On the Autoreducibility of Random Sequences

Todd Ebert[*] *Heribert Vollmer*[†]

Abstract

A language $A \subseteq \{0,1\}^*$ is called i.o. autoreducible if A is Turing-reducible to itself via a machine M such that, for infinitely many input words w, M does not query its oracle A about w. We examine the question if algorithmically random languages in the sense of Martin-Löf are i.o. autoreducible. We obtain the somewhat counterintuitive result that every algorithmically random language is polynomial-time i.o. autoreducible where the autoreducing machine poses its queries in a "quasi-nonadaptive" way; however, if in the above definition the "infinitely many" is replaced by "almost all," then every algorithmically random language is not autoreducible in this stronger sense. Further results obtained give upper and lower bounds on the number of queries of the autoreducing machine M and the number of inputs w for which M does not query the oracle about w.

1 Introduction

This paper arose from an investigation of the problem of deducing a property of a random binary sequence when some of the bits of the sequence upon which the property depends are not known. This occurs quite often in practice when, due to time and other resource constraints, a decision is made using only partial information. Not surprisingly, this consideration is closely related to complexity theory, since a decision must be made before a limited resource such as time has been exhausted.

To introduce the question we study in this paper, let us consider the following puzzle.

The N Prisoners Puzzle

A group of N prisoners ($N > 1$) were awaiting parole. The parole committee split on whether or not to grant all of them freedom. When splits occurred, the warden would decide a prisoner's fate by tossing a coin on his desk, covering it with his hand, and having the prisoner guess the side facing up. If the prisoner guessed correctly, he was set free; otherwise he would serve more time. Since they were tried and convicted of the same crime and had displayed equal amounts of good behavior, he used a slightly different approach with the N prisoners. He said to them, "I've tossed N coins onto my desk, one for each of you. Each of you will enter my office one at a time and have the opportunity to guess if your coin, which will be covered by my hand, is showing heads or tails. The other prisoners' coins will be visible to you. You may not ask me any questions whatsoever. Furthermore, after leaving my office, you will be unable to communicate anything to the other prisoners. Not all of you have to guess, but at least one of you must guess. If all the guesses made are correct, then all of you will go free. But if one of you guesses incorrectly, then *all* of you will remain in prison."

[*]DoCoMo Communications Laboratories USA, 250 Cambridge Ave., Palo Alto, Ca 94306 e-mail address: toddebert@dcl.docomo-usa.com

[†]Theoretische Informatik, Universität Würzburg, Am Hubland, 97074 Würzburg, Germany. e-mail address: vollmer@informatik.uni-wuerzburg.de.

At first sight, since every prisoner is unable to observe any event or obtain any information which the outcome of its coin toss is probabilistically dependent on, one expects that the prisoners should have no more than a 50% chance for freedom. However, since they converse before the game, we demonstrate how collaboration can benefit the entire group.

As an example, take $N = 3$ and suppose the prisoners agree to the following guessing strategy. Upon entering the warden's office, each prisoner observes the two unconcealed coins on the warden's desk. If both coins have the same side facing up, then the prisoner guesses his coin is showing the opposite side. However, if the coins have opposite sides facing up, then the prisoner leaves the office without venturing a guess. To compute the chances for freedom under this strategy, two possibilities must be considered. The first case occurs when all three coins have the same side facing up. In this case, each prisoner will incorrectly guess. The second case is two of the three coins show the same side facing up. Here exactly one prisoner will venture a correct guess, and freedom is won. Hence the probability for freedom using the above strategy is $1 - \frac{1}{8} - \frac{1}{8} = \frac{3}{4}$.

The General Problem

We study a more general problem in relativized complexity theory in which an oracle machine M with oracle A has unlimited time and space resources; yet in deciding if word x belongs to the oracle property L^A, machine M may not have access to some of the bits of A. We are particularly interested in the case when $L^A = A$. More precisely, we allow the deterministic oracle machine to use A as oracle set, but for infinitely many x, M may not query the oracle about the bit of A which encodes the answer to whether or not x belongs to A. However, in all cases M has to compute the correct result.

If we imagine that A is generated by the independent tosses of a fair coin, and that the sequence of independent random variables $\{\xi_0, \xi_1, \ldots\}$ represents the coin-toss outcomes, then the *a posteriori* probability of guessing the outcome of ξ_j given knowledge of $\xi_{i_1}, \ldots, \xi_{i_k}$ equals the *a priori* probability of guessing ξ_j provided $j \notin \{i_1, \ldots, i_k\}$. Thus it seems natural to regard the information obtained from observing ξ_i as unhelpful for guessing ξ_j , if $i \neq j$. Hence, at first glance it would seem certain that no oracle machine M with limitations as described above could possibly decide A, since for infinitely many x, M would have to guess the membership of x, and would fail half the time.

On the other hand, by the weak law of large numbers, for all $\epsilon > 0$, $\left\| \frac{\sum_{i=0}^{n} \xi_i}{n} - .5 \right\| < \epsilon$ with arbitrarily high probability when n is sufficiently large. Thus, independent random events considered collectively possess different statistical properties with asymptotically high probability, and the crux of the following investigation rests on determining degrees to which statistical properties may be used to occasionally determine the outcome of a random variable by observing the outcomes of other random variables independent of the one in question. For the scenario described above, we show how error-correcting codes can be used as the means for allowing oracle machines to decide random languages, despite their querying limitations.

Results

To accomplish this we use infinite Martin-Löf random sequences for our random sequence model, and the known concept of autoreduciblity for capturing the idea of using a language A as oracle set in order to decide A, yet not being able to query A about the bit to be decided.

Indeed, a sequence A is called *autoreducible* if A is Turing reducible to itself via an oracle machine M which almost always decides a bit of A without querying A about the bit. If the machine, after one round of nonadaptive queries, is able to either decide its input or recognize that it has to query its oracle about its input, then we say that A is quasi-tt-autoreducible. Moreover, if M infinitely often decides a bit of A without querying the oracle about the bit, then A is called *i.o. autoreducible*. Autoreducibility was first studied by Trakhtenbrot [19], see [20, pp. 483ff], and has since then been used in quite different contexts in recursion theory as well as complexity theory.

For reasons described above, we originally surmised that no Martin-Löf random sequence is i.o. autoreducible. Surprisingly, however, we prove that *every* Martin-Löf random sequence is i.o. quasi-tt-autoreducible, even via an autoreducing machine that runs in polynomial time. This result seems paradoxical in that the machine which witnesses the reducibility has the task of infinitely often guessing a bit of a random sequence, and guessing correctly each time despite a 50% chance of error for each guess. We then show how this result strongly relies on a Turing machine's ability to make an unlimited number of queries on each input. This is accomplished by proving no random sequence is i.o. quasi-btt-autoreducible.

Finally, we introduce the notion of autoreducibility with rate $t(n)$ as a gauge of how often an oracle machine can guess the bits of a random sequence. In words, a sequence is autoreducible with rate $t(n)$ iff it is i.o. autoreducible, and the nth guess by the witnessing oracle machine is for bit j_n, where $j_n \leq t(n)$. We prove that every Martin-Löf random sequence is autoreducible with rate $O(n^2 \log^2 n)$, and that this bound is near optimal in the sense that no Martin-Löf random sequence is autoreducible with rate $O(n)$.

2 Preliminaries and Some Basic Facts

2.1 Words, Languages, and Machines

To begin, $\mathcal{N}$ represents the set of nonnegative integers. We consider words over the binary alphabet $\{0,1\}$, and we assume the words are lexicographically ordered, where s_i, $i \geq 0$, denotes the ith word in the ordering. The set of infinite binary sequences is denoted by $\{0,1\}^\infty$. The set of infinite sequences which have a prefix x in common is denoted by $x\{0,1\}^\infty$, called the *cylinder set generated by x*. There is a natural way of forming a one-to-one correspondence between languages over the alphabet $\{0,1\}$ and infinite binary sequences: $A \subseteq \{0,1\}^*$ is identified with

$$c_A(s_0)c_A(s_1)\cdots,$$

where c_A denotes the characteristic function of A. Furthermore, $A[0\ldots n]$ denotes the first $n+1$ bits of the sequence, while $A(n)$ denotes the nth bit.

We are particularly interested in oracle machines, and we denote $L(M,A)$ as the language decided by Turing machine M using A as oracle set. Every input s_i to an oracle machine induces a binary *query tree* T_{s_i} whose nodes are labeled with query words, and whose leaves are labeled with either YES or NO. The computation of M on input s_i proceeds down the branch corresponding to the answers provided by the oracle. For example, if the root node is labeled with s_1, then M will proceed right if s_1 belongs to oracle set A, and left otherwise. s_i will belong to $L(M,A)$ iff the computation reaches a leaf labeled "YES". $M^A(x)$ will denote the computation of M on input x using oracle A. The computation is said to *halt* iff M traverses a finite branch of the query tree.

2.2 Martin-Löf random sequences

Now assume an effective enumeration $W_0, W_1, \ldots$ of all recursively enumerable sets of finite binary words. We may assume that each W_i is a prefix-free set, i.e. if x is a prefix of y and $x, y \in W_i$, then $x = y$. In fact, there is an effective way to convert every r.e. set W to a prefix-free set W' such that $W\{0,1\}^\infty = W'\{0,1\}^\infty$. With this fact in hand, we define a probability measure Pr on the set of r.e. sets. Indeed, given prefix-free r.e. set W,

$$\Pr(W) = \sum_{x \in W} \frac{1}{2^{|x|}}.$$

Definition 2.1 A set $\mathcal{B} \subset \{0,1\}^\infty$ is called a *constructive null set* iff there exists total recursive $g: \mathcal{N} \to \mathcal{N}$ such that $\mathcal{B} \subseteq \bigcap_{i=0}^{\infty} W_{g(i)}\{0,1\}^\infty$ and $\Pr(W_{g(i)}) < \frac{1}{2^i}$. Here, $\{W_{g(i)}\}_{i \in \mathcal{N}}$ is called a *constructive null cover* for $\mathcal{B}$.

Definition 2.2 The class **NULL** is defined as the union of all constructive null sets, and **RAND** is defined as **RAND** $= \{0,1\}^\infty -$ **NULL**.

$A \in \{0,1\}^\infty$ is called a *Martin-Löf random sequence* iff $A \in$ **RAND**. Since the set of recursive functions is countable, it immediately follows that $\Pr($**RAND**$) = 1$, while $\Pr($**NULL**$) = 0$, since **NULL** is contained in a countable union of sets having measure zero.

The above definition of randomness is attributed to Per Martin-Löf [15], and is widely accepted as having captured the essence of randomness among infinite binary sequences. In working with Martin Löf random sequences it helps to view them as the result of independently tossing an infinite sequence of fair coins; this is true in the sense that, whenever such a sequence is formed, the result must be Martin-Löf random, since **NULL** itself is a constructive null set (of impossible outcomes). Henceforth, by "random sequence" we will mean a Martin-Löf random sequence.

2.3 Autoreducibility

Definition 2.3 A sequence $A \in \{0,1\}^\infty$ is called *autoreducible* iff there exists a deterministic oracle machine M such that $A \leq_T A$ via M, and for almost every x, $x \notin Q(M, A, x)$, where $Q(M, A, x)$ is the set of query words occurring during the computation of $M^A(x)$. Moreover, if "for almost every" is replaced by "infinitely often", then A is called *i.o. autoreducible.*

In a similar way we define *tt-autoreducible*, *btt-autoreducible*, *i.o. tt-autoreducible*, and *i.o. btt-autoreducible.* For example, A is i.o. tt-autoreducible iff A is i.o. autoreducible via some machine M for which query tree T_{s_i} is finite for every $i \in \mathcal{N}$. If $f: \{0,1\}^* \to \mathcal{N}$ is such that $f(s_i)$ is an upper bound for the number of nodes in T_x, then we say that A is i.o. f-tt-autoreducible. Also, A is called i.o. btt-autoreducible iff A is i.o. k-tt-autoreducible for some constant $k \in \mathcal{N}$. From these definitions it immediately follows that every recursive sequence is 0-tt-autoreducible. Slightly more subtle is the observation that every r.e. sequence is i.o. 0-tt-autoreducible. This easily follows from the fact that every infinite r.e. sequence contains an infinite recursive subsequence. On the other hand, it can easily be shown using a standard diagonalization argument that there are are not i.o. autoreducible sequences.

Finally we say that a sequence A is quasi-tt-autoreducible, if A is autoreducible via a machine M that for every input x first, in a nonadaptive manner, poses a number of queries to A, all of which are different from x, and then determines if it can answer if $x \in A$ itself or it has to ask A about x. Only in this latter case, a second round of queries, only consisting of x, is necessary. Hence, these reductions are very near to tt-reductions. For a function f, we use f-quasi-tt-autoreducible, if the number of queries in the first round is bounded by f evaluated for the input length. If f is constant, we use the term quasi-btt-autoreducible.

Now consider infinite random sequences. Intuitively, no random sequence is autoreducible since this would require a deterministic oracle machine to correctly guess every bit of the random sequence. This will follow from a stronger result to be proved later. It would also seem plausible that no random sequence is i.o. autoreducible, since correctly guessing infinitely often without error seems just as implausible. However, we prove a surprising result: *every* random sequence is i.o. autoreducible. Indeed, for any random sequence A there is an oracle machine M^A which infinitely often decides a bit of A by correctly guessing after querying A about bits that come before and after the bit being decided. We will make use of the following well-known result, see [18, p. 255]:

Lemma 2.4 (Borel-Cantelli) Let $\mathcal{X}$ be a probability space with probability measure Pr, and events $E_0, E_1, \ldots$ such that

$$\sum_{i=0}^{\infty} \Pr(E_i) < \infty.$$

Then

$$\Pr\left(\bigcap_{i=0}^{\infty} \bigcup_{j=i}^{\infty} E_j\right) = 0.$$

In other words, with probability 0, an infinite number of E_i will be satisfied.

2.4 Error-Correcting Codes

Definition 2.5 A *perfect one-error-correcting code* is a set $\mathcal{C}_n \subseteq \{0,1\}^n$ such that, for every $x \in \{0,1\}^n$, either $x \in \mathcal{C}_n$ or

$$\exists! y \in \mathcal{C}_n : d(x,y) = 1,$$

where $d(x,y)$ denotes the Hamming-distance between x and y.

It is well known (see, e.g., [11]) that such codes exist for $n+1 = 2^k$, for all $k \geq 0$. This seems intuitively plausible since each code word forms the center of a unit sphere consisting of $n+1 = 2^k$ words, in which the other n words are error words. Since the spheres are mutually disjoint, 2^{n-k} equals the number of spheres, and hence code words. Furthermore, $\mathcal{C}_n$ is a binary $(n-k)$-dimensional vector subspace of $\{0,1\}^n$, and $x \in \mathcal{C}_n$ iff $xH^t = 0$; where H is a $k \times n$ *parity check matrix* for $\mathcal{C}_n$ whose n columns are simply the binary representations of the numbers 1 through n. Thus we have proved the following theorem.

Theorem 2.6 If $n+1 = 2^k$, then there exists a perfect one-error-correcting code $\mathcal{C}_n$ of size 2^{n-k}. Moreover, $\bigcup_{n\in\mathcal{N}} \mathcal{C}_n$ is polynomial-time decidable.

3 Results

Theorem 3.1 Every random sequence is i.o. quasi-tt-autoreducible.

Proof. Define a sequence $a_0 = \frac{1}{2^{n_0}}, a_1 = \frac{1}{2^{n_1}}, \ldots$ so that $\sum_k a_k$ converges. Partition the natural numbers so that the kth partition has size $2^{n_k} - 1$. Notice that, for every $k \in \mathcal{N}$, there is a perfect one-error-correcting code $\mathcal{C}_k$ contained in the set $\{0,1\}^{m_k}$, where $m_k = 2^{n_k} - 1$. We may further assume that the set of codes $\mathcal{C}_0, \mathcal{C}_1, \ldots$ is recursively enumerable.

Next define an oracle machine M in a manner in which M attempts to be a witness for i.o. autoreducibility for an arbitrary sequence A. The strategy M uses involves viewing A as the concatenation of words $v_0, v_1, \ldots$, where $|v_k| = 2^{n_k} - 1$. Furthermore, M assumes that $v_k \notin \mathcal{C}_k$, for every $k \in \mathcal{N}$.

With this in mind, on input s_j, M first determines which partition j belongs to. Assuming that j belongs to the kth partition, M first queries the oracle about all words s_l such that l belongs to the kth partition and $l \neq j$. Thus M determines two words x and y, and knows that the portion of A corresponding to the k th partition is either $w_0 = x0y$ or $w_1 = x1y$, where the uncertainty lies in the jth bit of A, and the size of both w_0 and w_1 is $2^{n_k} - 1$. If neither w_0 nor w_1 belong to $\mathcal{C}_k$, then M queries the oracle about s_j and accepts s_j iff the answer is YES. On the other hand, if $w_0 \in \mathcal{C}_k$, then $w_1 \notin \mathcal{C}_k$, since two code words cannot be a unit distance apart. In this case M accepts s_j, since this is consistent with a one in the jth position, and M is assuming that v_k is an error word. Likewise, M would reject s_j if $w_1 \in \mathcal{C}_k$.

The correctness of this strategy rests on the following

Claim: If $A = v_0 v_1 v_2 \cdots$, where v_k is an error word for code $\mathcal{C}_k$, then A is i.o. quasi-tt-autoreducible via M.

To prove this, consider v_k for arbitrary $k \geq 0$. If v_k is an error word for $\mathcal{C}_k$, then there exists a unique code word w_k such that $d(v_k, w_k) = 1$. Assume that the v_k and w_k differ at bit j, and suppose this bit encodes membership of s_{j_k} in A. Then M will decide bit j_k without querying the oracle about that bit. Indeed, M may query the bits occurring before and after j_k to determine that the portion of A corresponding to partition k is either v_k or w_k. It correctly chooses v_k since it is assuming that A is comprised of the concatenation of error words. Thus, M may decide bit j_k without querying about j_k. For all other bits occurring in the k th partition, M may follow the same procedure, but will find that v_k and w_k are both error words, and hence cannot distinguish between the two. In this case M concedes by querying the oracle about the bit to be decided. Therefore, A is i.o. quasi-tt-autoreducible via M, and the claim is proved.

To finish the proof of the theorem, simply note that a language A will not be quasi-tt-autoreducible via M iff there exists a k such that v_k is a code word. However, if there are only a finite number of such k, then A is i.o. quasi-tt-autoreducible via a machine M' which patches the finite number of mistakes made by M. Moreover, since

$$\Pr(v_k \text{ is a code word}) = \frac{1}{2^{n_k}} = a_k,$$

and $\sum_k a_k$ converges, a finite number of mistakes will occur with probability one (Borel-Cantelli). Finally, notice that the set of sequences $\mathcal{B}$ for which an infinite number of mistakes occur forms a constructive null set. Indeed, letting E_k denote the set of sequences for which v_k is a code word, we have

$$\mathcal{B} = \bigcap_{i=0}^{\infty} (\bigcup_{j=i}^{\infty} E_j),$$

and clearly there exists a recursive function $g: \mathcal{N} \to \mathcal{N}$ such that the expression in parentheses may be expressed as $W_{g(i)}\{0,1\}^{\infty}$. Hence the set is constructive null and every random sequence must be i.o. quasi-tt-autoreducible. □

We note in passing that, by choosing the a_k sufficiently small, we may have M witness i.o. autoreduciblity for a measure of sequences arbitrarily approaching 1.

Returning to the N prisoners puzzle introduced in Sect. 1, from what we have done above, it is obvious that the prisoners can optimize their chance for freedom by assuming the coin sequence forms an error word of an error-correcting code in some dimension $\leq N$. For example, if $N = 7$, then the set of code words forms a 4-dimensional vector subspace; thus there are $128 - 16 = 112$ error words in the space $\{0,1\}^7$, and with probability $\frac{112}{128} = .875$ the prisoners will be set free.

The oracle machine constructed in the proof of Theorem 3.1 has the feature of making an unlimited number of queries to the oracle before deciding membership of a word. Furthermore, if $r(x)$ is the number of queries needed to decide x, then $r(x)$ is a step-like function which grows without bound. Conversely, suppose $r: \{0,1\}^* \to \mathcal{N}$ is an unbounded, increasing recursive function. Using the error-correcting codes as in Theorem 3.1 we choose a set of bits $\{i_1, \ldots, i_n\}$ and assume that the binary vector $A(i_1) \cdots A(i_n)$ is an error-word for some error-correcting code. If $n+1$ is a power of 2 then we will be correct with probability $n/(n+1)$. From the definition of the autoreduction machine in the proof of Theorem 3.1, it is clear that for every k, $1 \leq k \leq n$, the query tree $T(s_{i_k})$ will have n nodes. Since r is unbounded, we may choose $i_1, \ldots, i_k$ in such a way that $r(s_{i_k}) \geq n$ for $1 \leq k \leq n$. Thus, we proved:

Theorem 3.2 Let $r: \{0,1\}^* \to \mathcal{N}$ be an unbounded, increasing recursive function. Then every random sequence is i.o. r-autoreducible.

The just given theorem readily allows us to conclude that the number of queries made by an i.o. autoreducing machine can be made polynomial in the length of the input. Since deciding if a word belongs to an error-correcting code is in polynomial-time we obtain:

Theorem 3.3 Every random sequence is polynomial-time i.o. autoreducible.

As was observed by Wolfgang Merkle [16], the proof of the above results does not really need the full power of constructive null-covers. Specifically it can be shown, using the above idea of stretching the intervals that contain code words far enough, that every set which is not polynomial-time i.o. autoreducible can be covered by a polynomial-time computable martingale. This implies that every p-random set (in the sense of Lutz [14]) is polynomial-time i.o. autoreducible.

The next theorem shows that having $r(x)$ grow without bound is a necessary condition for random sequences to be i.o. autoreducible. To prove this we will need the following lemma whose statement should be intuitively obvious, and whose proof may be found in [9]. The idea is that we should not be able to infinitely often correctly predict without error a bit of a random sequence before the bit is generated. We call any such effective mechanism for infinitely often guessing bits before they are generated a *predictor*. A general study of predictors may be found in [9], along with a proof of the following result.

Lemma 3.4 (Proof by Prediction) Suppose the bits of a sequence A are generated via a recursive permutation. In other words, there is some total recursive $f: \{0,1\}^* \to \mathcal{N}$ such that, if $x \in \{0,1\}^n$ represents the outcome of the first n generated bits, then the next generated bit will decide the value of $A(f(x))$. Furthermore, assume there exists total recursive $g: \{0,1\}^* \to \{-1,0,1\}$ such that, if $x \in \{0,1\}^n$ represents the outcome of the first n generated bits, then $g(x)$ decides if a prediction is made about the next generated bit. Moreover, a prediction is made iff $g(x) \geq 0$. Furthermore, if $g(x) \geq 0$, then $g(x)$ represents the predicted value of the next generated bit. Then if A is random and $g(x) \geq 0$ infinitely often, then infinitely often the predictions made by g are incorrect.

Theorem 3.5 No random sequence is i.o. quasi-btt-autoreducible.

Proof. If A is quasi-btt-autoreducible to B via M, then $A = L(M,B)$ and we may visualize the first round of the quasi-nonadaptive reduction via a query tree which is finite with at most b distinct query words on each tree, for some $b \geq 0$ independent of query tree. Let M be an oracle machine having the above property. We use the proof-by-prediction technique, and generate a recursive permutation of a random sequence A in identical stages. The aim is to show that A is not i.o. quasi-btt-autoreducible via M.

The beginning stage involves searching for a query tree T_{s_i} which has either no branches or a branch not containing s_i. If no such trees exist, then clearly M cannot witness i.o. quasi-btt-autoreducibility for A. So suppose such trees do exist and let T_{s_i} be such a tree. If T_{s_i} has no branches, then bit i of A may be immediately predicted by simply noting the result of the computation of M on input s_i. On the other hand, suppose T_{s_i} has a branch β with nodes labeled with words $s_{j_1}, s_{j_2}, \ldots, s_{j_k}$, where $k \leq b$. We then toss a fair coin k times to determine bits $j_1, \ldots, j_k$ of A. With probability $\geq \frac{1}{2^b}$, the coin-toss sequence will force the computation down branch β, implying that a query about s_i did not occur. Thus, bit i may be predicted by observing the computation result. Next generate bit i and, if a prediction was made about bit i, check to see if it is correct. If yes then proceed to the next stage; otherwise stop, since $A \neq L(M,A)$. If a prediction was not made, then proceed to the next stage.

If a next stage is needed, then update the query trees using the coin-toss results and whatever computation results they induced. We may assume that the coin-toss results induce at most a finite number of computation results, for if this were not the case, then an infinite number of bits of A could be predicted and hence recursively enumerated, which would imply A has an r.e. subsequence, and hence A is not random.

To complete the proof, we claim that the above procedure will terminate after a finite number of stages, and hence A is not i.o. quasi-btt-autoreducible via M. Suppose for the sake of argument that the above procedure does not terminate. Then either a finite number of predictions are made, or an infinite number of predictions are made, and each prediction is correct. The latter case cannot occur by Lemma 3.4. To see why the former case cannot occur, let P_i be the event that a bit of A is predicted during stage i. Then $\{P_i\}_{i\geq 0}$ is a collection of independent events, since, for $i \neq j$, P_i depends on a subset of bits of A disjoint from that which P_j depends upon. Moreover, $\Pr(P_i) \geq \frac{1}{2^b}$. Thus, an infinite number of the P_i will occur. Therefore, since M was arbitrary, A is not i.o. quasi-btt-autoreducible. □

From Theorem 3.1 we know that every random sequence is i.o. autoreducible. An interesting problem involves finding lower bounds to the frequency at which a deterministic oracle machine decides the bit of a random sequence without querying about that bit.

Definition 3.6 Suppose A is i.o. autoreducible via M, and suppose $i_0, i_1, \ldots$ is the increasing sequence of bit locations that M guesses (i.e., in the computation on input s_{i_k}, M does not query the oracle about s_{i_k} for $k = 0, 1, \ldots$), for every $n \in \mathcal{N}$. Then A is called autoreducible with rate $t(n)$ iff $i_n \leq t(n)$, for every $n \in \mathcal{N}$.

Thus, autoreducibility with rate $t(n)$ measures the rate at which M guesses bits of A. This yields a rate of $\frac{n}{t(n)}$ guesses per bits decided. So the question arises as to what is the highest rate a machine may achieve with a random sequence. We may find a partial answer to this in the proof of Theorem 3.1, by noting that $t(n)$ depends on the length of the codewords used for each of the error-correcting codes. In this case the codeword length for the k th code is given by

$$L_k = 2^{n_k} - 1 = \frac{1}{a_k} - 1.$$

Thus, in choosing a slowly converging sequence such as $a_k = \frac{1}{k \log^2 k}$, and letting n_k be the least integer such that 2^{n_k} exceeds $\frac{1}{a_k}$, we see that the n th guess will occur before the first $t(n)$ bits have been decided, where

$$t(n) = \sum_{k=1}^{n} 2k \log^2 k = O(n^2 \log^2 n).$$

And we have proved the following theorem.

Theorem 3.7 Every random sequence is quasi-tt-autoreducible with rate $O(n^2 \log^2 n)$.

The question is still open as to whether or not better guessing rates exist for random sequences. To this end we prove the following theorem, which states that one cannot guess the bits of a random sequence at a constant rate.

Theorem 3.8 No random sequence is quasi-tt-autoreducible with rate $O(n)$.

Proof. The proof is very similar to that of Theorem 3.5. We again begin with an arbitrary oracle machine M and generate random sequence A in stages. It suffices to show that A is not autoreducible with rate cn via M, for positive integer c.

Since the first guessed bit, i_0, is such that

$$i_0 \leq t(0) = c \cdot 0 = 0,$$

it follows that M must guess bit 0 if M witnesses the rate cn autoreducibility of A. Thus, stage 0 proceeds by examining the query tree T_{s_0} of M, and following along a tree branch which is determined by answering each encountered query "$s_n \in A$?" with the n th bit of A if it has already been generated, or by generating the n th bit with a fair coin toss, and then answering accordingly. If during the computation the query "$s_0 \in A$?" arises, then stop: A is not autoreducible with rate cn via M since bit 0 is not guessed. Otherwise, if no such query arises, and the computation halts, then use the result to predict $A(0)$, and generate bit 0 with a coin toss. If the prediction is incorrect, stop: A is not i.o. autoreducible via M. Otherwise proceed to the next stage.

The remaining stages $1, 2, \ldots$ are identical with stage 0, with a few exceptions. For the later stages we in general do not know *a priori* which bit of A will be guessed by M, and hence which query tree we should examine. Hence we must randomly select the next bit b_n to be guessed by M.

To begin the stage n selection, first let J_n denote the total number of bits of A that have been generated in the previous stages. Moreover, choose $K_n > J_n$ so large that

$$\frac{K_n - J_n}{cK_n - J_n} \geq \frac{1}{2c}.$$

We restrict our selection to the first cK_n bits of A. Of these bits, we know that at least $cK_n - J_n$ of them are undetermined, and that if we randomly chose one of these bits using the uniform distribution, then with probability at least

$$\frac{K_n - J_n}{cK_n - J_n} \geq \frac{1}{2c},$$

a bit guessed by M will be selected. Note that this probability is conditioned by the event that A is autoreducible with rate cn via M, and that there is nothing to prove if this condition is false.

Once bit b_n is selected, we examine query tree $T_{s_{b_n}}$ and procede as in stage 0, with the exception that we may not conclude that A is not rate cn autoreducible via M if the query "$s_{b_n} \in A$?" should arise. All we know is that b_n is not a guessed bit.

To finish the proof, we claim that either the above procedure terminates in a finite number of stages, or it will last an infinite number of stages, but only a finite number of predictions are casted. First assume that an infinite number of predictions are casted. Then Lemma 3.4 guarantees that at least one prediction will be incorrect: implying a finite number of stages, and that A is not i.o. autoreducible via M. On the other hand, if the procedure lasts an infinite number of stages, then only a finite number of predictions are casted, which implies that, almost always, the selection of b_n did not yield a bit guessed by M. But this contradicts the fact that at any stage the probability of selecting a guessed bit (on condition that A is autoreducible with rate cn via M) is uniformly bounded away from zero. Therefore, the condition that A is autoreducible with rate cn via M must be false. □

Letting $c = 1$, we have the following corollary.

Corollary 3.9 No random sequence is autoreducible.

4 Concluding Remarks

While we proved i.o. quasi-tt-autoreducibility of Martin-Löf random sequences and shed led on the question of how many queries of a self-reducing machine are actually necessary, the question remains if nonadaptive (i.e, tt-) auto-reducibility can be shown. We close by remarking that if we can show this, we have actually proven P $\neq$ NP, as can be seen using techniques similar to those needed to prove Theorem 6.1 in [7].

Acknowledgments. We are grateful to Klaus W. Wagner (Würzburg), Wolfgang Merkle (Heidelberg), Dieter van Melkebeek (Chicago) and Charles Akemann (UC Santa Barbara) for some very helpful comments. We especially thank Ken Rose (UC Santa Barbara) for his enlightening discussions on error-correcting codes.

References

[1] Alon N., Spencer J. *The Probabilistic Method.* John Wiley and Sons, New York, 1992.

[2] Bennett C., Gill J. *Relative to a Random Oracle A,* $\mathrm{P}^A \neq \mathrm{NP}^A \neq \mathrm{coNP}^A$ *With Probability 1.* SIAM Journal on Computing, 10, 96-113,1981.

[3] Blahut R. *Theory and Practice of Error Control Codes.* Addison Wesley, Reading, Ma, 1984.

[4] Book R. *On Languages Reducible to Algorithmically Random Languages.* SIAM Journal on Computing, Vol. 23, No. 6, 1275-1282, 1984.

[5] Book R., Lutz J., Wagner K. W. *An Observation on Probability Versus Randomness.* Math. Systems Theory, 27,201-209, 1994.

[6] Brualdi R. *Introductory Combinatorics.* North-Holland, NY, 1988.

[7] Buhrman H., van Melkebeek D., Regan K.W., Sivakumar D., Strauss M., A generalization of resource-bounded measure, with an application to the BPP vs. EXP problem. Manuscript, 1999. Preliminary versiond appeared in Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science, pp. 161-171, 1998, and as University of Chicago, Department of Computer Science, Technical Report TR-97-04, May 1997.

[8] Cutland N. *Introduction to Computability.* Cambridge Univ. Press, 1992.

[9] Ebert T. *Applications of Recursive Operators to Randomness and Complexity.* Ph.D. Thesis, University of California at Santa Barbara, 1998.

[10] Gács P. *Every Sequence is Reducible to a Random One.* Information and Control, 70, 186-192, 1986.

[11] Kim K.H., Roush F.W. *Applied Abstract Algebra.* John Wiley and Sons, New York, 1983.

[12] Li M., Vitanyi P. *An Introduction to Kolmogorov Complexity and its Applications.* Springer-Verlag, New York, 1993.

[13] Lutz J. *König's Lemma, Randomness, and the Arithmetical Hierarchy.* Unpublished Lecture Notes, Iowa St. Univ., 1993.

[14] Lutz J. *A Pseudorandom Oracle Characterization of BPP.* SIAM Journal on Computing, 22(5):1075-1086, 1993.

[15] Martin-Löf P. *On the Definition of Random Sequences.* Information and Control, 9, 602-619, 1966.

[16] Wolfgang Merkle. Personal communication, December 1999.

[17] Rogers H. *Theory of Recursive Functions and Effective Computability.* MIT Press, Cambridge, 1992.

[18] Shiryaev A.N. *Probability.* Springer-Verlag, New York, 1995.

[19] Trakhtenbrot, B.A. *On Autoreducibility.* Soviet Math dokl., 11:814-817, 1970.

[20] Wagner K. W., Wechsung G. *Computational Complexity.* Deutscher Verlag der Wissenschaften, Berlin, 1986.