QUADRATIC LOWER BOUNDS FOR DETERMINISTIC AND NONDETERMINISTIC ONE-TAPE TURING MACHINES

(Extended Abstract)

Wolfgang Maass *'**

Dept. of Mathematics and Computer Science Division

University of California, Berkeley

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 produce quadratic lower bounds for the simulation of 2-tape TM's by 1-tape TM's and thus answer a rather old question (problem No.1 and No.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 TM's.

Hartmanis and Stearns [5] have shown that one can simulate a multi-tape TM (Turing machine) that runs in time $t(n)$ by a 1-tape TM that runs in time $O(t^2(n))$. This holds if both machines are deterministic and if both are nondeterministic. However the question whether multi-tape TM's are really substantially more powerful than 1-tape TM's has been left open (except for the case of TM's without input tape; see below). In spite of quite a bit of work only nearly linear lower bounds have been achieved. Rabin [11] showed that there is some difference: not every det. 2-tape TM that runs in real time can be simulated by a det. 1-tape TM that runs in real time (a TM M runs in real time if there is a constant c_M s.t. M spends at most c_M steps before it reads the next input bit resp. halts, thus real time implies linear time). Wolfgang Paul introduced the notion of Kolmogorov complexity into computational complexity theory and used it to show that simulation of a real time det. 2-tape TM by an "on-line" det. 1-tape TM requires time $\Omega(n \cdot \log^{1/2} n)$ [9]. His notion of "on-line" machine helps to keep the simulator on a "shorter leash". On the other hand it prevents the translation of the lower bound result into the familiar terms of language acceptance (it is only significant for machines with large output).

For nondeterministic TM's all questions about the influence of the number of tapes on the computing power have been settled except for 2 tapes

* Written under support by the Heisenberg Programm of the Deutsche Forschungsgemeinschaft, Bonn

** Address after fall 84 : Dept. of Mathematics, Statistics and Computer Science, University of Illinois at Chicago, Chicago, IL 60680

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1984 ACM 0-89791-133-4/84/004/0401 $00.75

versus 1 (according to Book, Greibach, Wegbreit [1] one can simulate any nondet. TM without time loss by a nondet. 2-tape TM). Concerning 2 tapes versus 1 Duris and Galil [2] proved that a real time det. 2-tape TM cannot be simulated by a nondet. 1-tape TM in real time. In Duris, Galil, Paul, Reischuk [3] the lower bound was improved to $\Omega(n \cdot \log \log n)$ (apparently they have recently achieved $\Omega(n \cdot \log n)$).

The main disadvantage of a 1-tape TM is the fact that it needs $\Omega(s \cdot d)$ steps to move a string of s symbols on its work tape over d cells, while a 2-tape TM can do this in time $O(s+d)$. This observation allows to derive easily quadratic lower bounds 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), see Hennie [6]. Such machine requires quadratic time to check whether a string is a palindrome. The 1-tape TM with an extra one-way input tape --this is the model that is considered in the previously mentioned lower bound literature and which we will consider in this paper-- is quite a bit more powerful and can e.g. recognize palindromes in linear time. In addition such 1-tape TM has the option to choose a clever data structure for the representation of the input on its work tape (this may make it unnecessary to perform during the computation many time consuming copying operations). A clever data representation is for example used to simulate a det. k-tape TM (for any $k > 2$) with time bound $t(n)$ by a det. 2-tape TM without a severe time loss in time $O(t(n) \cdot \log t(n))$ (Hennie and Stearns [7]). In view of these facts 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 nondet. 1-tape TM's are already quite powerful. They accept already in linear time NP-complete problems like 3-COLORABILITY. From the technical point of view such machines have the additional option to use "individualized" data structures for the representation of the input on their work tape that are optimal for a particular computation on a particular input. It turns out that nevertheless the Hartmanis/Stearns simulation [5] is optimal in the deterministic case and nearly optimal in the nondeterministic case.

Theorem 1 There is a language that is accepted by a deterministic 2-tape TM in linear (even real) time but which requires time $\Omega(n^2)$ on any deterministic 1-tape TM.

Theorem 2 There is a language that is accepted by a deterministic 2-tape TM in linear (even real) time but which is not accepted in time $O(n^2/\log^5 n)$ by any nondeterministic 1-tape TM.

Remark: In a preliminary version of this paper we proved somewhat weaker lower bounds (we showed that a 1-tape TM cannot accept the considered language in time $O(n^{2-\delta})$, for any $\delta > 0$). We are grateful to Zvi Galil and Joel Seiferas who pointed out that with a more careful choice of parameters our argument yields stronger lower bounds. Further we use Galil's suggestion [4] to save a log -factor by recording sequences of numbers less than n in a more concise way (by their binary differences, one uses the convexity of the log function for the estimate of the total length).

We now compare machines that have the same storage facility (one work tape) but different control structures (det., nondet., co-nondet.). We write $DTIME_k(t(n))$ and $NTIME_k(t(n))$ for the classes of sets that are accepted in time $O(t(n))$ by det. resp. nondet. k-tape TM's (always with an additional one-way input tape). Known lower bound results are $NTIME_2(n) \not\subseteq \bigcup_{k \geq 1} DTIME_k((n \cdot \log^* n)^{1/4})$ (Paul, Pippenger, Szemeredi, Trotter [10]), $NTIME_2(n) \not\subseteq DTIME_1(n^{1.1})$ (Kannan [7]) and $NTIME_2(n) \not\subseteq DTIME_1(n^{1.22})$ (Maass and Schorr [8]; here the lower bound also holds for 1-tape TM's with a two-way input tape).

Theorem 3 Assume $f(n) = o(n^2)$. Then $NTIME_1(n) \not\subseteq DTIME_1(f(n))$.

Theorem 4 $CO\text{-}NTIME_1(n) \not\subseteq NTIME_1(n^2/\log^5 n)$.

To prove the preceding theorems we construct a language L_I for which the following three lemmata hold. Theorem 2 and Theorem 4 follow directly from these lemmata. The somewhat simpler proofs of Theorem 1 and Theorem 3 are sketched at the end of this paper.

Lemma 1 L_I is accepted by a deterministic 2-tape TM in real time.

Lemma 2 The complement of L_I is accepted by a nondet. 1-tape TM in real time.

Lemma 3 (Main Lemma) Assume $f(n) = o(n^2/\log^4 n \cdot \log\log n)$. Then $L_I \notin NTIME_1(f(n))$.

The language L_I consists of finite sequences of symbols 0,1,2,3,4 and is defined as follows:

$L_I = \{x_1 \ldots x_n z_1 \ldots z_k \mid x_1,\ldots,x_n \in \{0,1\}, z_1 = z_k = 4, \forall a \forall b((1 \leq a < b \leq k \wedge z_a = z_b = 4 \wedge \forall j(a < j < b \rightarrow z_j \neq 4)) \rightarrow (|\{j \mid a < j < b \wedge z_j = 2\}| = |\{j \mid a < j < b \wedge z_j = 3\}| = n \wedge \forall j((a < j < b \wedge z_j \in \{0,1\}) \rightarrow (z_{j-1} \in \{2,3\}$ and for $m := |\{j' \mid a < j' \leq j-1 \wedge z_{j'} = z_{j-1}\}|$ one has $z_j = x_{n+1-m}$ if the number of 4's in $z_1..z_a$ is odd and $z_j = x_m$ otherwise)))\}$.

In order to understand the structure of L_I it is helpful to go through the proof of Lemma 1, i.e. to construct a det. 2-tape TM M' that recognizes words from L_I in real time. M' interprets each symbol 0,1,2,3,4 as a command. Machine M' starts in "writing mode" and copies the initial segment of its input Y from left to right on both of its work tapes, until it encounters in Y a symbol $z \notin \{0,1\}$. M' rejects Y unless $z = 4$. M'then changes (forever) into its "testing mode". M' always interprets 4 as the command to change the direction of movement for both of its work heads (after moving both heads one cell further in the old direction). M' interprets 2 (3) as the command to move work head 1 (2) by one cell in the currently required direction. M' in testing mode interprets a symbol $y \in \{0,1\}$ in Y as the command to test whether the work head that moved at the previous step reads the symbol y in his cell. The input Y is in L_I if all these tests have a positive outcome, if the movements of the two work heads of M' in testing mode consist of full sweeps (in parallel) over the non-blank part of the work tapes and if the input ends with a 4.

As an example for words in L_I we note that a binary string $x_1 \ldots x_n y_1 \ldots y_n$ is a palindrome iff the string $x_1 \ldots x_n 42 y_1 \ldots 2 y_n 4$ is in L_I.

For the lower bound argument we will consider words in L_I of the following structure. Let $X = x_1 \ldots x_n$ be a binary string and let $L = \{l_1, l_2, ..\}$ and $R = \{r_1, r_2, ..\}$ 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 L and R are listed in the order of their appearance in X from right to left. Let $l_{i,1} \ldots l_{i,p}$ and $r_{i,1} \ldots r_{i,p}$ be the symbols of block l_i resp. r_i in the order from right to left. Let $d_L(i)$ ($d_R(i)$) be the number of bits between blocks l_i and l_{i+1} (r_i and r_{i+1}) in X. Further let $d_L(0)$ ($d_R(0)$) be the number of bits in X to the right of block l_1 (r_1). Then the following string $Y_{X,L,R}$ is in L_I:

$$Y_{X,L,R} = x_1 \ldots x_n \underbrace{4 2 \ldots 2}_{d_L(0) \text{ times}} 2 l_{1,1} 2 l_{1,2} \ldots 2 l_{1,p} \underbrace{3 \ldots 3}_{d_R(0) \text{ times}}$$

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

$3 r_{2,1} 3 r_{2,2} \ldots 3 r_{2,p} \ldots$ etc. (alternating through all blocks of L and R)4 .

The proof of Lemma 2 is easy (construct a nondet. 1-tape TM that accepts the complement of L_I : it guesses the "reason" why a string is not in L_I and verifies this guess with its single work head).

We now sketch the proof of Lemma 3 (Lemmata 4 to 6 are needed for this proof). Assume for a contradiction that there is a nondet. 1-tape TM M that accepts L_I in time $f(n)$ where $f(n) = o(n^2/(\log^4 n \cdot \log \log n))$.

We fix some canonical way of coding TM's $\tilde{M}$ by binary strings. $|\tilde{M}|$ is the length of the binary string that codes $\tilde{M}$. We will consider the Kolmogorov complexity $K(X|Y)$ of a string X relativ to a string Y (as in Paul [9]), where $K(X|Y) := \min\{|\tilde{M}| \mid \text{TM } \tilde{M} \text{ produces on input } Y \text{ the output } X\}$ and $K(X) := K(X|\varepsilon)$.

We now construct the input $X \hat{} Z$ on which we test TM M . We choose for X some binary string with $K(X) \geq |X| =: n$ (such X exist for every n ,as a simple counting argument shows). We choose n s.t. $n >> |M|$ (thus X looks like a random string to M). Precise conditions on the size of n will come out of the following arguments.

In order to define Z we partition X into $n' := n/8 \cdot \log \log n$ blocks of length $8 \cdot \log \log n$. We number these blocks in X from left to right by binary sequences of length $\log n'$ in lexicographical order (we assume that n is chosen so that $\log n'$ is a natural number). For $i \in \{1, .., \log n'\}$ we say that two blocks are <u>i-connected</u> if their associated binary sequences differ exactly at the i-last bit. This induces a partition of the blocks of X into sets L_i and R_i : if blocks b_1, b_2 are i-connected and b_1 is left of b_2 in X then we place b_1 in L_i and b_2 in R_i . The second part Z of the input has the form $Z = 4 \hat{} Z_1 \hat{} 4 \hat{} Z_2 \hat{} \ldots 4 \hat{} Z_{\log n'} \hat{} 4$. Z_1 is defined such that $X \hat{} 4 \hat{} Z_1 \hat{} 4 = Y_{X,L_1,R_1}$ (see the definition of $Y_{X,L,R}$ in the preceding example). Z_2 is an analogous command sequence that tells the previously considered 2-tape TM M' to check all blocks in L_2 and R_2 during one sweep in parallel of both work heads from left to right (head 1 checks the blocks in L_2 and head 2 checks the blocks in R_2 ; this is done in alternation so that both heads look in general at the same time at quite different parts of X). Z_3 is defined so that $X \hat{} 4 \hat{} Z_3 \hat{} 4 = Y_{X,L_3,R_3}$ (i.e. Z_3 tells M' to check the blocks in L_3 and R_3 in alternation

during one sweep of both work heads from right to left). Z_4 describes like Z_2 a sweep of both heads from left to right, where the blocks from L_4 and R_4 are checked in alternation. Etc.

It is obvious from the construction that $X\hat{}Z \in L_I$ and that $|X\hat{}Z| \leq 10\ n \cdot \log n$. Thus the nondet. 1-tape TM M accepts $X\hat{}Z$ via some computation C with g(n) steps, where $g(n) = o(n^2/(\log^2 n \cdot \log \log n))$.

Lemma 4 ("Desert Lemma") For large enough n there is an interval D ("desert") of $\tilde{n} := n/(100 \cdot \log n)$ cells on the work tape of M and there are two sets L_D and R_D, each consisting of $n'/2 - 2n/(100 \cdot \log n \cdot \log \log n)$ blocks B of X, s.t. in computation C the work head of M is always left (right) of D during those steps where its input head reads from a block B in X that belongs to L_D (R_D).

Sketch of the proof of Lemma 4: We write $\hat{n}$ for $n/(\log n \cdot \log \log n)$. For large n there are less than $\hat{n}/100$ blocks B in X s.t. in computation C the work head of M moves over $\tilde{n}$ or more cells while its input head reads B. Therefore it is sufficient to find an interval D_0 of $3\tilde{n}$ cells on the work tape of M (D will be its middle third) such that

i) there are $\geq n'/2 - \hat{n}/100$ blocks B in X s.t. at some step of computation C the work head of M is left of D_0 and simultaneously the input head is in B and

ii) there are $\geq n'/2 - \hat{n}/100$ blocks B in X s.t. at some step of computation C the work head of M is right of D_0 and simultaneously the input head is in B.

We define D_0 as the leftmost interval on the work tape of M that has property i). Assume for a contradiction that this interval D_0 does not satisfy 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 there is a set T of $2\hat{n}/100$ blocks B in X s.t. in computation C the work head of M is always inside interval D_1 at every step where its input head reads from a block B in X that belongs to T. Let c_L (c_R) be a cell to the left (right) of D_1 within distance $\tilde{n}$ from D_1 s.t. the work head of M scans this cell during at most $\hat{n}/100$ steps of C. Let S_L (S_R) be the crossing sequence for cell c_L (c_R) which records for every step of C where the work head scans this cell the current machine state and the number of cells by which the input head has advanced since the last crossing of this cell (in binary). Both crossing sequences together require only $4n/(100 \cdot \log n)$ bits (according to Galil [4] the differences d_i of the input head positions that occur in S_L (S_R) add up to n, thus

$$\sum_{i=1}^{\hat{n}/100} (1 + \log d_i) \leq (\hat{n}/100) \cdot (1 + \log(100n/\hat{n}))$$

by the convexity of the log function).

We define X^C to be a variation of X where all blocks that belong to T have been "censored" (i.e. substituted by equally long sequences of a new symbol ■). Let I be the inscription of the interval between cells c_L and c_R at that step t_0 of computation C where the input head moves off the last symbol of X. Let $T\hat{}$ be the concatenation of the blocks in T (in their natural order from left to right). One can define an auxiliary TM P that produces the output $T\hat{}$ from an input that consists of X^C, I, the initial parts

of S_L and S_R until step t_0, the cell numbers of c_L and c_R and the state and head positions of M at step t_0 of C. Roughly P tries all possible substitutes for the censored parts of input X^C until it finds one for which M has a computation that generates the same data as those which are recorded in the input of P. To verify the correctness of P's output we use a cut and paste argument (here it is essential that S_L, S_R also record the current positions of the input head). The existence of P implies that $K(T^\wedge | X^C) \leq 10n/(100 \cdot \log n)$. On the other hand our choice of X implies that $K(T^\wedge | X^C) \geq 12n/(100 \cdot \log n)$. This contradiction finishs the proof of Lemma 4.

The second part Z of the input has the important property that any two blocks b_1, b_2 of X that are i-connected for some $i \leq \log n'$ are checked in immediate succession in Z (i.e. in terms of the previously described 2-tape TM M' work head 1 of M' checks all bits of one of the two blocks and immediately afterwards work head 2 of M' moves to the place where it has recorded the other block and checks all its bits). We call a subsequence of successive symbols of Z an $L_D - R_D$ pair if it consists of the commands to check in immediate succession two blocks where one belongs to L_D and the other to R_D (L_D and R_D are the sets from Lemma 4). We need the following purely combinatorial Lemma.

Lemma 5 Assume that the n' blocks of X have been partitioned into three sets L, R, G. Then there are at least $\min\{|L|, |R|\} - |G| \cdot \log n'$ pairs of blocks s.t. one belongs to L, the other to R and both are i-connected for some $i \leq \log n'$.

Proof: Induction on $\log n'$. Set $l := \log n'$. For $i = 0,1$ let H_i be the subset of sequences of length $l+1$ that begin with i. The given partition of the binary sequences of length $l+1$ into L, R, G induces partitions of H_i into L_i, R_i, G_i (for $i = 0,1$).

Case 1: $\min(|L_i|, |R_i|) = |L_i|$ for $i = 0$ and $i = 1$.

Then $\min(|L|, |R|) = \min(|L_0|, |R_0|) + \min(|L_1|, |R_1|)$ and the claim follows immediately from the induction hypothesis.

Case 2: $\min(|L_0|, |R_0|) = |L_0|$ and $\min(|L_1|, |R_1|) = |R_1|$.

Since $R_0 = 2^l - |L_0| - |G_0|$ and at most $|R_1| + |G_1|$ elements of H_1 are not in L_1, at least $2^l - |L_0| - |G_0| - |R_1| - |G_1|$ elements of R_0 are $(l+1)$-connected with elements from L_1. Further by induction hypothesis at least $|L_0| - |G_0| \cdot l$ L - R pairs inside H_0 and at least $|R_1| - |G_1| \cdot l$ L - R pairs inside H_1 are i-connected for some $i \leq l$. Thus altogether at least $2^l - |G| \cdot (l+1) \geq \min(|L|, |R|) - |G| \cdot (l+1)$ L - R pairs are i-connected for some $i \leq l+1$.

In our application of Lemma 5 we set $L := L_D$, $R := R_D$ and $G :=$ {the remaining blocks of X}. We derive in this way that there are at least $n'/2 - 2\hat{n}/100 - 4\hat{n}/(100 \cdot \log n) \geq n/(100 \cdot \log \log n)$ $L_D - R_D$ pairs in Z (for large n).

Machine M cannot move within its time bound too many blocks to different locations once they are written down somewhere on the work tape. Therefore it is intuitively plausible that for most of the $L_D - R_D$ pairs in Z, where some blocks b_1

and b_2 are checked with $b_1 \in L_D$ and $b_2 \in R_D$, the work head of M has to go close to the area where it had originally written down these blocks, i.e. close to the left end of desert D for checking b_1 and close to the right end of D for checking b_2 (otherwise one could "fool" M and make it accept strings that are not in L_I). The following Lemma verifies that this intuition is correct. We write D_l, D_m, D_r for the left, middle resp. right third of the desert D from Lemma 4.

Lemma 6 For at least 2/3 of the $n/(100 \cdot \log \log n)$ L_D - R_D pairs in Z the work head of M touches a cell left of D_m (right of D_m) during those steps in computation C where M's input head reads from that pair.

Proof (sketch): Assume for a contradiction that for 1/3 of the L_D - R_D pairs in Z the work head of M stays during those steps of computation C where M's input head reads from that pair always to the right of D_l. Let $\tilde{L}$ be a set of $\hat{n}/300$ different blocks from L_D that occur in these pairs (note that the same block may occur in up to $\log n'$ L_D - R_D pairs). Let c be a cell in D_l that is scanned during at most $\hat{n}/300$ steps. Let S_c be a crossing sequence for cell c like in the proof of Lemma 4. Thus the information in S_c can be recorded with no more than $2n/(300 \cdot \log n)$ bits.

We write $X^{C} \hat{} Z^{C}$ for the variation of input $X \hat{} Z$ where all blocks that belong to $\tilde{L}$ have been censored. We use an auxiliary TM $\tilde{P}$ that computes the concatenation $\tilde{L}\hat{}$ of all blocks in $\tilde{L}$ from $X^{C} \hat{} Z^{C}$, S_c, the number of cell c and the final machine state of computation C. Similarly as the TM P in the proof of Lemma 4 this TM $\tilde{P}$ tries successively all possible fill-ins for the censored blocks of $\tilde{L}$. A somewhat delicate cut and paste argument shows that $\tilde{L}\hat{}$ is the only fill-in that allows a computation of M on the resulting input which generates S_c on cell c and halts in the same final state as computation C.

The existence of TM $\tilde{P}$ implies that $K(\tilde{L}\hat{} \mid X^{C} \hat{} Z^{C}) \leq 3n/(300 \cdot \log n)$. On the other hand our choice of X implies that $K(\tilde{L}\hat{} \mid X^{C} \hat{} Z^{C}) \geq 5n/(300 \cdot \log n)$. This contradiction finishs the proof of Lemma 6.

Lemma 6 implies that the head of M crosses in computation C at least $n/(300 \cdot \log \log n)$ often the $n/(300 \cdot \log n)$ cells of D_m. This takes at least $n^2/(300^2 \cdot \log n \cdot \log \log n)$ steps. On the other hand computation C uses by assumption only $g(n) = o(n^2/(\log^2 n \cdot \log \log n)$ steps. This contradiction finishs the proof of Lemma 3 (Main Lemma).

Sketch of the proofs of Theorem 1 and Theorem 3 :

In the deterministic case a simpler proof and --as we will see below-- a sublanguage L of L_I suffice. Fix a deterministic 1-tape TM M that accepts L_I. Assume for a contradiction that for every $c \in \mathbb{N}$ there is an infinite set $N_c \subseteq \mathbb{N}$ s.t. M accepts all inputs $I \in L_I$ with $|I| \in N_c$ in at most $|I|^2/c$ steps. Since M is deterministic it processes the first part X of the input $X \hat{} Z$ independently from the choice of the second part Z. Therefore we can wait with the definition of Z until M has processed X. For

X we choose as before a string with $K(X) \geq |X| =: n$. We cut X into blocks of length $c^{1/3}$. The "Desert Lemma" yields a "desert" D of $\tilde{n} := n/c^{1/3}$ cells on the work tape of M and sets L_D and R_D of $\tilde{n}/2 - 2n/c^{1/2}$ blocks each with properties as in Lemma 4 . We then define Z s.t. $X \hat{} Z = Y_{X,L_D,R_D}$. One might have to add some padding to this input to make sure that its length is in N_c , but this causes no problem for the estimate below. There are obviously $\tilde{n}/2 - 2n/c^{1/2} \geq$ (for large c) $\tilde{n}/4$ $L_D - R_D$ pairs in Z . Via Lemma 6 one sees again that for at least $1/3$ of these pairs the work head of M crosses D_m while its input head reads from that pair. Thus machine M uses at least $\tilde{n}^2/\ 3 \cdot 4 \cdot 3 = n^2/(36 \cdot c^{2/3})$ steps. For large enough c this exceeds M's time bound of $|X \hat{} Z|^2/c \leq 16n^2/c$ steps.

The preceding argument shows that in the deterministic case it is sufficient to consider a somewhat simpler sublanguage L of L_I where the virtual heads of M' perform at most one sweep in each direction.

REFERENCES

[1] R.V. Book, S.A. Greibach, B. Wegbreit, Time and tape bounded Turing acceptors and AFL's, J. Comp. Syst. Sci. 4 (1970), 606 - 621

[2] P. Duris, 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, R. Reischuk, Two nonlinear lower bounds, Proc. 15th ACM STOC (1983), 127 - 132

[4] Z. Galil, private communication

[5] J. Hartmanis, R.E. Stearns, On the computational complexity of algorithms, Trans. Amer. Math. Soc. 117 (1965), 285 - 306

[6] F.C. Hennie, One-tape, off-line Turing machine computations, Inf. and Control 8 (1965), 553 - 578

[7] F.C. Hennie, R.E. Stearns, Two-tape simulation of multitape Turing machines, J. ACM 13 (1966), 533 - 546

[8] W. Maass, A. Schorr, Speed-up of 1-tape Turing machines by bounded alternation, in preparation

[9] W.J. Paul, On-line simulation of k+1 tapes by k tapes requires nonlinear time, Proc. 23rd IEEE FOCS (1982), 53 - 56

[10] W.J. Paul, N. Pippenger, E. Szemeredi, W. Trotter, On determinism versus nondeterminism and related problems, Proc. 24th IEEE FOCS (1983), 429 - 438

[11] M.O. Rabin, Real time computation, Israel J. of Math. 1 (1963), 203 - 211