}_i(x) = \lim_{n\to\infty} g(i, x, n)$. In fact, *every* partial function ψ such that $\psi(x) \geqq 3$ and $\psi(x) = \lim_{n\to\infty} g(x, n)$ for some recursive function g nondecreasing in n is the average run time of some probabilistic algorithm.

(ii) $\{\phi_i, \bar{T}_i\}$ does not satisfy axiom (B1). On the one hand, $\bar{T}_i(x)$ may be finite even if $\phi_i(x)$ is undefined, for example, if $\Pr\{M_i(x) = 0\} = \Pr\{M_i(x) = 1\} = \frac{1}{2}$. A similar situation occurs for tape and tape reversal measures for deterministic Turing machines. This difficulty could be removed by redefining $\bar{T}_i(x)$ to be infinite whenever $\phi_i(x)$ is undefined. However, $\bar{T}_i(x)$ may be infinite even when $\phi_i(x)$ is defined.

(iii) $\{\phi_i, \bar{T}_i\}$ does not satisfy axiom (B2). This can be shown directly using the recursion theorem as in [8], or it can be inferred from the following result.

PROPOSITION 3.1. *Every recursively enumerable set is accepted by some probabilistic Turing machine with finite average run time.*

Proof. By definition, the set accepted by a Turing machine (deterministic or probabilistic) is the domain of the function computed by the machine. Let W be any r.e. set, and suppose that W is accepted by a deterministic Turing machine M. A probabilistic machine M' accepting W executes the following program:

```
1 repeat
2   simulate one step of $M(x)$;
3   if $M(x)$ accepted at last step then accept;
4 until cointoss() = heads;
5 if cointoss() = heads then accept else reject.
```

Here cointoss is a Boolean function that returns the result of an unbiased coin toss.

For x not in W, the domain of M, the above procedure can terminate only at line 5, and so it rejects and accepts with probability $\frac{1}{2}$; thus x is not in the domain of M'. On the other hand, if x is in W, then M' accepts on line 3 with positive probability, and so M' accepts x with probability greater than $\frac{1}{2}$. Therefore M' accepts W. A straightforward calculation bounds the average run time of M' by 5. □

Suppose that we modify the flowchart in the proof of Proposition 3.1 by changing line 3 to

```
3' if $M(x)$ accepted at last step then loop forever;
```

Let $\bar{T}(x)$ denote the average run time of M' with input x. Then $\bar{T}(x) = \infty$ for x in W and $\bar{T}(x) \leqq 5$ for x not in W. Thus x belongs to W iff $\bar{T}(x) > 5$. We conclude that $\{\phi_i, \bar{T}_i\}$ does not satisfy axiom (B2), since otherwise every r.e. set would be recursive.

The proof of the following result is similar to that of Proposition 3.1.

COROLLARY 3.2. *Every* 0, 1-*valued recursive function can be computed probabilistically with finite average run time.*

A relatively simple diagonalization shows that the restriction of Corollary 3.2 to 0, 1-valued functions in necessary.

PROPOSITION 3.3. (i) *For every recursive function h there is a* 0, 1, 2-*valued recursive function f requiring average run time more than $h(x)$ a.e.*

(ii) *Every recursive function f requires average run time more than $\frac{1}{2}|f(x)|$ on input x.*

The procedure described in Proposition 3.1 is not a useful method for accepting r.e. sets, since the error probability is very nearly $\frac{1}{2}$. According to Proposition 3.7 below, average run time is a more reasonable cost measure when restricted to PTMs with bounded error probability.

Maximum run time is also inadequate for measuring the time of probabilistic computations, primarily because maximum run time can be infinite even for algorithms with finite average run time. We are thus led to the following "theoretical" definition of probabilistic time. Recall that $\Pr\{M_i(x)=y \text{ in time } n\}$ is the probability that M_i with input x gives output y in some computation of at most n steps.

DEFINITION 3.4. The *Blum run time T_i* of probabilistic Turing machine M_i is defined by

$$T_i(x)=\begin{cases}\text{least } n \text{ such that} & \\ \Pr\{M_i(x)=\phi_i(x) \text{ in times}\}>\frac{1}{2} & \text{if } \phi_i(x) \text{ is defined,}\\ \infty & \text{if } \phi_i(x) \text{ is undefined.}\end{cases}$$

This definition, which also appears in [30], is analogous to the definition of the run time of a nondeterministic Turing machine as the length of the shortest accepting computation.

PROPOSITION 3.5. $\{\phi_i, T_i\}$ *is a Blum complexity measure.*

Proof. To verify the first axiom, we must show that $T_i(x)<\infty$ iff $\phi_i(x)$ is defined. By definition, $T_i(x)=\infty$ if $\phi_i(x)$ is not defined. Conversely, suppose that $\phi_i(x)$ is defined. Then $\Pr\{M_i(x)=\phi_i(x)\}>\frac{1}{2}$, and so $\Pr\{M_i(x)=\phi_i(x) \text{ in time } n\}>\frac{1}{2}$ for all large enough n. Therefore $T_i(x)<\infty$.

The second axiom is also easily verified. For any i, x, and n, we show how to decide deterministically in time $O(n^2 2^n)$ whether $T_i(x)\leqq n$. As in the proof of Proposition 2.3, we can calculate $\Pr\{M_i(x)=y \text{ in time } n\}$ by a straightforward simulation of all possible computations of $M_i(x)$. Similarly we can calculate the probability that y is a prefix of the output of $M_i(x)$ in time n, which we denote by $\Pr\{M_i(x)\sqsupset y \text{ in time } n\}$. These probabilities can be computed in time $O(n2^n)$. Now we proceed by induction on m to determine if the output of $M_i(x)$ in time n is of length m. For $m=0$ we merely check if $\Pr\{M_i(x)=\Lambda \text{ in time } n\}>\frac{1}{2}$. Suppose that for some $m<n$ we have found y of length m such that $\Pr\{M_i(x)\sqsupset y \text{ in time } n\}>\frac{1}{2}$ but $\Pr\{M_i(x)=y \text{ in time } n\}\leqq\frac{1}{2}$. We compute $\Pr\{M_i(x)=y' \text{ in time } n\}$ and $\Pr\{M_i(x)\sqsupset y' \text{ in time } n\}$ for each y' of length $m+1$ that extends y. If $\Pr\{M_i(x)=y' \text{ in time } n\}>\frac{1}{2}$ for some such y' then $T_i(x)\leqq n$. Otherwise either $\Pr\{M_i(x)\sqsupset y' \text{ in time } n\}>\frac{1}{2}$ for some y' of length $m+1$ or $\phi_i(x)$ is undefined and therefore $T_i(x) > n$. Thus either for some $m \leqq n$ we find the output $\phi_i(x)$ of length m and conclude that $T_i(x) \leqq n$ or we verify that $T_i(x) > n$. The entire procedure can be performed in time $O(n) \cdot O(n2^n) = O(n^2 2^n)$. □

PROPOSITION 3.6. *If M is a probabilistic Turing machine with Blum run time T, then M can be simulated by a deterministic Turing machine in time $O(T^2 2^T)$.*

Proof. For $n = 1, 2, 3, \cdots$ we use the procedure outlined in the proof of Proposition 3.5 to decide if $T(x) \leqq n$. For $n = T(x)$ the procedure produces the output of $M(x)$. The total computation time is at most $O(1^2 \cdot 2^1) + O(2^2 \cdot 2^2) + \cdots + O(T(x)^2 2^{T(x)}) = O(T(x)^2 2^{T(x)})$. □

The next result gives evidence that Definition 3.4 is a reasonable notion of probabilistic run time.

PROPOSITION 3.7. *If M_i is a probabilistic Turing machine with bounded error probability, then there is a constant $c > 0$ such that $T_i(x) \leqq c\bar{T}_i(x)$ whenever $\phi_i(x)$ is defined.*

Proof. If M_i has bounded error probability then there is a constant $\varepsilon < \frac{1}{2}$ such that $e_i(x) < \varepsilon$ for every x in the domain of ϕ_i. Let $c = 1/(\frac{1}{2} - \varepsilon)$.

If $\bar{T}_i(x) = \infty$ then there is nothing to prove. So suppose that $\bar{T}_i(x) < \infty$. Obviously Pr {run time of $M_i(x) > c\bar{T}_i(x)$} $< 1/c = \frac{1}{2} - \varepsilon$, or equivalently, Pr {run time of $M_i(x) \leqq c\bar{T}_i(x)$} $> \frac{1}{2} + \varepsilon$. Since Pr {$M_i(x) \neq \phi_i(x)$ in time $c\bar{T}_i(x)$} $\leqq$ Pr {$M_i(x) \neq \phi_i(x)$} $< \varepsilon$, it follows that Pr {$M_i(x) = \phi_i(x)$ in time $c\bar{T}_i(x)$} $> \frac{1}{2}$. Therefore $T_i(x) \leqq c\bar{T}_i(x)$. □

No reverse inequality is true in general; there are PTMs with bounded error probability for which $\bar{T}_i(x) = \infty$ and $T_i(x) < \infty$. Proposition 3.7 states that for PTMs with bounded error probability, the average run time cannot be much less than the Blum run time, and therefore the average run time does not assign too small a cost to computations. The pathology of Proposition 3.1 cannot occur for PTMs with bounded error probability.

COROLLARY 3.8. *If M is a probabilistic Turing machine with bounded error probability and average run time $\bar{T}$, then M can be simulated by a deterministic Turing machine in time $2^{O(\bar{T})}$.*

Proof. Let $\varepsilon < \frac{1}{2}$ be a uniform error probability bound for M. Then by Propositions 3.6 and 3.7, if $c = 1/(\frac{1}{2} - \varepsilon)$, then M can be simulated deterministically in time $O((c\bar{T})^2 2^{c\bar{T}}) = 2^{O(\bar{T})}$. □

To conclude this section, we list fundamental open problems about the computational power of probabilistic Turing machines. We state these problems as positive conjectures.

Conjecture 1. There is a function computable probabilistically in polynomial time but not computable deterministically in polynomial time.

Conjecture 2. There is a function computable probabilistically with bounded error probability in polynomial time but not computable deterministically in polynomial time.

Conjecture 3. There is a function computable probabilistically with zero error probability in polynomial bounded *average* running time but not computable deterministically in polynomial time.

Conjecture 2 obviously implies Conjecture 1, and it can be shown, as in Proposition 5.2, that Conjecture 3 implies Conjecture 2. In § 5 we observe that Conjecture 1 is true if $P \neq NP$; thus Conjecture 1 is quite plausible.

A function satisfying the conditions of Conjecture 2 would be much more interesting. One candidate is PRIMES, the characteristic function of the set of prime numbers. Solovay and Strassen [26] and Rabin [20] have described efficient Monte Carlo tests for primality, thus showing that the primes can be recognized probabilistically in polynomial time with bounded error probability. (However, there is also the possibility that the primes can be recognized *deterministically* in polynomial time, which is the case if the extended Riemann hypothesis is true [14].) We suggest that other difficult (but not NP-hard) problems might be solvable probabilistically in polynomial time, such as graph isomorphism, linear inequalities, and optimum trees with unequally weighted branches.

In the next section we give an affirmative answer to a much weaker version of Conjecture 2.

4. An example of probabilistic speedup.

In this section we exhibit a language that can be recognized more rapidly by one-tape probabilistic Turing machines than by one-tape deterministic Turing machines. The advantage of restricting ourselves to one-tape machines is the availability of crossing sequence techniques [19], [10], [28] for establishing lower bounds on the complexity of some simple computational problems.

We denote the ith symbol of a string w by $w[i]$. For any string w of even length $2n$, let $r(n)$ be the fraction of symbols in the first half of w which equal the corresponding symbols in the latter half of w; that is, $r(w)=m/n$, where m is the number of indices i between 1 and n such that $w[i]=w[n+i]$. For any real number λ let P_λ be the palindrome-like language of binary strings w of even length such that $r(w)\geqq\lambda$. In particular, $P_1=\{ww: w \text{ is a binary string}\}$ is a subset of P_λ for every $\lambda<1$. We denote by P_1^{2n} the strings of P_1 of length $2n$.

THEOREM 4.1. *Suppose that λ is a rational number such that $0<\lambda<1$. There is a one-tape probabilistic Turing machine M that recognizes P_λ with bounded error probability and runs faster for infinitely many inputs than any one-tape deterministic Turing machine that recognizes P_λ. That is, if M' is any one-tape deterministic Turing machine that recognizes P_λ, then the (maximum) run time of M is less than the run time of M' for infinitely many inputs.*

Proof. In Lemma 4.2 we describe a one-tape probabilistic Turing machine M which recognizes P_λ in time $O(n \log n)$ for inputs in P_1. In Lemma 4.3 we prove that for every one-tape deterministic Turing machine M' that recognizes P_λ there is a constant $c>0$ such that the maximum run time of M' on inputs in P_1^{2n} is at least cn^2. Therefore Lemmas 4.2 and 4.3 provide the proof of the theorem. □

LEMMA 4.2. *For every rational number λ such that $0<\lambda<1$ there is a one-tape probabilistic Turing machine M that recognizes P_λ with bounded error probability and maximum run time $O(n \log n)$ for inputs in P_1.*

Proof. For any error probability bound $\varepsilon>0$ let m be an integer large enough that $((1+\lambda)/2)^m<\varepsilon$. Suppose that M_0 is any standard one-tape deterministic Turing machine that recognizes P_λ. (Straightforward one-tape Turing machine programming techniques yield a machine M_0 with run time $O(n^2)$.) Let M be a one-tape probabilistic machine that with input w operates as follows:

(i) M checks that the input length is even, say $2n$, and calculates $\tilde{n}$, the binary representation of n. (By the usual one-tape machine method for converting a unary input to binary [15, p. 123], this stage can be performed in $O(n \log n)$ steps.)

(ii) M randomly selects m numbers $i_1, i_2, \cdots, i_m$ such that $1 \leqq i_j \leqq n$. M selects the index i_j by letting $i_j = 1 + (\alpha_j \bmod n)$, for a randomly chosen bit string α_j of length $|\tilde{n}|$. Note that i_j assumes values of numbers between 1 and n with probabilities $2^{-|\tilde{n}|}$ or $2^{-|\tilde{n}|+1}$. (Obtaining these random samples requires only $O(\log n)$ steps.)

(iii) M compares $w[i_j]$ and $w[n+i_j]$ for $j = 1, 2, \cdots, m$. If $w[i_j] = w[n+i_j]$ for every j, then M accepts w. (Each comparison of $w[i_j]$ with $w[n+i_j]$ can be made in $O(n \log n)$ steps, using a familiar technique for selecting the ith symbol of the tape. In this technique, we load i into a counter stored on the tape, then while decrementing the counter advance the tape head and simultaneously drag the counter along.)

(iv) If $w[i_j] \neq w[n+i_j]$ for some j, then M simulates the deterministic machine M_0 to determine if w belongs to P_λ.

Clearly, inputs in P_1 are accepted by M at the end of stage (iii). Therefore M runs in time $O(n \log n)$ on P_1.

We must show that M recognizes P_λ with error probability less than ε. Because M does not falsely reject strings in P_λ, it is enough to bound above the probability of falsely accepting a string w not in P_λ. Let $2n$ be the length of w. Since w is not in P_λ, more than $(1-\lambda)n$ of the numbers i between 1 and n satisfy $w[i] \neq w[n+i]$. It follows easily that the probability that $w[i_j] \neq w[n+i_j]$ for any one of the random samples i_j is at least $(1-\lambda)/2$, and so the probability that $w[i_j] = w[n+i_j]$ is at most $(1+\lambda)/2$. Since the samples $i_1, i_2, \cdots, i_m$ are independent, the probability that M accepts w, which is the probability that $w[i_j] = w[n+i_j]$ for every j, is at most $((1+\lambda)/2)^m < \varepsilon$. Therefore the error probability of M is less than ε. □

LEMMA 4.3. *Suppose that* $0 < \lambda < 1$. *For every one-tape deterministic Turing machine* M' *that recognizes* P_λ *there is a constant* $c > 0$ *such that the maximum run time of* M' *on inputs in* P_1 *of length* $2n$ *is at least* cn^2.

Proof. We use a crossing sequence argument [10]. Let $\mu = 1 - \lambda$ and let ν be any number such that $\mu < \nu < 1$. Suppose that M' is a one-tape deterministic machine that recognizes P_λ. Fix n. For each k such that $\nu n \leqq k \leqq n$ we examine the crossing sequences of M' at boundary k of words in P_1^{2n}.

Suppose that uu and vv are strings in P_1^{2n} with the same crossing sequence at boundary k. Write uu and vv as $u_1u_2u_1u_2$ and $v_1v_2v_1v_2$, where $|u_1| = |v_1| = k$. Since both uu and vv are in P_1 and hence in P_λ, they are both accepted by M'. By the fundamental property of crossing sequences, M' accepts the hybrid string $u_1v_2v_1v_2$, and so $u_1v_2v_1v_2$ is in P_λ. Therefore u_1 and v_1 differ in at most $(1-\lambda)n = \mu n$ positions.

For a fixed crossing sequence ξ and string v_2 of length $n-k$, let $S(\xi, v_2)$ be the set of strings v_1 of length k such that $v_1v_2v_1v_2$ has crossing sequence ξ at boundary k. We shall show that there is a constant $\rho < 1$ such that the number $N(\xi, v_2)$ of elements in $S(\xi, v_2)$ is at most $2^{\rho k}$. In fact,

$$N(\xi, v_2) \leqq \sum_{i=0}^{\mu n/2} \binom{k}{i} \leqq 2^{kH(\mu n/(2k))} \leqq 2^{kH(\mu/(2\nu))}, \tag{4.1}$$

where H is the binary entropy function [7, p. 78] defined by

$$H(x) = -x \log_2 x - (1-x)\log_2(1-x).$$

The first inequality in (4.1) follows from a theorem of Kleitman [12] on the maximum number of binary sequences of length k, no two of which differ in more than μn positions. The second inequality is an application of the Chernoff bound [2] and can be found in [17, p. 466]. The final inequality holds because $H(x)$ is strictly increasing for $0 \leqq x \leqq \frac{1}{2}$. Let $\rho = H(\mu/(2\nu))$. Then $\rho < 1 = H(\frac{1}{2})$ because $\mu/2\nu < \frac{1}{2}$. Thus inequality (4.1) can be rewritten $N(\xi, v_2) \leqq 2^{\rho k}$.

For a fixed v_2 of length $n-k$, at most $2^{\rho k}$ strings $v_1 v_2 v_1 v_2$ in P_1^{2n} have crossing sequence ξ at boundary k. Summing over v_2, we see that the number $N(\xi)$ of strings in P_1^{2n} with crossing sequence ξ at boundary k is bounded by $2^{n-\delta n}$ for some $\delta > 0$. In fact,

$$N(\xi) \leqq 2^{n-k} \cdot 2^{\rho k} = 2^n \cdot 2^{-(1-\rho)k} \leqq 2^n \cdot 2^{-(1-\rho)\nu n}, \tag{4.2}$$

since $k \geqq \nu n$. We can take $\delta = (1-\rho)\nu$. There are 2^n sequences in P_1^{2n}, and so among all strings in P_1^{2n} at least $2^{\delta n}$ distinct crossing sequences must occur at boundary k.

Next we estimate the average, over all words in P_1^{2n}, of the crossing sequence length at boundary k. If s is the number of states of M', then there are s^i crossing sequences of length i. Define l by

$$\sum_{i=0}^{l} s^i \leqq 2^{\delta n} < \sum_{i=0}^{l+1} s^i. \tag{4.3}$$

(Note that for some string in P_1^{2n} a crossing of length at least l must occur at boundary k.) From (4.3) we infer that $s^l \geqq s^{-2} 2^{\delta n}$, and consequently $l \geqq (\delta/\log_2 s)n - 2$. The average crossing sequence length at boundary k is at least

$$2^{-n} \sum_{i=0}^{l} i s^i 2^{n-\delta n} \geqq l s^l 2^{-\delta n} \geqq l/s^2 \geqq (\delta/(s^2 \log_2 s))n - 2/s^2, \tag{4.4}$$

because at most $s^i 2^{n-\delta n}$ strings have crossing sequences of length i at boundary k.

The computation time of M' on any input is the sum of the crossing sequence lengths over all boundaries. Summing the average crossing sequence length over all boundaries k such that $\nu n \leqq k \leqq n$, we obtain the following lower bound for the average computation time of M' on inputs in P_1^{2n}:

$$(n - \nu n)[(\delta/(s^2 \log_2 s))n - 2/s^2] = an^2 - bn \tag{4.5}$$

for some constants $a, b > 0$. Therefore the average computation time is at least cn^2 for some $c > 0$, and so the maximum run time of M' on P_1^{2n} is at least cn^2. □

The significance of Theorem 4.1 is that it gives an example of a problem and a machine model for which we can *prove* that probabilistic algorithms are faster than deterministic algorithms. The chief limitations of this result are that the amount of the speedup is small and that the machine model is of limited practical interest.

Vaiser [31] constructed a one-tape probabilistic Turing machine that recognizes the palindromes in *linear* time but with unbounded error probability.

Freivald [6] has improved Theorem 4.1 by showing that P_1 can be recognized by a one-tape probabilistic Turing machine with bounded error probability in time $O(n \log^2 n)$.

5. Probabilistic polynomial languages.

5. Probabilistic polynomial languages. We now define several classes of languages computable probabilistically in polynomial time and investigate relationships between these classes.

A probabilistic or nondeterministic Turing machine is *polynomial bounded* if there is a polynomial $p(n)$ such that every possible computation of the machine on inputs of length n halts in at most $p(n)$ steps. A probabilistic machine *recognizes* a language if the machine computes the characteristic function of the language.

DEFINITION 5.1. (i) *PP* is the class of languages recognized by polynomial bounded PTMs.

(ii) *BPP* is the class of languages recognized by polynomial bounded PTMs with bounded error probability.

(iii) *ZPP* is the class of languages recognized by PTMs with polynomial bounded *average* run time and zero error probability.

PROPOSITION 5.2. (i) $ZPP \subseteq BPP \subseteq PP \subseteq PSPACE$.

(i) *PP, BPP, and ZPP are closed under complementation.*

(iii) *BPP and ZPP are closed under union and intersection.*

Proof. (i) It is clear from the proof of Proposition 3.5 that every polynomial bounded Turing machine can be simulated in polynomial space, and so $PP \subseteq PSPACE$. By definition, $BPP \subseteq PP$. To show that $ZPP \subseteq BPP$, suppose that a language L is recognized by a probabilistic machine M with zero error probability and average run time bounded by a polynomial $p(n)$. For any constant $c > 2$, let M' be a PTM that recognizes L by simulating M for up to $cp(n)$ steps on inputs of length n. If the simulated computation of M does not halt within this time, then M' halts with an arbitrary answer. Since M requires more than $cp(n)$ steps with probability less than $1/c$, the error probability of the polynomial bounded machine M' is at most $1/c < \frac{1}{2}$.

Part (ii) is obvious from the definitions.

(iii) Suppose that L_1 and L_2 are recognized with zero error probability by M_1 and M_2 with average run times bounded respectively by $p_1(n)$ and $p_2(n)$. By a standard construction we obtain from M_1 and M_2 a machine that recognizes $L_1 \cup L_2$ with zero error probability and average run time at most $n + p_1(n) + p_2(n)$. Therefore *ZPP* is closed under union.

We note that every language in *BPP* can be recognized by a polynomial bounded PTM with error probability smaller than any desired positive constant, since we can increase the reliability of a probabilistic computer by repeating its computations a sufficiently large number of times and giving as output the majority result. Suppose that L_1 and L_2 belong to *BPP*. For every $\varepsilon > 0$ there exist polynomial bounded machines M_1 and M_2 that recognize L_1 and L_2 with error probability at most $\varepsilon/2$. The standard machine derived from M_1 and M_2 that recognizes $L_1 \cup L_2$ is polynomial bounded and has error probability at most $\varepsilon/2 + \varepsilon/2 = \varepsilon$. Therefore *BPP* is closed under union.

The closure of *ZPP* and *BPP* under intersection follows from closure under union and complementation. □

It is not known whether *PP* is closed under union and intersection. If *PP* is not closed under union, then by Propositions 5.2 and 5.3, $NP \subsetneqq PP \subsetneqq PSPACE$.

PROPOSITION 5.3. $P \subseteq ZPP \subseteq NP \subseteq PP$.

Proof. Every polynomial bounded deterministic Turing machine computes with zero error probability. Therefore $P \subseteq ZPP$.

Suppose that L is in *ZPP* and M is a PTM that recognizes L with zero error probability and polynomial bounded average running time. Then M considered as a nondeterministic Turing machine accepts L in polynomial time, because for every input in L there is at least one accepting computation of M of polynomial bounded length. Therefore L is in *NP*, and so $ZPP \subseteq NP$.

Finally suppose that L is in *NP*. Without loss of generality, we may assume that L is accepted by a polynomial bounded nondeterministic machine M for which each state of the machine permits at most two possible next actions. If M is considered to be a probabilistic machine, then L is the set of strings for which there exists an accepting computation; that is, x is in L iff $\Pr\{M(x) \text{ accepts}\} > 0$. To show that L belongs to *PP*, we replace M by a machine M' such that $\Pr\{M(x) \text{ accepts}\} > 0$ iff $\Pr\{M'(x) \text{ accepts}\} > \frac{1}{2}$. The machine M' tosses a coin at the beginning of its computation and accepts immediately if the result is heads; otherwise M' simulates M probabilistically, accepting iff M accepts.

There remains one small detail. If x is not in L then it is possible that $\Pr\{M'(x) \text{ accepts}\} = \frac{1}{2}$. Thus M' might not compute the characteristic function of L. We must make a final modification to obtain a machine M'' such that $\Pr\{M''(x) \text{ accepts}\} < \frac{1}{2}$ for x not in L.

Let $p(n)$ be a polynomial bounding the run time of M. Every x in L of length n is accepted by M with probability at least $2^{-p(n)}$, since there is at least one accepting computation and every computation of length $p(n)$ has probability at least $2^{-p(n)}$. A probabilistic machine M'' recognizing L operates as follows. At the beginning of its computation, M'' tosses $p(n)+1$ coins and accepts without further computation with probability $\frac{1}{2}-2^{-p(n)-1}$; otherwise M'' simulates M, accepting iff M accepts. It is easily calculated that M'' rejects inputs not in L with probability $\frac{1}{2}+2^{-p(n)-1}$ and accepts inputs in L with probability at least $\frac{1}{2}+2^{-2p(n)-1}$. Therefore M'' recognizes L probabilistically in polynomial time. We conclude that $NP \subseteq PP$. □

The most important question about the classes of probabilistic polynomial languages is whether they properly contain P. We believe that $P \subsetneqq ZPP$, which can be seen to be equivalent to Conjecture 3 of § 3. Furthermore, there is evidence that $P \subsetneqq BPP$ and $P \subsetneqq PP$.

It appears quite likely that $P \subsetneqq PP$, since $NP \subseteq PP$. In fact, Simon [25] has listed a large number of combinatorial problems that are polynomial complete in *PP*. These problems seem to be intermediate in complexity between *NP*-complete problems and *PSPACE*-complete problems, which suggests that $NP \subsetneqq PP$.

PRIMES, the set of prime numbers, is the leading candidate for a language in $BPP - P$. Rabin [20] and Solovay and Strassen [26] have devised probabilistic algorithms that recognize the prime numbers in polynomial time with bounded error probability, and so PRIMES is in *BPP*. (Rabin has used his probabilistic algorithm to discover a 400-bit number that is "very probably" a prime.) If it can be shown that PRIMES cannot be recognized deterministically in polynomial time, then PRIMES is in $BPP - P$.

We can suggest no language in *ZPP* − *P*. Rabin noted [20] that both Rabin's and Solovay–Strassen's primality testing algorithms always correctly identify prime numbers. If a probabilistic primality testing algorithm can be found which always correctly identifies composite numbers (but may make mistakes on prime numbers), then this algorithm can be combined with Rabin's algorithm to yield a procedure that recognizes prime numbers with zero error probability in polynomial bounded average time. This observation leads to the following definition and proposition.

DEFINITION 5.4. *VPP* is the class of languages recognized by polynomial bounded PTMs which have zero error probability for inputs not in the languages. Equivalently, L is in *VPP* iff L is recognized by a probabilistic Turing machine M such that $\Pr\{M(x) \text{ accepts}\} = 0$ for every x not in L.

The composite numbers are an example of a language in *VPP*.

PROPOSITION 5.5. (i) *L is in VPP iff L is accepted by a PTM whose average run time is polynomial bounded on L.*

(ii) $VPP \subseteq NP \cap BPP$.

(iii) [Rabin] *L is in ZPP iff both L and $\bar{L}$ are in VPP.*

We omit the easy proof of Proposition 5.5.

It does not appear that either $BPP \subseteq NP$ or $NP \subseteq BPP$. We note that $NP \subseteq BPP$ if every language in *NP* can be accepted by a polynomial bounded nondeterministic machine such that for inputs in the language accepting computations are a large fraction of possible computations. The following diagram summarizes known relations among the classes *P*, *ZPP*, *VPP*, *BPP*, *NP*, *PP*, and *PSPACE*.

$$\begin{tikzcd}[column sep=small, row sep=tiny] & & & BPP \arrow[dr, phantom, "\subseteq"] & & \\ P \subseteq ZPP \subseteq VPP \arrow[urr, phantom, "\subseteq"] \arrow[drr, phantom, "\subseteq"] & & & & PP \subseteq PSPACE. \\ & & & NP \arrow[ur, phantom, "\subseteq"] & & \end{tikzcd}$$

None of the inclusions are known to be proper. We also know little about the relation between *PP* and the polynomial arithmetic hierarchy of Meyer and Stockmeyer [13], [27], which is also contained in *PSPACE*.

The following proposition is analogous to the characterization [3] of *NP* as the class of languages L for which there exist a polynomially computable relation $R(x, y)$ and a polynomial $p(n)$ such that $L = \{x : R(x, y) \text{ holds for some string } y \text{ of length} \leqq p(|x|)\}$.

PROPOSITION 5.6. *A language L belongs to PP iff there exist a polynomially computable relation $R(x, y)$ and a polynomial $p(n)$ such that $L = \{x : R(x, y)$ holds for a majority of strings y of length $p(|x|)\}$.*

To conclude this section, we describe a simple polynomial complete language for *PP*. A language B is *polynomial m-reducible* to another language A if there is a function f computable in polynomial time such that x is in B iff $f(x)$ is in A. If A and B are polynomial m-reducible to each other, then A and B are *polynomial m-equivalent.* A language A is *polynomial m-complete* in a class if A belongs to that class and every language in that class is polynomial m-reducible to A.

An *interpretation* of a propositional formula $F(x_1, \cdots, x_n)$ is any assignment of truth values to the propositional variables $x_1, \cdots, x_n$ of F. A *satisfying interpretation* of $F(x_1, \cdots, x_n)$ is an interpretation under which F is true. The set of formulas for which there is at least one satisfying interpretation is denoted by SAT. It is well known that SAT is polynomial m-complete for *NP* [3].

DEFINITION 5.7. (i) MAJ is the set of propositional formulas satisfied by a majority of their interpretations; that is, $F(x_1, \cdots, x_n)$ is in MAJ iff $F(x_1, \cdots, x_n)$ is true for more than 2^{n-1} assignments of truth values to $x_1, \cdots, x_n$.

(ii) #SAT is the set of pairs $\langle i, F\rangle$ such that propositional formula F has more than i satisfying interpretations.

Simon [25] has shown that *PP* is the class of languages accepted by polynomial bounded threshold machines and that #SAT and a large number of similar combinatorial problems are polynomial m-complete for threshold machines and hence for *PP*.

LEMMA 5.8 [Simon]. *#SAT is polynomial m-complete for PP.*

LEMMA 5.9. MAJ *and* #SAT *are polynomial m-equivalent.*

Proof. It is clear that MAJ is polynomial m-reducible to #SAT, because $F(x_1, \cdots, x_n)$ is in MAJ iff $\langle 2^{n-1}, F(x_1, \cdots, x_n)\rangle$ is in #SAT.

To show that #SAT is polynomial m-reducible to MAJ, suppose that w is an arbitrary input of the form $\langle i, F(x_1, \cdots, x_n)\rangle$. We may assume that $i < 2^n$, since it is obvious that w is not in #SAT if $i \geqq 2^n$. Let $G(x_1, \cdots, x_n)$ be a formula, computable in polynomial time, that has exactly $2^n - i$ satisfying interpretations,[1] and let x_0 be a propositional variable not occurring in $F(x_1, \cdots, x_n)$. Then the formula $H(x_0, x_1, \cdots, x_n) = x_0F(x_1, \cdots, x_n) \vee \bar{x}_0G(x_1, \cdots, x_n)$ has more than 2^n satisfying interpretations iff $F(x_1, \cdots, x_n)$ has more than i satisfying interpretations. Therefore $\langle i, F\rangle$ is in #SAT iff H is in MAJ. □

PROPOSITION 5.10. MAJ *is polynomial m-complete for PP.*

Proof. This follows immediately from Lemmas 5.8 and 5.9. For completeness, we show that MAJ is in *PP* by describing a probabilistic polynomial time algorithm for recognizing MAJ. With input $F(x_1, \cdots, x_n)$ we select equiprobably one of the 2^n possible interpretations and evaluate $F(x_1, \cdots, x_n)$ for this interpretation. If $F(x_1, \cdots, x_n)$ is false for this interpretation then we reject, while if $F(x_1, \cdots, x_n)$ is true then we accept with probability $1 - 2^{-n-1}$. It is easily calculated that formulas in MAJ are accepted with probability greater than $\frac{1}{2} + 2^{-n-1}$ and formulas not in MAJ are rejected with probability greater than $\frac{1}{2} + 2^{-n-2}$. □

We have been unable to construct polynomial complete languages for *ZPP* or *BPP*.

6. Tape.

The chief purpose of this section is to show that the definition of probabilistic tape analogous to that of probabilistic time (Definition 3.4) yields a Blum complexity measure. As a corollary of justifying the definition, we see how to simulate deterministically a probabilistic machine in at most exponentially more space. This is the best result now known. At the end of the section we point out that every language accepted by a nondeterministic Turing machine can be recognized in the same tape by a PTM with bounded error probability. This is evidence that probabilistic machines might be more efficient than deterministic machines in the use of tape.

[1] $G(x_1, \cdots, x_n) = x_1 \cdots x_{r_1} \vee x_1 \cdots \bar{x}_{r_1} \cdots x_{r_2} \vee \cdots \vee x_1 \cdots \bar{x}_{r_1} \cdots \bar{x}_{r_{k-1}} \cdots x_{r_k}$, where $2^n - i = 2^{n-r_1} + 2^{n-r_2} + \cdots + 2^{n-r_k}$ and $1 \leqq r_1 < r_2 \cdots < r_k \leqq n$.

DEFINITION 6.1. The *Blum tape* S_i of probabilistic Turing machine M_i is defined by

$$S_i(x)=\begin{cases}\text{least } n \text{ such that} & \\ \Pr\{M_i(x)=\phi_i(x) \text{ in tape } n\}>\frac{1}{2} & \text{if } \phi_i(x) \text{ is defined,}\\ \infty & \text{if } \phi_i(x) \text{ is undefined.}\end{cases}$$

The proof that probabilistic tape is a complexity measure is rather involved. We require a preliminary result bounding the length of the output of a PTM. (The corresponding result for deterministic machines is trivial.) We need a lemma about finite state Markov chains.

LEMMA 6.2. *Suppose that $\mathcal{M}$ is a Markov chain with s states. Let $f_{ij}(n)$ be the probability that $\mathcal{M}$, when started in state i at time 0, first reaches state j at time n. If $n \geqq s$ and $i \neq j$ then $f_{ij}(n) \leqq \frac{1}{2}$.*

Proof. Let I be the finite set of states and (p_{ij}) the transition probability matrix of $\mathcal{M}$. Fix a state j. Without loss of generality, we can assume that j is a trap state, since the first entry probabilities $f_{ij}(n)$ for $i \neq j$ do not depend on the transition probabilities from state j.

For any set of states H not including j, define $f_{ij}(n;H)$ and $g_i(n;H)$ as follows:

(i) $f_{ij}(n;H)$ is the probability that $\mathcal{M}$, started in i at time 0, first enters j at time n, without passing through any state of H at times $1, 2, \cdots, n-1$;

(ii) $g_i(n;H)$ is the probability that $\mathcal{M}$, started in i at time 0, passes through no state of H at times $1, 2, \cdots, n$.

To establish the lemma, we prove something stronger: If j is a trap state and (a) i is not in H and $n \geqq |I-H|$ or (b) i is in H and $n \geqq |I-H|+1$, then

$$f_{ij}(n;H) \leqq \tfrac{1}{2} g_i(n;H), \tag{6.1}$$

where $|I-H|$ denotes the number of states in $I-H$. The lemma is the special case $H=\varnothing$. The proof of (6.1) is by induction on $|I-H|$.

If $|I-H|=1$ then $H=I-\{j\}$. Since $i \neq j$ implies that i is in H, only case (b) of (6.1) need be considered. Obviously $f_{ij}(n;H)=0$ if $n \geqq 2=|I-H|+1$, and so (6.1) is true for $|I-H|=1$.

Now suppose that $|I-H|>1$. There are two cases.

Case (a). i is not in H and $n \geqq |I-H|$. Since j is a trap state,

$$\sum_{m=1}^{n} f_{ij}(m;H) \leqq g_i(n;H). \tag{6.2}$$

If $f_{ij}(m;H) > \frac{1}{2} g_i(n;H)$ for any $m<n$, then (6.2) implies that $f_{ij}(n;H) < \frac{1}{2} g_i(n;H)$. So in what follows we may assume that $f_{ij}(m;H) \leqq \frac{1}{2} g_i(n;H)$ for $m<n$. By case (b) of the induction hypothesis,

$$f_{ij}(n;H\cup\{i\}) \leqq \tfrac{1}{2} g_i(n;H\cup\{i\}), \tag{6.3}$$

since $|I-(H\cup\{i\})| < |I-H|$ and $n \geqq |I-H| = |I-(H\cup\{i\})|+1$. Next we note that

$$g_i(n;H) = g_i(n;H\cup\{i\}) + \sum_{m=1}^{n} f_{ii}(m;H) g_i(n-m;H), \tag{6.4}$$

and therefore

$$(6.5)\quad \begin{aligned} g_i(n;H\cup\{i\}) &= g_i(n;H) - \sum_{m=1}^{n} f_{ii}(m;H)g_i(n-m;H) \\ &\leqq g_i(n;H)\left(1-\sum_{m=1}^{n} f_{ii}(m;H)\right), \end{aligned}$$

since $g_i(n;H) \leqq g_i(n-m;H)$. Combining (6.3) and (6.5) we obtain

$$(6.6)\quad f_{ij}(n;H\cup\{i\}) \leqq \tfrac{1}{2}g_i(n;H)\left(1-\sum_{m=1}^{n} f_{ii}(m;H)\right).$$

As in (6.4) we can use (6.6) and the inequalities $f_{ij}(m;H) \leqq \frac{1}{2}g_i(n;H)$ for $m<n$ to bound $f_{ij}(n;H)$:

$$(6.7)\quad \begin{aligned} f_{ij}(n;H) &= f_{ij}(n;H\cup\{i\}) + \sum_{m=1}^{n} f_{ii}(m;H\cup\{j\})f_{ij}(n-m;H) \\ &\leqq \tfrac{1}{2}g_i(n;H)\left(1-\sum_{m=1}^{n} f_{ii}(m;H)\right) + \sum_{m=1}^{n} \tfrac{1}{2}g_i(n;H)f_{ii}(m;H) \\ &= \tfrac{1}{2}g_i(n;H). \end{aligned}$$

Case (b). i is in H and $n \geqq |I-H|+1$. Then

$$(6.8)\quad \begin{aligned} f_{ij}(n;H) &= \sum_{k\notin H} p_{ik}f_{kj}(n-1;H) \\ &\leqq \sum_{k\notin H} p_{ik}\tfrac{1}{2}g_k(n-1;H) \text{ by case (a)} \\ &= \frac{1}{2}\sum_{k\notin H} p_{ik}g_k(n-1;H) = \tfrac{1}{2}g_i(n;H). \quad \square \end{aligned}$$

A probabilistic Turing machine M with a fixed input x can be thought of as a discrete time Markov process [23]. The states of this Markov process are the instantaneous descriptions of M with input x, and the transition probabilities are determined by the state-transition probabilities of M. An *instantaneous description* (ID) of $M(x)$ consists of the position of the input tape head, the state of the finite control, the contents of the worktapes, and the positions of the worktape heads. (The contents of the output tape are not included.) The number of instaneous descriptions of $M(x)$ within worktape n is bounded by $s(|x|+2)n^k d^n$, where s is the number of states of the finite control, k is the number of worktapes, and d is the number of symbols in the tape alphabet. For $n \geqq \log|x|$ this bound can be replaced by c^n for some constant c that depends on M. Let us denote by $I(i, x, n)$ the exact number of instantaneous descriptions of $M_i(x)$ that use at most n worktapes squares. For any computation $M_i(x)$ we let BEGIN be the start ID. Without loss of generality we may assume that there is a unique halt ID, denoted END. (Any PTM can be replaced by one using the same workspace which cleans its worktapes and rewinds its input tape before halting.)

PROPOSITION 6.3. *If S_i is the Blum tape of the probabilistic Turing machine M_i, then the length of the output of $M_i(x)$ is no more than the number of instantaneous* *descriptions of $M_i(x)$ in tape $S_i(x)$. In particular, if $S_i(x) \geqq \log|x|$, then there is a constant c depending on M_i such that $|\phi_i(x)| \leqq c^{S_i(x)}$.*

Proof. An ID of $M_i(x)$ is called a *writing* ID if the next action of $M_i(x)$ specified by the ID includes writing a symbol on the output tape. We construct a Markov chain whose states are the writing IDs of $M_i(x)$ in worktape $S_i(x)$ together with the halt ID END. For any two writing IDs I and I' the one-step transition probability $p(I'|I)$ is defined to be the probability that $M_i(x)$ starting in instantaneous description I reaches I' in a finite number of steps without passing through another writing ID, that is, after writing only one output symbol. The transition probability $p(\text{END}|I)$ is the probability that $M_i(x)$ starting in I halts or loops without writing. END is defined to be a trap state, that is, $p(\text{END}|\text{END}) = 1$.

Suppose that the start ID BEGIN is not a writing ID. Let $f(n)$ be the probability that $M_i(x)$ in tape $S_i(x)$ halts with an output of length exactly n. Then

$$f(n) \leqq \sum_I p(I|\text{BEGIN}) f_{I,\text{END}}(n), \tag{6.9}$$

where $p(I|\text{BEGIN})$ is the probability that I is the first writing ID reached by $M_i(x)$, and $f_{I,\text{END}}(n)$ is the probability that the Markov chain defined above, starting in state I, first reaches the trap state END in exactly n steps. By Lemma 6.2 this latter probability is at most $\frac{1}{2}$ when n exceeds the number of states of the process. Therefore if $n \geqq I(i, x, S_i(x))$ then $f(n) \leqq \frac{1}{2}$ and clearly no string of length n can be the output of $M_i(x)$ with probability greater than $\frac{1}{2}$.

If BEGIN is a writing ID, then $f(n) = f_{\text{BEGIN,END}}(n) \leqq \frac{1}{2}$ if $n \geqq I(i, x, S_i(x))$. Therefore $|\phi_i(x)| \leqq I(i, x, S_i(x))$. □

THEOREM 6.4. *$\{\phi_i, S_i\}$ is a Blum complexity measure.*

Proof. The first axiom is verified as in the proof of Proposition 3.5. To establish the second axiom, we sketch a method for deciding $S_i(x) \leqq n$. The procedure uses space bounded by a polynomial in $I(i, x, n)$.

By Proposition 6.3, if $S_i(x) \leqq n$ then $|\phi_i(x)| \leqq I(i, x, n)$. To decide if $S_i(x) \leqq n$ it is sufficient to determine if $\Pr\{M_i(x) = y \text{ in tape } n\} > \frac{1}{2}$ for each y of length up to $I(i, x, n)$. To answer the latter questions, we shall in fact calculate $\Pr\{M_i(x) = y \text{ in tape } n\}$ for all such y.

Fix y of length at most $I(i, x, n)$. We construct a Markov chain $\mathcal{M}$ whose states are a trap state FAIL together with all pairs $\langle I, w\rangle$ where I is an ID of $M_i(x)$ in tape n and w is a prefix of y. The number of states of this Markov chain is no more than $2I(i, x, n)^2$. The transition probabilities of the process are defined so that the Markov process simulates those computations of $M_i(x)$ within tape n which produce output y. The one-step transition probabilities of $\mathcal{M}$ are determined by the transition probabilities of M_i:

$$p(\langle I', w'\rangle|\langle I, w\rangle) = \begin{cases} \Pr\{M_i(x) \text{ starting in } I \text{ enters } I' \text{ in the next step without writing}\} & \text{if } w' = w \\ \Pr\{M_i(x) \text{ starting in } I \text{ writes } a \text{ and enters } I' \text{ at the next step}\} & \text{if } w' = wa \end{cases}$$

and

$$p(\text{FAIL}|\langle I, w\rangle) = 1 - \sum_{\langle I', w'\rangle} p(\langle I', w'\rangle|\langle I, w\rangle);$$

$p(\text{FAIL}|\langle I, w\rangle)$ is the probability that $M_i(x)$ starting in I performs in the next step any action inconsistent with giving y as output in a computation within worktape n. Since M_i is a coin-tossing PTM, all transition probabilities are 0, 1, or $\frac{1}{2}$.

The state $\langle\text{BEGIN}, \Lambda\rangle$ of the Markov chain corresponds to the overall state of $M_i(x)$ at the beginning of its computation, and the state $\langle\text{END}, y\rangle$ corresponds to the overall state of $M_i(x)$ at the end of a computation that has produced output y. By the definition of the simulating Markov chain, $\Pr\{M_i(x)=y \text{ in tape } n\}$ equals $\Pr\{\langle\text{BEGIN}, \Lambda\rangle \vdash^* \langle\text{END}, y\rangle\}$, the probability that the simulating process starting in state $\langle\text{BEGIN}, \Lambda\rangle$ eventually reaches state $\langle\text{END}, y\rangle$. We show how to calculate $\Pr\{\langle\text{BEGIN}, \Lambda\rangle \vdash^* \langle\text{END}, y\rangle\}$.

First we determine the states $\langle I, w\rangle$ such that $\Pr\{\langle I, w\rangle \vdash^* \langle\text{END}, y\rangle\} > 0$. These states can be found by computing the transitive closure of the directed graph of states of $\mathcal{M}$. (Finding these states can be accomplished in space bounded by a polynomial in $I(i, x, n)$.) Let the states satisfying this condition be $s_1, s_2, \cdots, s_m$. Let p_{jk} be the transition probability of moving in one step from s_j to s_k and let x_j denote $\Pr\{s_j \vdash^* \langle\text{END}, y\rangle\}$. The probabilities x_j satisfy the system of linear equations

$$(6.10) \qquad x_j = \sum_k p_{jk} x_k + p(\langle\text{END}, y\rangle | s_j) \qquad (j = 1, 2, \cdots, m),$$

which can be rewritten

$$(6.11) \qquad \sum_k 2(p_{jk} - \delta_{jk}) x_k = -2p(\langle\text{END}, y\rangle | s_j) \qquad (j = 1, 2, \cdots, m),$$

a system with coefficients 0, ± 1, ± 2. By eliminating states $\langle I, w\rangle$ such that $\Pr\{\langle I, w\rangle \vdash^* \langle\text{END}, y\rangle\} = 0$, we have guaranteed that this system has a unique solution.

The system above can now be solved for $x_1, x_2, \cdots, x_m$ by brute force. By Cramer's rule, the solution can be written as $N_1/D, N_2/D, \cdots, N_m/D$, where D is the determinant of the coefficient matrix and each N_j is an integer such that $0 < N_j \leqq D$. By expanding the coefficient matrix by minors along rows, we see that $D \leqq 4^m$. Therefore the space required for storing trial solutions is $O(m^2) = O(I(i, x, n)^4)$, and so the solution of the linear system (6.10) can be found in space bounded above by a polynomial of $I(i, x, n)$. Once the system is solved, we have the value of $\Pr\{\langle\text{BEGIN}, \Lambda\rangle \vdash^* \langle\text{END}, y\rangle\}$.

We summarize the proof. For each possible output string y, we compute $\Pr\{M_i(x)=y \text{ in tape } n\}$ by solving a linear system of order $O(I(i, x, n)^2)$. Then $S_i(x) \leqq n$ iff $\Pr\{M_i(x)=y \text{ in tape } n\} > \frac{1}{2}$ for some y of length at most $I(i, x, n)$.

COROLLARY 6.5. *The probabilistic Turing machine M_i can be simulated deterministically in tape $O(I(i, x, S_i(x))^4)$. In particular, if $S_i(x) \geqq \log |x|$ there is a constant c depending on M_i such that M_i can be simulated deterministically in tape c^{S_i}.*

Proof. The output $\phi_i(x)$ is discovered as a by-product of the procedure of Theorem 6.4 for deciding $S_i(x) \leqq n$. A deterministic method for simulating M_i consists of deciding $S_i(x) \leqq n$ for $n = 1, 2, \cdots$, until $\phi_i(x)$ is found. □

LEMMA 6.6. *If $e_i(x)$ denotes the error probability of $M_i(x)$, then $e_i(x) \leqq \frac{1}{2}(1 - 16^{-I(i,x,S_i(x))^2})$. In particular, if $S_i(x) \geqq \log |x|$ then there is a constant c depending on M_i such that $e_i(x) \leqq \frac{1}{2}(1 - 2^{-c^{S_i(x)}})$.*

Proof. From the proof of Theorem 6.4, we can write $\Pr\{M_i(x)=\phi_i(x)$ in tape $S_i(x)\}$ as N/D for some $D \leqq 4^m$, where $m \leqq 2I(i, x, S_i(x))^2$. Thus $D \leqq 16^{I(i,x,S_i(x))^2}$. Since $N/D > \frac{1}{2}$ and N is an integer, $N/D - \frac{1}{2} \geqq 1/(2D)$. Therefore $\Pr\{M_i(x)=\phi_i(x)$ in tape $S_i(x)\} - \frac{1}{2} \geqq \frac{1}{2} \cdot 16^{-I(i,x,S_i(x))^2}$. □

PROPOSITION 6.7. *For each probabilistic Turing machine M_i with run time T_i and tape S_i there is a constant c such that $T_i(x) \leqq c^{I(i,x,S_i(x))}$. In particular, if $S_i(x) \geqq \log|x|$ then there is a constant c such that $T_i(x) \leqq 2^{c^{S_i(x)}}$.*

Proof. For brevity, we write $I(i, x, S_i(x))$ simply as I. It is easily seen that for every k, the probability that $M_i(x)$ halts after more than kI steps is at most $(1-2^{-I})^k$. If $k \geqq (1+\ln 16I^2)2^I$, then $(1-2^{-I})^k < (\frac{1}{2})16^{-I^2}$. Since $M_i(x)$ halts in more than kI steps with probability less than $(\frac{1}{2})16^{-I^2}$, by Lemma 6.6, $\Pr\{M_i(x)=\phi_i(x)$ in time $kI\} > \frac{1}{2}$. Therefore $T_i(x) \leqq kI < c^I$ for some $c > 2$. □

Neither of the bounds of Lemma 6.6 or Proposition 6.7 can be improved significantly. Corollary 6.5 states that deterministic machines require at most exponentially more tape than probabilistic machines. It is an open question whether this exponential bound can be improved. It is also unknown if every probabilistic machine using tape S can be simulated deterministically in tape bounded by a polynomial of S. The next result provides evidence that probabilistic machines might be more efficient in the use of tape than deterministic machines.

PROPOSITION 6.8. *Every language accepted by a nondeterministic Turing machine in tape $S(x) \geqq \log|x|$ can be recognized by a probabilistic Turing machine with bounded error probability in tape $S(x)$.*

Proof. We consider only the case that S is a constructable tape function [11, p. 149]. The general case requires a minor modification [8].

Suppose that L is a language accepted by some nondeterministic machine M in tape $S(x) \geqq \log|x|$. There is a constant c such that every x in L is accepted by some computation of length less than $c^{S(x)}$. Let $d = c + 1$. A probabilistic machine M' that recognizes L operates as follows. With input x:

(i) M' marks off $S(x)$ worktape squares. This requires only tape $S(x)$ because S is constructible.

(ii) M' simulates up to $c^{S(x)}$ steps of a computation of M, choosing the next action by a coin toss when there is a nondeterministic choice. If the computation requires more than $c^{S(x)}$ steps, or attempts to use more than $S(x)$ worktape squares, or halts in the allotted time without accepting, then M' goes to (iii). Otherwise, the simulated computation of M was an accepting computation, and so M' accepts x.

(iii) M' tosses $d^{S(x)}$ coins. If all tosses result in heads, then M' halts and rejects. Otherwise, M' clears its worktapes, rewinds its input tape, and goes back to (ii).

Note that steps (ii) and (iii) can be performed within tape $S(x)$.

Obviously M' rejects inputs not in L with probability 1. If x is in L then any single execution of step (ii) will find an accepting computation with probability at least $2^{-c^{S(x)}}$. Therefore for x in L, we calculate easily that

$$(6.12) \qquad \Pr\{M' \text{ accepts } x\} \geqq 2^{-c^{S(x)}} \sum_{i=0}^{\infty} [(1-2^{-c^{S(x)}})(1-2^{-d^{S(x)}})]^i \geqq \tfrac{2}{3}.$$

Therefore M' accepts L with bounded error probability. □

Acknowledgments. This research was supported by National Science Foundation Grant GK-43121 at Stanford University and was completed at the Forschungsinstitut für Mathematik of the Eidgenössische Technische Hochschule, Zürich. Some of the results appear in the author's doctoral dissertation, supervised by Manuel Blum and supported by National Science Foundation Grant GJ-708 at the University of California, Berkeley. A preliminary version of this paper was presented in [9]. The author thanks the referees for their comments and corrections, and thanks Istvan Simon for helpful discussions.

REFERENCES

[1] M. BLUM, *A machine-independent theory of the complexity of recursive functions*, J. Assoc. Comput. Mach., 14 (1967), pp. 322–336.

[2] H. CHERNOFF, *A measure of asymptotic efficiency for tests based on the sums of observations*, Ann. Math. Statist., 23 (1952), pp. 493–507.

[3] S. A. COOK, *The complexity of theorem-proving procedures*, Proc. 3rd ACM Symp. Theory of Computing, Shaker Heights, OH, 1971, pp. 151–158.

[4] K. DE LEEUW, E. F. MOORE, C. E. SHANNON AND N. SHAPIRO, *Computability by probabilistic machines*, Automata Studies, Annals of Mathematics Studies no. 34, Princeton University Press, Princeton, NJ, 1956, pp. 183–212.

[5] W. FREIBERGER AND U. GRENANDER, *A Short Course in Computational Probability and Statistics*, Applied Mathematical Sciences vol. 6, Springer-Verlag, Berlin, 1971.

[6] R. V. FREIVALD, *Fast computation by probabilistic Turing machines*, Theory of Algorithms and Programs, no. 2, Latvian State University, Riga, 1975, pp. 201–205. (In Russian.)

[7] R. G. GALLAGER, *Information Theory and Reliable Communication*, John Wiley, New York, 1968.

[8] J. T. GILL III, *Probabilistic Turing machines and complexity of computation*, Ph.D. dissertation, Dept. of Mathematics, University of California, Berkeley, 1972.

[9] ———, *Computational complexity of probabilistic Turing machines*, Proc. 6th ACM Symp. Theory of Computing, Seattle, WA, 1974, pp. 91–95.

[10] F. C. HENNIE, *One-tape, off-line Turing machine computations*, Information and Control, 8 (1965), pp. 553–578.

[11] J. E. HOPCROFT AND J. D. ULLMAN, *Formal Languages and their Relation to Automata*, Addison-Wesley, Reading, MA, 1969.

[12] D. J. KLEITMAN, *On a combinatorial conjecture of Erdös*, J. Combinatorial Theory, 1 (1966), pp. 209–214.

[13] A. R. MEYER AND L. J. STOCKMEYER, *The equivalence problem for regular expressions with squaring requires exponential tape*, Proc. 13th IEEE Symp. Switching and Automata Theory, College Park, MD, 1972, pp. 125–129.

[14] G. E. MILLER, *Riemann's hypothesis and tests for primality*, Proc. 7th ACM Symp. Theory of Computing, Albuquerque, NM, 1975, pp. 234–239.

[15] M. MINSKY, *Computation: Finite and Infinite Machines*, Prentice-Hall, Englewood Cliffs, NJ, 1967.

[16] A. PAZ, *Introduction to Probabilistic Automata*, Academic Press, New York, 1971.

[17] W. W. PETERSON AND E. J. WELDON, *Error-Correcting Codes*, second edition, MIT Press, Cambridge, MA, 1972.

[18] M. O. RABIN, *Probabilistic automata*, Information and Control, 6 (1963), pp. 230–245; also in *Sequential Machines*, E. F. Moore, ed., Addison-Wesley, Reading, MA, 1964, pp. 98–114.

[19] ———, *Real-time computation*, Israel J. Math., 1 (1963), pp. 203–211.

[20] ———, *Probabilistic algorithms*, Algorithms and Complexity: New Directions and Recent Results, J. F. Traub, ed., Academic Press, New York, 1976, pp. 21–39.

[21] H. ROGERS, JR., *Gödel numberings of partial recursive functions*, J. Symbolic Logic, 23 (1958), pp. 331–341.

[22] ———, *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, New York, 1967.

[23] E. S. SANTOS, *Probabilistic Turing machines and computability*, Proc. Amer. Math. Soc., 22 (1969), pp. 704–710.

[24] ———, *Computability by probabilistic Turing machines*, Trans. Amer. Math. Soc., 159 (1971), pp. 165–184.

[25] J. SIMON, *On some central problems in computational complexity*, Tech. Rep. TR 75-224, Dept. of Computer Sci., Cornell University, Ithaca, NY, 1975.

[26] R. SOLOVAY AND V. STRASSEN, *A fast Monte-Carlo test for primality*, this Journal, 6 (1977), pp. 84–85.

[27] L. J. STOCKMEYER, *The polynomial-time hierarchy*, Theoretical Computer Science, 3 (1977), pp. 1–22.

[28] B. A. TRAKHTENBROT, *Complexity of algorithms and computation*, Novosibirsk State University, Novosibirsk, 1967. (In Russian.)

[29] ———, *Notes on computational complexity of probabilistic machines*, Theory of Algorithms and Mathematical Logic, Computing Center of the USSR Academy of Sciences, Moscow, 1974, pp. 159–176. (In Russian.)

[30] ———, *On problems solvable by successive trials*, Proc. Symp. Mathematical Foundations of Computer Science, Lecture Notes in Computer Science no. 32, J. Becvar, ed., Springer-Verlag, Berlin, 1975.

[31] A. V. VAISER, *Computational complexity and reliable recognition of languages by probabilistic finite automata*, System Management, no. 1, Tomsk, USSR, 1975, pp. 172–181. (In Russian.)