TRANSACTIONS OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 292, Number 2, December 1985

COMBINATORIAL LOWER BOUND ARGUMENTS FOR DETERMINISTIC AND NONDETERMINISTIC TURING MACHINES

BY
WOLFGANG MAASS[1]

ABSTRACT. We introduce new techniques for proving quadratic lower bounds for deterministic and nondeterministic 1-tape Turing machines (all considered Turing machines have an additional one-way input tape). In particular, we derive for the simulation of 2-tape Turing machines by 1-tape Turing machines an optimal quadratic lower bound in the deterministic case and a nearly optimal lower bound in the nondeterministic case. This answers the rather old question whether the computing power of the considered types of Turing machines is significantly increased when more than one tape is used (problem Nos. 1 and 7 in the list of Duris, Galil, Paul, Reischuk [**3**]). Further, we demonstrate a substantial superiority of nondeterminism over determinism and of co-nondeterminism over nondeterminism for 1-tape Turing machines.

1. Introduction and definitions. A major goal of computational complexity theory is the development of methods for proving optimal (or at least significant) lower bounds on the computation time for various abstract computer models. Such lower bounds are needed for the classification of mathematical problems according to their inherent computational complexity. But they are also of practical importance for the development of optimal computer hardware and software. Unfortunately the number of significant lower bound results in (machine based) complexity theory is very small. The "$P = NP$?" question is most prominent, but a closer look shows that nearly every question that requires a nontrivial lower bound argument for a general-purpose computer is open. Many of these questions are not related to nondeterminism, which suggests that there is more missing than just a single "trick". It may be necessary to develop a new mathematical discipline that provides tools for sharp lower bound results.

We introduce in this paper some new combinatorial techniques for proving lower bounds on the computation time of Turing machines that are equipped with one work tape and an additional one-way input tape ("one-way" means that the input head on the input tape can move only in one direction). These techniques allow to settle a rather basic question that is even older than the "$P = NP$?" question: How

Received by the editors July 31, 1984 and, in revised form, February 18, 1985. The results of this paper were presented at the 16th Symposium on the Theory of Computing of the ACM (Washington, May 1984) and at the Special Session on Mathematical Computer Science at the spring meeting of the AMS in Chicago (Chicago, March 23, 1985).

1980 *Mathematics Subject Classification*. Primary 03D15, 03D10, 68C25, 68C40; Secondary 05A17, 94A17.

[1]Written under support by the Heisenberg Programm der Deutschen Forschungsgemeinschaft, Bonn.

much computation time can be saved when the considered Turing machines are equipped with more than one work tape? The motive for this question is obvious. One-tape Turing machines are perhaps the most widely used and simplest models for a general-purpose computer. On the other hand, one notices that certain problems can be solved quite fast on a Turing machine with more than one work tape, whereas all *known* programs for the same problem on a 1-tape Turing machine require substantially more computation time. But although similarly fast programs for 1-tape Turing machines have not been found, one also cannot prove that they do not exist (except for the relatively trivial case of Turing machines without extra input tape, see below).

The first lower bound result concerning the two tapes versus one problem is due to Rabin [**23**], who showed in 1963 that there is *some* difference in power between the two classes of Turing machines (henceforth TM's). He constructed a language that is accepted in real time by a 2-tape TM, but which is not accepted in real time by any 1-tape TM (a machine runs in *real time* if it uses only a constant number of computation steps between reading two successive input bits and halts within the same number of steps after reading the last input bit; thus real time implies linear time). Concerning *upper* bounds, Hartmanis and Stearns [**7**] proved in 1965 that every 2-tape TM that runs in time $t(n)$ can be simulated by a 1-tape TM that runs in time $O(t^2(n))$. This result holds if both machines are deterministic and if both are nondeterministic.

During the following two decades the main contribution towards closing the wide gap between existing upper and lower bounds was made by Wolfgang Paul [**21**]. He had earlier introduced the notion of Kolmogorov complexity into computational complexity theory and he used it in 1982 to show that "on-line simulation" of a real time deterministic 2-tape TM by a deterministic 1-tape TM requires time $\Omega(n \log^{1/2} n)$. The restriction to "on-line simulation" is nontrivial insofar as it prevents the formulation of the result in terms of language acceptance or set theoretic relationships between complexity classes (it only applies to computations with many output bits).

For *nondeterministic* TM's all questions about the influence of the number of tapes on the computing power had been settled except for two tapes versus one (according to Book, Greibach, Wegbreit [**1**] one can simulate for every k a nondeterministic k-tape TM by a nondeterministic 2-tape TM without an increase in computation time). Concerning two tapes versus one Duris and Galil [**2**] proved that a real time deterministic 2-tape TM cannot be simulated by a real time nondeterministic 1-tape TM. In Duris, Galil, Paul, Reischuk [**3**] this lower bound was improved to $\Omega(n \log \log n)$ (apparently they have later achieved $\Omega(n \log n)$).

The main disadvantage of a 1-tape TM is the fact that it needs $\Omega(s \cdot d)$ steps to move on its work tape a string of s symbols over a distance of d cells. A 2-tape TM can perform this operation in only $O(s + d)$ steps. Therefore, one can easily derive a quadratic lower bound for a weak form of 1-tape TM's that do not have an additional input tape (they receive the input on the work tape). Hennie [**8**] showed already in 1965 that this machine needs quadratic time to decide whether an input string is a palindrome. The 1-tape TM with an extra one-way input tape is quite a bit more powerful and can, for example, recognize palindromes in linear time (provided there is a marker in the middle). This is the TM model that has been considered in the previously mentioned lower bound literature and which we will consider in this paper. Such a 1-tape TM has the option to choose a clever data structure for the representation of the input on its work tape. A suitable arrangement of the input on the work tape (note that w.l.o.g. the work tape has several "tracks" and each input bit may be recorded at many different locations on the work tape) makes it perhaps unnecessary to perform during the computation a large number of time-consuming string-moving operations). A clever data arrangement allows, for example, to simulate any k-tape TM that runs in time $t(n)$ by a 2-tape TM that runs in time $O(t(n) \cdot \log t(n))$ (Hennie and Stearns [**9**]). In view of these facts and because of the virtually unlimited number of possibilities for representing the input on the work tape, the correct value of the optimal lower bound for two tapes versus one was somewhat dubious.

In addition, in the case of two tapes versus one, for *nondeterministic* TM's one has to be aware that nondeterministic 1-tape TM's are quite powerful: they accept in linear time NP-complete problems like 3-COLORABILITY. From the technical point of view, such machines have the additional option to use in a nonuniform way "individualized" data structures for the representation of the input on their work tape that are helpful only for a particular computation on a particular input.

We construct in this paper the language L of "polydromes" which extends the well-known language of palindromes. This language L is accepted in linear (even real) time by a deterministic 2-tape TM, but every 1-tape TM that accepts L uses $\Omega(n^2)$ steps (Theorem 2.1). This solves the two tapes versus one problem and shows that the Hartmanis/Stearns simulation [7] is in fact optimal for the considered two classes of Turing machines.

In §3 we define a slight extension L_I ("iterated polydromes") of the language L. We show that L_I is accepted in real time by a deterministic 2-tape TM, whereas no *nondeterministic* 1-tape TM accepts L_I in time $o(n^2/\log^2 n \log\log n)$ (Theorem 3.3). This shows that for nondeterministic Turing machines the Hartmanis/Stearns simulation is at least nearly optimal.

The lower bound arguments of this paper also yield new results concerning the question whether nondeterministic machines are more powerful than deterministic machines, and whether co-nondeterministic machines are more powerful than nondeterministic machines. We write $\mathrm{DTIME}_k(t(n))$, $\mathrm{NTIME}_k(t(n))$ for the classes of languages that are accepted in time $O(t(n))$ by deterministic, respectively nondeterministic, k-tape TM's (always with an additional one-way input tape). CO-$\mathrm{NTIME}_k(t(n))$ consists of those languages whose complement is in $\mathrm{NTIME}_k(t(n))$. The known separation results are

$$\mathrm{NTIME}_2(n) \not\subseteq \bigcup_{k \geq 1} \mathrm{DTIME}_k\left(n(\log^* n)^{1/4}\right)$$

(Paul, Pippenger, Szemeredi and Trotter [**22**]),

$$\mathrm{NTIME}_2(n) \not\subseteq \mathrm{DTIME}_1(n^{1.1})$$

(Kannan [**11**]) and

$$\mathrm{NTIME}_2(n) \not\subseteq \mathrm{DTIME}_1(n^{1.22})$$

(Maass and Schorr [**22**]; here the lower bound also holds for 1-tape TM's with two-way input tape).

We show in §4 of this paper that

$$\mathrm{NTIME}_1(n) \not\subseteq \mathrm{DTIME}_1(o(n^2)) \quad \text{(Theorem 4.1)}$$

and

$$\text{CO-NTIME}_1(n) \not\subseteq \mathrm{NTIME}_1(o(n^2/\log^2 n \log\log n)) \quad \text{(Theorem 4.2)}$$

(note that these classes arise from machines that only differ in their control structures, not in their number of tapes).

The proofs of this paper rely on combinatorial arguments in combination with the notion of Kolmogorov complexity. It turns out that if a binary sequence X is "incompressible" (i.e. X has Kolomogorov complexity $K(X) \geqslant |X|$) then one can handle the information in X like a given set of elements in combinatorics. In particular, the information pieces that are contained in X are partitioned and rearranged as if they were (static) elements of a given sets. This is possible because, due to the incompressibility of X, no information piece from X can disappear (without leaving a trace which has the same number of bits) and reappear later. The "Desert Lemmata" (Lemma 2.2 and Lemma 3.4) may be viewed as somewhat complicated applications of the pigeon hole principle. They assert that, however the information contained in X is stored on the work tape of a 1-tape TM, there are two regions ("pigeon holes") on the tape that are separated by a long interval (the "desert"), such that for each of these two regions there are *large* portions of information about X of which only this region has knowledge.

It is often instructive to view a lower bound argument as the construction of a winning strategy in a 2 person game (where the opponent claims that he can beat the lower bound). We have made this view more explicit in a recent survey paper [**16**]. Compared with previously existing lower bound arguments the opponent has in our case a large amount of freedom (for example, in the way he arranges the input on the work tape). Therefore, in order to beat him one has to find a sufficiently general "weak point" that is an invariant of all reasonable strategies of the opponent (provided here by the "Desert Lemmata").

In §3 we study a finitary analogue of a well-known infinitary structure (from ergodic theory): We demonstrate in Theorem 3.1 with a purely combinatorial proof a strong ergodicity property of the finitary version of the "doubling transformation" $T(x) = 2x \pmod 1$ which maps the interval $[0, 1)$ of real numbers into itself. We use this strong ergodicity property to construct *one* input that beats all possible strategies of the opponent (this is necessary in the nondeterministic case). This combinatorial result appears to be also of some independent interest (we discuss relationships to expander graphs in §3).

We have tried to make this paper largely self-contained. In particular, all relevant definitions are given at the end of this section.

Slightly weaker versions of the results in this paper were first publicized in [**14** and **15**]. A discussion of some recursion theoretic aspects of our new lower bound arguments was given in §4 of [**16**]. Sketches of the proofs were published in an extended abstract [**17**]. Subsequently, Ming Li [**13**] derived a quadratic lower bound for one tape versus two stacks. Paul Vitanyi [**24**] proved quadratic lower bounds for "on-line simulation" (see discussion above) of one queue or two pushdown stores by one tape (we learnt that in addition Vitanyi had reported at ICALP 83 a $n^{1.5}$ lower bound for on-line simulation of one pushdown store by an oblivious 1-tape TM).

Recently Zvi Galil suggested an alternative to our construction of a suitable input Z in §3: Instead of our linear graph with weaker expansion properties one might use his two-dimensional expander graph from [**4**] and realize the edges of it with $\log k$ sweeps over X (apparently this would give about the same lower bound as Theorem 3.3).

The following definitions are used in this paper. All work tapes of Turing machines are two-way infinite. For simplicity we assume that all Turing machines use only binary tape and input symbols (all other symbols are coded). A *nondeterministic* TM is characterized by the fact that its transition function is multiple-valued. A nondeterministic TM *accepts* an input if it has some computation path for this input that ends in an accepting final state. A *language* is simply a set of words over some fixed alphabet. A TM *accepts* a *language* L if it accepts exactly those words over the corresponding alphabet that belong to L. A (deterministic or nondeterministic) TM runs in *time* $t(n)$ if no computation path for an input of length n has more than $t(n)$ steps. We refer to Hopcroft and Ullman [**10**] for further details about Turing machines (this book also contains the mentioned simulation results [**7** and **9**]).

In order to define the Kolmogorov complexity of a binary sequence X, we assume that all Turing machines $\tilde{M}$ (with any number of tapes) are coded in some fixed way by binary sequences. We write $|\tilde{M}|$ for the length of the binary sequence that codes the Turing machine $\tilde{M}$. One defines the Kolmogorov complexity of X relative to another binary sequence Y by

$$K(X|Y) := \min\left\{ |\tilde{M}| \,\middle|\, \tilde{M} \text{ is a deterministic Turing machine that} \right.$$
$$\left. \text{produces from input } Y \text{ the output } X \right\}.$$

The Kolmogorov complexity of X is defined by $K(X) := K(X|\varepsilon)$, where ε is the empty sequence.

It is obvious that for every natural number n there is a binary sequence X of length n with $K(X) \geqslant n$ (because there are only $2^n - 1$ Turing machines $\tilde{M}$ with $|\tilde{M}| < n$).

The language L that we will consider in this paper extends the well-known language of palindromes. Words in L represent the running of *two* independent heads over a recorded sequence X. Therefore we call them "polydromes" (poly (Greek) = many, dromos (Greek) = a running). We define L first in terms of command sequences for a "virtual" 2-tape TM M'. A more concise but somewhat less intuitive definition is given afterwards. The virtual TM M' interprets each input symbol from $\{0,1,2,3\}$ as a command. M' starts in "writing mode" and copies all binary symbols at the beginning of the input onto both work tapes (from left to right). As soon as M' encounters a symbol from $\{2,3\}$ on the input tape, it changes for the rest of the computation into its "testing mode". M' interprets 2 (3) as the command to move work head 1 (2) one cell to the left. In the testing mode M' interprets a symbol $y \in \{0,1\}$ as the command to test whether that work head which moved last reads the same symbol y in the currently scanned cell. The complete input Y is in L if and only if all these tests have a positive outcome.

It is obvious from this definition that L is accepted in real time by the 2-tape TM M'.

This is another—purely combinatorial—definition of L:

$$L := \Big\{ x_1 \cdots x_n z_1 \cdots z_k | x_1, \dots, x_n \in \{0,1\} \wedge z_1 \in \{2,3\} \wedge \forall j \in \{2,\dots,k\}\Big(z_j \in \{0,1\} \to \Big(z_{j-1} \in \{2,3\} \text{ and } z_j = x_{n+1-m}, \text{ where } m := \Big|\Big\{ j' \leqslant j-1 | z_{j'} = z_{j-1}\Big\}\Big| \leqslant n\Big)\Big)\Big\}.$$

As an example for words in L, we note that a binary string $x_1 \cdots x_n y_1 \cdots y_n$ is a palindrome if and only if the word $x_1 \cdots x_n 2y_1 \cdots 2y_n$ is in L.

For the lower bound argument in the deterministic case (§2), we will consider the following words $Y_{X,\tilde{L},\tilde{R}}$ from L. Let X be a binary sequence and let $\tilde{L} = \{l_1, l_2, \dots\}$ and $\tilde{R} = \{r_1, r_2, \dots\}$ be two arbitrary sets of subsequences of consecutive bits ("blocks") from X. We assume that each block has length p and that the blocks in $\tilde{L}$ and $\tilde{R}$ are listed in the order of their appearance in X from right to left. Let $l_{i,1} \cdots l_{i,p}$ and $r_{i,1} \cdots r_{i,p}$ be the symbols of block l_i, respectively r_i, in the order from right to left. Let $d_{\tilde{L}}(i)$ ($d_{\tilde{R}}(i)$) be the number of bits between blocks l_i and l_{i+1} (r_i and r_{i+1}) in X. Further, let $d_{\tilde{L}}(0)$ ($d_{\tilde{R}}(0)$) be the number of bits in X to the right of block l_1 (r_1). Then the following string $Y_{X,\tilde{L},\tilde{R}}$ is in L:

$$Y_{X,\tilde{L},\tilde{R}} := x_1 \cdots x_n \underbrace{2 \cdots 2}_{d_L(0) \text{ times}} 2l_{1,1}2l_{1,2} \cdots 2l_{1,p} \underbrace{3 \cdots 3}_{d_R(0) \text{ times}} 3r_{1,1}3r_{1,2} \cdots 3r_{1,p}$$

$$\underbrace{2 \cdots 2}_{d_L(1) \text{ times}} 2l_{2,1}2l_{2,2} \cdots 2l_{2,p} \underbrace{3 \cdots 3}_{d_R(1) \text{ times}} 3r_{2,1}3r_{2,2} \cdots 3r_{2,p} \cdots \text{etc.}$$

(alternating through all blocks of $\tilde{L}$ and $\tilde{R}$).

2. Two tapes versus one for deterministic Turing machines.

THEOREM 2.1. *The language L of polydromes is accepted in linear* (*even real*) *time by a deterministic* 2-*tape Turing machine, but any deterministic* 1-*tape Turing machine that accepts L requires time* $\Omega(n^2)$ (*a reminder*: *all Turing machines in this paper have an additional one-way input tape*).

PROOF. According to §1 it is obvious that the language L is accepted in real time by some deterministic 2-tape TM. Assume for a contradiction that there is a deterministic 1-tape TM M that accepts L but does not require time $\Omega(n^2)$. This implies that for every natural number c there is an infinite set N_c of natural numbers such that M accepts every word $Y \in L$ with $|Y| \in N_c$ in at most $|Y|^2/c$ steps (note that the considered assumption does *not* imply that M runs in time $o(n^2)$). We fix M for the following and we choose c large enough (this will be made more precise in the following). For convenience we assume that c is chosen in such a way that $c^{1/6}$ is a natural number.

In order to get a contradiction we construct a word $X^\frown Z \in L$ such that $|X^\frown Z| \in N_c$ and M needs more than $|X^\frown Z|^2/c$ steps to accept the input $X^\frown Z$. We fix a large enough number $n_c \in N_c$ (precise conditions on the size of n_c will come out of the following arguments). We set $n := \lfloor n_c/8 \rfloor$ and choose a binary string X of length n with $K(X) \geqslant n$ (see §1 for the definition of the Kolmogorov complexity $K(X)$). With the help of padding it will be easy to arrange that the length of the constructed input $X^\frown Z$ is equal to n_c (we assume for simplicity that all considered Turing machines only use binary symbols, thus every symbol in $\{0,1,2,3\}$ requires 2 bits). From an intuitive point of view this choice of X guarantees that X looks like a random number to M (provided that $n \gg |M|$). In particular, M will not be able to deduce from knowledge about some parts of X any significant information about the rest of X.

Since M is *deterministic*, it processes the first part X of the input $X^\frown Z$ independently of the second part Z. Therefore, we can wait with the definition of Z until M has processed X. Further we can define this "test sequence" Z in such a way that it exploits particular "weak points" of this first part of M's computation. The following Lemma 2.2 shows that because of the linear structure of the single work tape of M there will always be *some* weak point in the first part of the computation.

For the rest of this proof we partition the string X into $\tilde{n} := \lceil n/c^{1/3} \rceil$ blocks of length $c^{1/3}$ (the last block may be shorter).

LEMMA 2.2 ("DESERT LEMMA"). *If n is large enough, then there is an interval D of $\tilde{n}$ cells (the "desert") on the work tape of M and there are two sets L_D and R_D which each contain $\lfloor \tilde{n}/2 \rfloor - 2\lceil n/c^{1/2} \rceil$ blocks from X, such that the work head of M is always left (right) of D during those steps where the input head reads from a block in X that belongs to L_D (R_D).*

PROOF OF LEMMA 2.2. There are at most $n/c^{1/2}$ blocks B in X such that the work head of M moves over $\tilde{n}$ or more cells, while the input head reads B (if c is large enough then $\lceil n/c^{1/2} \rceil \cdot \tilde{n} \geqslant n^2/c^{5/6} > n_c^2/c$). Therefore, it will be sufficient to find an interval D_0 of $3\tilde{n}$ cells on the work tape of M with the following two properties (define D to be the middle third of D_0):

(i) there are at least $\lfloor \tilde{n}/2 \rfloor - \lceil n/c^{1/2} \rceil$ blocks B such that the work head is at *some* step left of D_0 while the input head reads from B,

(ii) there are at least $\lfloor \tilde{n}/2 \rfloor - \lceil n/c^{1/2} \rceil$ blocks B such that the work head is at *some* step right of D_0 while the input head reads from B.

We define D_0 to be the leftmost interval of length $3\tilde{n}$ on the work tape of M that satisfies (i). Assume for a contradiction that this interval D_0 does not have property (ii). Define another interval D_1 of length $3\tilde{n} + 1$ that consists of D_0 together with the next cell to the left of D_0. By definition of D_1 there is a set T of $2 \cdot \ulcorner n/c^{1/2} \urcorner$ blocks B such that the work head of M is always inside D_1 at those steps where the input head looks at B. Let c_L (c_R) be a cell that lies between 1 and $\tilde{n}$ cells to the left (right) of D_1 such that the work head of M scans c_L (c_R) during at most $\tilde{n}$ steps (such cells exist because for c large enough one has $\tilde{n}^2 \geqslant n^2/c^{2/3} > n_c^2/c$). Let S_L (S_R) be the crossing sequence for cell c_L (c_R) which records for every crossing of this cell the current machine state and (in binary code) the number of cells by which the input head has advanced since the last crossing. Both crossing sequences require only $\tilde{n} \cdot (s_M + 2 \cdot \ulcorner \log c \urcorner)$ bits, where the constant s_M depends only on the number of states of machine M. We use at this point the fact that, according to Galil [5], for any sequence $m_1, \ldots, m_{\tilde{n}}$ of natural numbers with $\sum_{j=1}^{\tilde{n}} m_j = n$ the convexity of the log-function implies that

$$\sum_{j=1}^{\tilde{n}} (1 + \log m_j) \leqslant \tilde{n} \cdot (1 + \log n/\tilde{n}) \leqslant \tilde{n} \cdot (1 + \log c).$$

We define X^C to be a variation of string X where all blocks that belong to T are "censored", i.e. replaced by equally long sequences of a new symbol ■. Let I be the inscription on the segment of M's work tape between the cells c_L and c_R at that step t_0 where the input head moves off the last symbol of X (if the input head of M never moves so far, one gets trivially a contradiction). Let $T^\frown$ be the concatenation of all blocks in T (in the order from left to right as they appear in X). We define an auxiliary TM P (P can have an arbitrary number of tapes) that produces the output $T^\frown$ from an input that consists of X^C, I, S_L, S_R, the cell numbers of c_L and c_R, t_0, the state and head positions of M (on an input that starts with X) at step t_0. P considers successively all binary sequences U that have the same length as $T^\frown$. For each such sequence U it replaces the censored parts in X^C by U (call the resulting binary sequence $X^C[U]$). Then P simulates M on input $X^C[U]$ until step t_0. P outputs the sequence U if it finds that TM M on input $X^C[U]$ generates on the cells c_L and c_R the crossing sequences S_L, respectively S_R, produces between c_L and c_R at step t_0 the inscription I and reaches at step t_0 the same state and head positions as M on input X. Otherwise P tests in the same manner the lexicographically next string after U.

The constructed TM P may use exponentially many computation steps, but it will always output some string U (because the string $T^\frown$ has all the properties that are expected from U). We assume for a contradiction that P outputs a string U that differs from $T^\frown$. At this point we have to give attention to the fact that P does not test whether the input $X^C[U]$ produces at step t_0 also *outside* of the interval (c_L, c_R) on the work tape of M the same inscription as the input X (P cannot test this because it would then need as part of its input the corresponding inscription for input X; this would make P's input too long for our purposes). At this point it becomes crucial that the crossing sequences S_L and S_R did not just (like usual crossing sequences) record for each crossing the current machine state, but in addition the current position of the input head. We know (from the definition of X^C) that at those steps where its work head is outside of the interval (c_L, c_R), the

TM M on input X reads only those parts of X where X and X^C (and therefore X and $X^C[U]$) agree. Since M on input $X^C[U]$ also produces the crossing sequences S_L, S_R on cell c_L, respectively c_R, the same fact holds for M on input $X^C[U]$. Thus whenever the work head of M (for input $X^C[U]$) is outside of the interval (c_L, c_R), M only processes those parts of the input where X and $X^C[U]$ agree. Therefore, on input X and on input $X^C[U]$ the machine M produces automatically at step t_0 the same tape inscription outside of (c_L, c_R). Since M arrives at step t_0 for both of the inputs X and $X^C[U]$ not only with the same tape inscription but also in the same state and with the same head positions, we can easily fool M in the following way: We set $\tilde{Z} := 2x_n 2x_{n-1} \cdots 2x_1$ (where $x_1 \cdots x_n = X$). Obviously, $X^\frown \tilde{Z} \in L$ and $X^C[U]^\frown \tilde{Z} \notin L$ (since $X \neq X^C[U]$). Since M accepts $X^\frown \tilde{Z}$ and since M reaches on input $X^C[U]^\frown \tilde{Z}$ at step t_0 the same configuration as on input $X^\frown \tilde{Z}$, M also accepts $X^C[U]^\frown \tilde{Z}$. This contradiction implies that $U = T^\frown$.

The existence of the previously described TM P implies that

$$(1) \qquad K(T^\frown | X^C) \leqslant K_M \cdot \log c \cdot n/c^{1/3}$$

for some constant K_M that depends only on machine M.

On the other hand, if some TM produces (without input) the output X^C and if another TM produces on input X^C the output $T^\frown$, we can combine both TM's into one TM that produces (without input) the output X. Therefore (we ignore additive constants in our estimates)

$$(2) \qquad n \leqslant K(X) \leqslant K(X^C) + K(T^\frown | X^C).$$

Further one gets

$$(3) \qquad K(X^C) \leqslant n - 2 \cdot n \cdot c^{1/3}/c^{1/2} + 4 \cdot n \cdot (1 + \log c)/c^{1/2}.$$

We use here that X^C contains $2 \cdot \lceil n/c^{1/2} \rceil \cdot c^{1/3}$ censored bits and that we can describe the location of all censored blocks by giving their distances in binary code (the sum of distances between consecutive censored blocks is at most n, thus one can use the same trick as for the estimate of the lengths of S_L, S_R before).

The combination of (2) and (3) yields

$$(4) \qquad K(T^\frown | X^C) \geqslant 2 \cdot n/c^{1/6} - 4 \cdot n \cdot (1 + \log c)/c^{1/2}.$$

Finally, from (1) and (4) we get

$$(5) \qquad 2/c^{1/6} \leqslant K_M \cdot \log c/c^{1/3} + 4 \cdot (1 + \log c)/c^{1/2}.$$

Thus in order to get the desired contradiction we just have to choose c large enough so that (5) becomes wrong. This finishes the proof of Lemma 2.2.

For the following we fix a desert D and two sets L_D, R_D of $\lfloor \tilde{n}/2 \rfloor - 2 \cdot \lceil n/c^{1/2} \rceil$ blocks from X as in Lemma 2.2. We define the second part Z of the constructed input so that $X^\frown Z = Y_{X, L_D, R_D}$ (see §1 for the definition of $Y_{X, \tilde{L}, \tilde{R}}$ for arbitrary sets $\tilde{L}$, $\tilde{R}$ of blocks). This "test sequence" Z forces M to check all blocks of L_D and R_D in alternation. We call each segment of Z, where one checks in immediate succession a block from L_D and a block from R_D, an $L_D - R_D$ pair. For large enough c one has $\lfloor \tilde{n}/2 \rfloor - 2 \cdot \lceil n/c^{1/2} \rceil \geqslant \lceil \tilde{n}/4 \rceil$. Thus we can assume that Z contains at least $\lceil \tilde{n}/4 \rceil L_D - R_D$ pairs, where one first checks a block from L_D and immediately afterwards a block from R_D.

From an intuitive point of view, it is plausible that the considered TM M is not able to process the constructed input $X^\cap Z$ within the given time bound. Because of the length of the desert, M cannot transfer a large amount of information about blocks in L_D or R_D across the desert without exceeding its time bound. Therefore, for most of the $L_D - R_D$ pairs in Z, where some blocks $b_1 \in L_D$ and $b_2 \in R_D$ are checked in immediate succession, the work head of M has to move close to that area of the tape where it had originally recorded information about these blocks. Thus it has to move close to the left end of desert D in order to check b_1 and close to the right end of desert D in order to check b_2. The following Lemma 2.3 verifies this intuition.

We write D_l, D_m, D_r for the left, middle, respectively right third, of desert D.

LEMMA 2.3. *For at least* $2/3$ *of the* $L_D - R_D$ *pairs in* Z *the work head of* M *touches a cell left of* D_m *during those steps where* M*'s input head reads from that* $L_D - R_D$ *pair.*

PROOF OF LEMMA 2.3. Assume for a contradiction that for at least $1/3$ of the $L_D - R_D$ pairs in Z the work head of M stays to the right of D_l at all steps where the input head reads from that $L_D - R_D$ pair. Let $\tilde{L}$ be a set of $\ulcorner \tilde{n}/12 \urcorner$ blocks from L_D that occur in these $L_D - R_D$ pairs. Let $X^{C\cap}Z^C$ be a variation of the input where all blocks that belong to $\tilde{L}$ have been censored (i.e. replaced by equally long sequences of a new symbol ■). Note that in Z the bits from each block B of $\tilde{L}$ occur in alternation with the symbol "2". In Z^C these symbols "2" remain unchanged while the bits from B are replaced by ■.

Let c_l be a cell in D_l that is scanned during at most $\tilde{n}$ steps (such cell c_l exists because we assume that c is so large that $\tilde{n}^2/3$ exceeds the time bound $|X^\cap Z|^2/c$). Let S_l be a crossing sequence for cell c_l that records for each crossing the current state and input head position (analogously as S_L, S_R in Lemma 2.2). S_l requires at most $\tilde{n} \cdot s_M \cdot \log c$ bits (same argument as in Lemma 2.2), where the constant s_M depends only on machine M.

Let $\tilde{L}^\cap$ be the concatenation of all blocks in $\tilde{L}$ (in the order as they appear in X). We construct an auxiliary TM $\tilde{P}$ that computes $\tilde{L}^\cap$ from $X^{C\cap}Z^C$, the number of cell c_l, S_l and the final state and head positions of TM M on input $X^\cap Z$. Analogously as the TM P in the proof of Lemma 2.2 this machine $\tilde{P}$ tries successively all binary sequences V of length $|\tilde{L}^\cap|$. $\tilde{P}$ simulates for each such sequence V the TM M on input $(X^{C\cap}Z^C)[V]$ (the latter term is defined in the obvious way; in particular, $(X^{C\cap}Z^C)[\tilde{L}^\cap] = X^\cap Z$). $\tilde{P}$ outputs the first string V such that M on input $(X^{C\cap}Z^C)[V]$ generates on cell c_l the crossing sequence S_l and halts with the same state and head positions as M on input $X^\cap Z$. Obviously the string $V = \tilde{L}^\cap$ has all these properties and thus $\tilde{P}$ produces *some* output. Assume for a contradiction that $\tilde{P}$ produces an output $V \neq \tilde{L}^\cap$. We apply a "cut and paste" argument where we cut the computations of M on input $X^\cap Z$ and on input $(X^{C\cap}Z^C)[V]$ at all those points where the work head scans cell c_l. Then we combine all those parts of M's computation on input $X^\cap Z$ where the work head is left of cell c_l with all those parts of M's computation on input $(X^{C\cap}Z^C)[V]$ where the work head is not to the left of cell c_l (this is possible because both computations generate on cell c_l the same crossing sequence S_l). The constructed combination yields an accepting computation of TM M on input $X^\frown Z^C[V]$. We use at this point that M on input $X^\frown Z$ reads, during those phases of the computation where its work head is left of cell c_l, only those parts of the input where $X^\frown Z$ and $X^\frown Z^C[V]$ agree (this follows from the definition of $\tilde{L}$ and of cell c_l). Further, since $\tilde{L} \subseteq L_D$ and since the crossing sequence S_l records the current position of the input head for each crossing of c_l, M on input $(X^{C\frown}Z^C)[V]$ reads, during those phases of the computation where its work head is not to the left of c_l, only those parts of the input where $(X^{C\frown}Z^C)[V]$ and $X^\frown Z^C[V]$ agree. The resulting accepting computation of M on input $X^\frown Z^C[V]$ yields a contradiction, since our assumption $V \neq \tilde{L}^\frown$ implies that $X^\frown Z^C[V] \notin L$. Thus we have shown that TM $\tilde{P}$ delivers the desired output $\tilde{L}^\frown$.

We now proceed similarly as in the proof of Lemma 2.2. The existence of TM $\tilde{P}$ implies that (for sufficiently large c)

$$K(\tilde{L}^\frown | X^{C\frown}Z^C) \leqslant 2 \cdot s_M \cdot \log c \cdot n/c^{1/3} \leqslant n/48. \tag{6}$$

On the other hand

$$K(X^{C\frown}Z^C | X^C) \leqslant n \cdot (1 + \log c)/c^{1/3}, \tag{7}$$

because we can describe the locations of those blocks in X which are tested in Z with the help of their distances (in binary code). Further (in analogy to (2) and (3) in Lemma 2.2) we have

$$\begin{aligned} n \leqslant K(X) &\leqslant K(X^C) + K(\tilde{L}^\frown | X^C) \\ &\leqslant n - n/12 + n \cdot (1 + \log c)/c^{1/3} + K(\tilde{L}^\frown | X^C). \end{aligned}$$

This implies

$$K(\tilde{L}^\frown | X^C) \geqslant n/12 - n \cdot (1 + \log c)/c^{1/3}. \tag{8}$$

Finally it is obvious that

$$K(\tilde{L}^\frown | X^C) \leqslant K(X^{C\frown}Z^C | X^C) + K(\tilde{L}^\frown | X^{C\frown}Z^C). \tag{9}$$

From (7), (8) and (9) we get

$$\begin{aligned} K(\tilde{L}^\frown | X^{C\frown}Z^C) &\geqslant K(\tilde{L}^\frown | X^C) - K(X^{C\frown}Z^C | X^C) \\ &\geqslant n/12 - 2 \cdot n \cdot (1 + \log c)/c^{1/3} \geqslant n/24 \end{aligned}$$

(for sufficiently large c). This contradicts (6) and thus the proof of Lemma 2.3 is complete.

The same argument as in Lemma 2.3 shows that for at least $2/3$ of the $L_D - R_D$ pairs in Z the work head of M touches a cell to the right of the middle third D_m of desert D while M's input head reads from that pair. This fact in combination with Lemma 2.3 implies that for at least $1/3$ of the $L_D - R_D$ pairs in Z the work head of M crosses all of D_m during those steps where M's input head reads from that pair. Thus M uses on input $X^\frown Z$ at least $1/3 \cdot \tilde{n}/4 \cdot \tilde{n}/4 \geqslant n^2/(36 \cdot c^{2/3})$ computation steps. For large enough c this exceeds M's time bound of $|X^\frown Z|^2/c \leqslant 81 \cdot n^2/c$ steps. This contradiction completes the proof of Theorem 2.1.

3. Two tapes versus one for nondeterministic Turing machines and a helpful combinatorial result. The language L_I of "iterated polydromes" that we will consider for the nondeterministic case is a slight extension of our previously discussed language L. We introduce a new command "4" for the virtual 2-tape TM M' from §1 that tells M' to change the direction of movement for both of its work heads (this new command will only be used when M' is in its testing phase). Whereas in language L the command "2" ("3") always required M' to move work head 1 (2) one cell to the left, the same command instructs M' in the language L_I to move work head 1 (2) one cell to the *left*, if the number of preceding commands "4" is *even*, and one cell to the *right*, otherwise. With the help of this new command "4" we can instruct the 2-tape TM M' to perform during its testing phase *several* sweeps over its work tapes. Analogously as for L, a word Y over $\{0, 1, 2, 3, 4\}$ is in L_I if and only if all "tests" in Y have a positive outcome (where Y is interpreted as command sequence for the 2-tape TM M'). The equivalent purely combinatorial definition of the language L_I is obvious but tedious (see [**17**]).

We will use in the following proof only very simple words in L_I, where the new command "4" occurs only once. More precisely, only the following words $Y_{X,p} \in L_I$ will be considered. Let X be a binary sequence. Assume that X is partitioned into k blocks $B_0, \ldots, B_{k-1}$ (listed in their order from left to right in X) of length p. We define $Y_{X,p}$ as the command sequence $X^\frown Z$ for M', where Z is the following test sequence. First, during one simultaneous sweep of both work heads of M' from right to left we arrange that, for $0 \leqslant i < k/2$, head 1 checks block B_i immediately after head 2 has checked block B_{2i} and immediately before head 2 checks block B_{2i+1} (a block is "checked" in the same way as described for language L in §1). In the second part of the test sequence Z both work heads of M' move from left to right and during this sweep head 1 checks, for $0 \leqslant i < k/2$, block $B_{k/2+i}$ immediately after head 2 has checked B_{2i+1} and immediately before head 2 checks B_{2i}.

Note that for $n := |X|$ the length of the word $Y_{X,p} := X^\frown Z \in L_I$ is bounded by $10n$. The only property of language L_I that will be used in the proof of Theorem 3.3 is the fact that $Y_{X,p} = X^\frown Z \in L_I$, whereas $X^\frown \tilde{Z} \notin L_I$ for every variation $\tilde{Z}$ of Z where some binary symbols in Z have been replaced by other binary symbols.

The movement pattern of the two work heads of M' during the test phase of the command sequence $Y_{X,p}$ is very simple: during both sweeps head 2 moves twice as fast as head 1. In this way every block B_b is checked by head 1 immediately after head 2 has checked block $B_{b^{(0)}}$ and immediately before head 2 checks block $B_{b^{(1)}}$; where

$$b^{(0)} \equiv 2b \pmod{k}, \; b^{(1)} \equiv b^{(0)} + 1 \quad \text{and} \quad b, b^{(0)}, b^{(1)} \in \{0, \ldots, k-1\}.$$

The following Theorem 3.1 extracts an important combinatorial property from this movement pattern, which will be used in the proof of Theorem 3.3. This combinatorial property appears to be also of independent interest (in a forthcoming paper [**19**] we analyze some related combinatorial structures–which we call "meanders" [**18**]–in a systematic fashion). It provides a finitary analogue to a well-known infinitary property of the "doubling transformation" $T(x) = 2x \pmod 1$, which maps the interval $[0,1)$ of real numbers into itself. This transformation is studied in ergodic theory (see e.g. Halmos [6]) as a standard example for a measure-preserving function (i.e. $\mu(T^{-1}[E]) = \mu(E)$ for every measurable set $E \subseteq [0,1)$) that is not invertible, but *ergodic*, which means that the only measurable sets E that are invariant under T (E is invariant under T if $T^{-1}[E] = E$) are trivial sets (i.e. $\mu(E) = 0$ or $\mu([0,1)\setminus E) = 0$). Theorem 3.1 asserts a strong ergodicity property of the finitary analogue of this transformation T: every set of measure $1/2$ is not only not invariant, it is moved by a *considerable amount.*

The following result gives also a partial answer to a question about expansion properties of graphs, which was raised by Klawe [12]. Klawe proved in [12] that every rational linear mapping T between one-dimensional graphs of k vertices yields at most an expansion factor of $1 + c/\log k$ for some constant c (i.e. $|T[X]| \leqslant (1 + c/\log k)|X|$ for some $X \subseteq \{0,\dots,k-1\}$). We show here that this upper bound on the expansion factor can actually be achieved by the (rational and linear) doubling transformation T for all sets $X \subseteq \{0,\dots,k-1\}$ of size $k/2$. Thus Klawe's upper bound on the expansion factor is optimal for such X (up to the constant c), which was conjectured by Klawe's (see the question at the end of [12]).

THEOREM 3.1. *Let S be a sequence of numbers from $\{0,\dots,k-1\}$ (possibly with repetitions), where $k = 2^l$ for some natural number l. Assume that every number $b \in \{0,\dots,k-1\}$ is somewhere in S adjacent to the number $b^{(0)} \equiv 2b \pmod k$ and somewhere in S adjacent to the number $b^{(1)} = b^{(0)} + 1$. Then for every partition of $\{0,\dots,k-1\}$ into two sets G and R of equal size $k/2$ there are at least $k/4\log k$ elements of G that occur somewhere in S adjacent to a number from R.*

REMARK 3.2. One can easily construct a sequence S with properties as in Theorem 3.1 whose length is bounded by $3k$ (see the construction of the command sequence $Y_{X,p}$).

PROOF OF THEOREM 3.1. Fix a partition of $\{0,\dots,k-1\}$ into two sets G and R such that $|G| = |R| = k/2$.

Let $e \in \{0,\dots,k-1\}$ be fixed. Then there is for every number $d \in \{0,\dots,k-1\}$ a unique sequence $\langle d_0^e,\dots,d_l^e\rangle$ of elements from $\{0,\dots,k-1\}$ such that $d_0^e = e$, $d_{r+1}^e = (d_r^e)^{(0)}$ or $d_{r+1}^e = (d_r^e)^{(1)}$, and $d_l^e = d$ (simply transform the binary sequence of length l which represents e in l many steps into the binary representation of d). We write G_r^e for the set of $b \in G$ such that for some d we have $d_r^e = b$ and $d_{r+1}^e \in R$, where $\langle d_0^e,\dots,d_l^e\rangle$ is the sequence associated with d. Note that by assumption on S every element of G_r^e is somewhere in S adjacent to an element from R.

CLAIM 1. Assume $e \in G$. Then $k/2 \leqslant \sum_{r=0}^{l-1}|G_r^e| \cdot 2^{l-r}$.

PROOF OF CLAIM 1. For every $d \in R$ there is in the associated sequence $\langle d_0^e,\dots,d_l^e\rangle$ a maximal index $\tilde{r} < l$ such that $d_{\tilde{r}}^e \in G$ and $d_{\tilde{r}+1}^e \in R$ (since $e = d_0^e \in G$ and $d = d_l^e \in R$). Note that $d_{\tilde{r}}^e \in G_{\tilde{r}}^e$ (by definition of $G_{\tilde{r}}^e$). Further, at most $2^{l-\tilde{r}}$ of the $k/2$ numbers $d \in R$ yield both the same index $\tilde{r}$ and the same number $d_{\tilde{r}}^e$ (by definition of the sequence $\langle d_0^e,\dots,d_l^e\rangle$ for d).

Claim 2. For every index $r \in \{0, \dots, l-1\}$ and every number $b \in \{0, \dots, k-1\}$ there are at most 2^r numbers e such that $b \in G_r^e$.

Proof of Claim 2. If $b \in G_r^e$ then we can write b in the form $b = \sum_{i=0}^{r-1} a_i \cdot 2^i + \sum_{j=0}^{l-r-1} e_j \cdot 2^{r+j}$, where $e = \sum_{j=0}^{l-1} e_j \cdot 2^j$ and $a_i, e_j \in \{0,1\}$. For every fixed b and r this relationship to e holds for at most 2^r many e (since the $l-r$ least significant bits of e are determined by b).

In order to finish the proof of Theorem 3.1 we add up the $k/2$ many inequalities for all $e \in G$ in Claim 1. Thus

$$k/2 \cdot k/2 \leqslant \sum_{e \in G} \sum_{r=0}^{l-1} |G_r^e| \cdot 2^{l-r} = \sum_{r=0}^{l-1} 2^{l-r} \Big(\sum_{e \in G} |G_r^e| \Big)$$
$$\leqslant \sum_{r=0}^{l-1} 2^{l-r} \cdot \Big| \bigcup_{e \in G} G_r^e \Big| \cdot 2^r,$$

where we used Claim 2 for the last inequality. Further,

$$\sum_{r=0}^{l-1} 2^{l-r} \cdot \Big| \bigcup_{e \in G} G_r^e \Big| \cdot 2^r = 2^l \cdot \sum_{r=0}^{l-1} \Big| \bigcup_{e \in G} G_r^e \Big| = k \cdot \sum_{r=0}^{l-1} \Big| \bigcup_{e \in G} G_r^e \Big|.$$

Together this implies (since $l = \log k$) that $|\bigcup_{e \in G} G_r^e| \geqslant k/4 \log k$ for some $r \in \{0, \dots, l-1\}$. The claim of Theorem 3.1 follows from this inequality by the definition of the sets G_r^e.

The next result yields the desired (nearly) quadratic lower bound for *nondeterministic* 1-tape Turing machines (always with an additional one-way input tape).

Theorem 3.3. *The language L_I of iterated polydromes is accepted in linear* (*even real*) *time by a deterministic* 2*-tape Turing machine, but L_I is not accepted in time $o(n^2/\log^2 n \log\log n)$ by any nondeterministic* 1*-tape Turing machine.*

Proof. Assume for a contradiction that M is a nondeterministic 1-tape TM that accepts L_I in time $t(n) = o(n^2/\log^2 n \log\log n)$. Compared with the proof of Theorem 2.1 the following new difficulty arises because M is nondeterministic. On any input $X^\frown Z$ such a machine M can guess during the first part of its computation (while its input head reads X) what the second part Z of the input will be. In particular, M might process X in such a way that just this specific "test sequence" Z can be executed with relatively few steps (say $n^{1.5}$ many steps). This additional ability appears to be fatal to our argument from §2. There we had relied on the possibility to wait with the definition of Z until that point where M had already processed X (we chose Z so that it exploited a particular "weak point" of M's processing of X).

According to the previous considerations we must now specify immediately the *complete* input $X^\frown Z$ on which we challenge M. We set $X^\frown Z = Y_{X,p}$ (defined at the beginning of this section) for some "random" string X and a suitable number p. We show that Theorem 3.1 implies that this test sequence Z has the following property: For *any* two sufficiently large disjoint sets L and R of blocks of length p from X there is a reasonably large number of $L-R$ pairs in Z. We need this strong property of Z because when we define the input $X^\frown Z$, we do not yet know *which* sets L_D and R_D will arise in the "Desert Lemma" (Lemma 3.4).

For the precise proof we fix a natural number n that is sufficiently large (exact conditions will come out of the following arguments). For convenience we choose n so that all later occurring terms such as $\log\log n$, $n/8\log\log n$, etc., have natural numbers as values. Further, we fix (as in §2) a binary string X of length n with $K(X) \geqslant n$. We partition X into $k := n/8\log\log n$ blocks of length $8\log\log n$. We choose Z so that $X^\frown Z = Y_{X,p}$ for $p := 8\log\log n$ (see the definition at the beginning of this section). This definition implies that $X^\frown Z \in L_I$. We fix an accepting computation C of the nondeterministic TM M on input $X^\frown Z$. By assumption on M, the length of C is bounded by $t(10n)$, where $t(n) = o(n^2/\log^2 n \log\log n)$.

LEMMA 3.4. ("DESERT LEMMA"). *If n is large enough then there is an interval D of $2^{-12}n/\log n$ cells on the work tape of M and there are two sets L_D and R_D which contain each $k/2 - 2^{-10}n/\log n \log\log n$ blocks from X such that in computation C the work head of M is always left (right) of D during those steps where the input head reads from a block in X that belongs to L_D (R_D).*

PROOF OF LEMMA 3.4. The proof is analogous to the proof of Lemma 2.2. For large n there are less than $2^{-11}n/\log n \log\log n$ blocks B in X such that in computation C the work of M moves over $2^{-12}n/\log n$ or more cells while its input head reads B (since $t(n) = o(n^2/\log^2 n \log\log n)$). Therefore it will be sufficient to find an interval D_0 of $3 \cdot 2^{-12}n/\log n$ cells on the work tape of M which has the following two properties (define D as the middle third of D_0):

(i) there are at least $k/2 - 2^{-11}n/\log n \log\log n$ blocks B in X such that at *some* step of computation C the work head of M is left of D_0 while the input head reads from B,

(ii) there are at least $k/2 - 2^{-11}n/\log n \log\log n$ blocks B in X such that at *some* step of computation C the work head of M is right of D_0 while the input head reads from B.

Define D_0 as the leftmost interval of length $3 \cdot 2^{-12}n/\log n$ which satisfies property (i). Assume for a contradiction that this interval does not have property (i). Define another interval D_1 of length $1 + 3 \cdot 2^{-12}n/\log n$ that consists of D_0 together with the next cell to the left of D_0. By definition of D_1 there is a set T of $2^{-10}n/\log n \log\log n$ blocks B such that in computation C the work head of M is always inside D_1 while the input head reads from B. Let c_L (c_R) be a cell that lies between 1 and $3 \cdot 2^{-12}n/\log n$ cells to the left (right) of D_1 such that the work head of M scans c_L (c_R) during at most $2^{-11}n/\log n \log\log n$ steps of C. Let S_L (S_R) be the crossing sequence for cell c_L (c_R) which records for every crossing of this cell the current machine state and (in binary code) the number of cells by which the input head has advanced since the last crossing. The differences d_i of the input head positions that occur in S_L (S_R) add up to n (only that part of computation C is relevant where X is read). Thus

$$\sum_{i=1}^{2^{-11}n/\log n \log\log n} (1 + \log d_i) \leqslant \left(2^{-11}n/\log n \log\log n\right) \cdot \left(1 + \log(2^{11}\log n \log\log n)\right)$$

and S_L and S_R together require no more than $2^{-9}n/\log n$ bits.

In the same way as in the proof of Lemma 2.2 one defines an auxiliary TM P in order to show that

$$(10) \qquad K(T^{\frown}|X^C) \leqslant 2^{-8}n/\log n$$

(where $T^{\frown}$ is again the concatenation of all blocks in T, X^C is a variation of X where all blocks that belong to T have been censored). The only difference to Lemma 2.2 is the fact that here the TM P requires more computation time: it has to simulate for every considered input $X^C[U]$ all possible computations of the nondeterministic TM M on this input. However, the result of the argument is not affected by this difference.

In order to achieve the desired contradiction to (10) one shows in analogy to (3) in Lemma 2.2 that

$$\begin{aligned}(11) \qquad K(X^C) &\leqslant n - (2^{-10}n/\log n \log\log n) \cdot 8 \cdot \log\log n \\ &\quad + (2^{-10}n/\log n \log\log n) \cdot (1 + \log(2^{10} \log n \log\log n)) \\ &\leqslant n - 6 \cdot 2^{-10}n/\log n.\end{aligned}$$

It is obvious that $n \leqslant K(X) \leqslant K(X^C) + K(T^{\frown}|X^C)$ and therefore we get, from (11), that

$$K(T^{\frown}|X^C) \geqslant n - K(X^C) \geqslant 6 \cdot 2^{-10}n/\log n.$$

This contradiction to (10) completes the proof of Lemma 3.4.

We now consider the sequence S of block indices $\in \{0, \dots, k-1\}$ in the order as they are checked in the considered test sequence Z (for which $X^{\frown}Z = Y_{X,p}$ is the input for M). We noted earlier that S satisfies the hypothesis of Theorem 3.1. Let $\tilde{L}_D$ and $\tilde{R}_D$ be the sets of indices of those blocks that belong to the sets L_D, respectively R_D, in Lemma 3.4. We fix any two disjoint sets L, $R \subseteq \{0, \dots, k-1\}$ with $|L| = |R| = k/2$, $\tilde{L}_D \subseteq L$ and $\tilde{R}_D \subseteq R$. According to Theorem 3.1 there is a set H of at least $k/4 \log k$ elements of L that are somewhere in S adjacent to an element of R. We have $|L - \tilde{L}_D|$, $|R - \tilde{R}_D| \leqslant 2^{-9}n/\log n \log\log n$. Therefore, at most $2^{-9}n/\log n \log\log n$ elements of H belong to $L - \tilde{L}_D$ and at most $6 \cdot 2^{-9}n/\log n \log\log n$ elements of H are adjacent to an element of $R - \tilde{R}_D$ (every element of $\{0, \dots, k-1\}$ occurs three times in S). Thus there is a set $H' \subseteq H$ of at least

$$k/4 \log k - 7 \cdot 2^{-9}n/\log n \log\log n \geqslant 2^{-6}n/\log \mathrm{n} \log\log \mathrm{n}$$

elements of $\tilde{L}_D$ that are somewhere in S adjacent to an element of $\tilde{R}_D$. Therefore, there is a set $H'' \subseteq H'$ of $2^{-6}n/6 \log n \log\log n$ elements of $\tilde{L}_D$ that are all adjacent to *different* elements of $\tilde{R}_D$. Each of these pairs of adjacent numbers in S corresponds to an $L_D - R_D$ pair in Z (i.e. a pair of blocks from L_D, respectively R_D, that are checked in immediate succession in Z). In this way we get a set $\tilde{H}$ of $2^{-6}n/6 \log n \log\log n$ $L_D - R_D$ pairs in Z which all contain different blocks from L_D and different blocks from R_D. We write D_l, D_m, D_r for the left middle, respectively right third, of the "desert" D from Lemma 3.4.

LEMMA 3.5. *For at least $2/3$ of the $2^{-6}n/6\log n\log\log n$ $L_D - R_D$ pairs in $\tilde{H}$ the work head of M touches a cell left of D_m during those steps in computation C where M's input head reads from that pair.*

PROOF OF LEMMA 3.5. Assume for a contradiction that for $1/3$ of the $L_D - R_D$ pairs in $\tilde{H}$ the work head of M stays to the right of D_l during those steps where the input head reads from that pair in Z. Let $\tilde{L}$ be a set of $2^{-11}n/\log n\log\log n$ blocks from L_D that occur in these $L_D - R_D$ pairs. Let c_l be a cell in D_l that is scanned during at most $2^{-12}n/\log n\log\log n$ steps of computation C. Let S_l be the crossing sequence for cell c_l in computation C (analogously as in Lemma 2.3). In the same way as in Lemma 3.4 one can show that S_l requires at most $2^{-1}n/\log n$ bits.

We write $X^C{}^\cap Z^C$ for the variation of input $X^\cap Z$ where all blocks that belong to $\tilde{L}$ have been "censored" (see §2 for a definition). Let $\tilde{L}^\cap$ be the concatenation of all blocks in $\tilde{L}$ (in the order as they appear in X). We construct an auxiliary TM $\tilde{P}$ that computes $\tilde{L}^\cap$ from $X^C{}^\cap Z^C$, S_l, the number of cell c_l and the final machine state of computation C. P tries successively (in the same way as TM $\tilde{P}$ in the proof of Lemma 2.3) all possible fill-ins for the censored blocks in $X^C{}^\cap Z^C$; in addition it tries all possible computations of the nondeterministic TM M on the resulting inputs. The same "cut and paste" argument as in Lemma 2.3 shows that $\tilde{L}^\cap$ is the only fill-in that allows a computation of M on the resulting input which generates S_l on cell c_l and halts in the same final state as computation C. The existence of TM $\tilde{P}$ implies that $K(\tilde{L}^\cap | X^C{}^\cap Z^C) \leqslant 2^{-10}n/\log n$. On the other hand,

$$K(X^C) \leqslant n - (2^{-11}n/\log n\log\log n)\cdot(8\log\log n)$$
$$+ (2^{-11}n/\log n\log\log n)\cdot(2\log\log n).$$

Therefore (because $K(X) \geqslant n$)

$$K(\tilde{L}^\cap | X^C{}^\cap Z^C) \geqslant n - K(X^C) - K(X^C{}^\cap Z^C | X^C) \geqslant 6\cdot 2^{-11}n/\log n$$

(note that $K(X^C{}^\cap Z^C | X^C) = O(1)$). This contradicts the previously derived upper bound and the proof of Lemma 3.5 is complete.

A symmetrical version of Lemma 3.5 shows that for at least $2/3$ of the $L_D - R_D$ pairs in $\tilde{H}$ the work head of M touches a cell to the *right* of D_m while the input head reads from that pair. Together with Lemma 3.5 this implies that for at least $1/3$ of the $2^{-6}n/6\log n\log\log n$ $L_D - R_D$ pairs in $\tilde{H}$ the work head of M crosses the middle third D_m of the "desert" D while the input head reads from that pair. We have $|D_m| = 2^{-12}n/3\log n$ and therefore M uses in computation C at least

$$(2^{-6}n/6\log n\log\log n)\cdot(2^{-12}n/3\log n) = 2^{-18}\cdot n^2/18\log^2 n\log\log n$$

steps. This contradicts the assumed time bound of $t(n) = o(n^2/\log^2 n\log\log n)$ steps for TM M and the proof of Theorem 3.3 is complete.

4. Separation results for determinism, nondeterminism and co-nondeterminism. So far we have compared the computing power of Turing machines that have the same control structure but different memory structures. We now compare Turing machines that all have the same type of memory: one work tape (in addition to the one-way input tape), but different control structures: deterministic, nondeterministic resp. co-nondeterministic. The corresponding complexity classes $\mathrm{DTIME}_1(t(n))$, $\mathrm{NTIME}_1(t(n))$ and $\text{CO-NTIME}_1(t(n))$ were defined in §1.

THEOREM 4.1. *Assume* $t(n) = o(n^2)$. *Then* $\mathrm{NTIME}_1(n) \not\subseteq \mathrm{DTIME}_1(t(n))$.

PROOF. The complement of the language L is obviously in $\mathrm{NTIME}_1(n)$ (the machine can guess the "reason" why a word is not in L and verify this guess with its single work head). According to Theorem 2.1 L is not in $\mathrm{DTIME}_1(t(n))$.

THEOREM 4.2. *Assume* $t(n) = o(n^2/\log^2 n \log\log n)$. *Then* $\text{CO-NTIME}_1(n) \not\subseteq \mathrm{NTIME}_1(t(n))$.

PROOF. The complement of language L_I is in $\mathrm{NTIME}_1(n)$ (use an analogous real time algorithm as for L). According to Theorem 3.3 L_I is not in $\mathrm{NTIME}_1(t(n))$.

ACKNOWLEDGMENTS. We would like to thank Andreas Blass, Joel Seiferas and Zvi Galil for helpful suggestions. In particular, Zvi Galil [5] pointed out a simple way of saving a log-factor in our estimates.

REFERENCES

1. R. V. Book, S. A. Greibach and B. Wegbreit, *Time and tape bounded Turing acceptors and AFL's*, J. Comput. System Sci. **4** (1970), 606–621.
2. P. Duris and Z. Galil, *Two tapes are better than one for nondeterministic machines*, Proc. 14th ACM STOC (1982), 1–7.
3. P. Duris, Z. Galil, W. J. Paul and R. Reischuk, *Two nonlinear lower bounds*, Proc. 15th ACM STOC, 1983, pp. 127–132.
4. O. Gabber and Z. Galil, *Explicit constructions of linear size superconcentrators*, Proc. 20th IEEE FOCS, 1979, pp. 364–370.
5. Z. Galil, private communication
6. P. R. Halmos, *Ergodic theory*, Lecture Notes, University of Chicago, 1955.
7. J. Hartmanis and R. E. Stearns, *On the computational and complexity of algorithms*, Trans. Amer. Math. Soc. **117** (1965), 285–306.
8. F. C. Hennie, *One-tape off-line Turing machine computations*, Inform. and Control **8** (1965), 553–578.
9. F. C. Hennie and R. E. Stearns, *Two-tape simulation of multitape Turing machines*, J. Assoc. Comput. Mach. **13** (1966), 533–546.
10. J. E. Hopcroft and J. D. Ullman, *Introduction to automata theory, languages and computation*, Addison-Wesley, Reading, Mass., 1979.
11. R. Kannan, *Alternation and the power of nondeterminism*, Proc. 15th ACM STOC, 1983, pp. 344–346.
12. M. Klawe, *Non-existence of one-dimensional expanding graphs*, Proc. 22th IEEE FOCS, 1981, pp. 109–113.
13. M. Li, *On one tape versus two stacks*, preprint (February 1984).
14. W. Maass, *Simulation of two tapes by one tape requires quadratic time*, abstract (August 1983).
15. ______, *Quadratic lower bounds for deterministic and nondeterministic Turing machines*, abstract (September 1983).
16. ______, *Are recursion theoretic arguments useful in complexity theory* (Proc. Internat. Conf. on Logic, Methodology and Philosophy of Science, Salzburg, 1983), North-Holland, Amsterdam, 1983.
17. ______, *Quadratic lower bounds for deterministic and nondeterministic one-tape Turing machines*, Proc. 16th ACM STOC, 1984, pp. 401–408.
18. ______, *An optimal lower bound for random access machines and other applications of Ramsey's theorem*, abstract (June 1984).
19. ______, *Meanders, Ramsey's theorem and lower bound arguments* (in preparation).

20. W. Maass and A. Schorr, *Speed-up of one-tape Turing machines by bounded alternation* (in preparation).

21. W. J. Paul. *On-line simulation of $k + 1$ tapes by k tapes requires nonlinear time*, Proc. 23rd IEEE FOCS, 1982, pp. 53–56.

22. W. J. Paul, N. Pippenger, E. Szemeredi and W. Trotter, *On determinism versus nondeterminism and related problems*, Proc. 24th IEEE FOCS, 1983, pp. 429–438.

23. M. O. Rabin, *Real time computation*, Israel J. Math. **1** (1963), 203–211.

24. P. M. B. Vitânyi, *One queue or two pushdown stores take square time on a one-head tape unit*, Report CS-R8406 of the Centre for Mathematics and Computer Science, Amsterdam, 1984.

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE DIVISION, UNIVERSITY OF CALIFORNIA, BERKELEY, BERKELEY, CALIFORNIA 94720

Current address: Department of Mathematics, Statistics and Computer Science, University of Illinois at Chicago, Chicago, Illinois 60680