INFORMATION AND CONTROL 60, 1–11 (1984)

Two Nonlinear Lower Bounds for On-Line Computations

PAVOL DŪRIŠ*

Slovak Academy of Science

ZVI GALIL*

Columbia University and Tel-Aviv University

WOLFGANG PAUL

IBM, San Jose

AND

RUEDIGER REISCHUK

University of Bielefeld

The following lower bounds for *on-line* computation are proved: (1) Simulating two-tape *nondeterministic* machines by one-tape machines requires $\Omega(n \log n)$ time. (2) Simulating k-tape (deterministic) machines by machines with k-pushdown stores requires $\Omega(n \log^{1/(k+1)} n)$ time.

INTRODUCTION

One of the major goals of theoretical computer science is proving nontrivial lower bounds on the (time or space) complexity of specific problems. Unfortunately, despite continued research effort for the last ten years, the success in proving lower bounds has been minimal. The only known general lower bounds are at least exponential [2, Chap. 11]. For no specific problem in *NP* can we prove a nontrivial lower bound on its time complexity. Being unable to prove general lower bounds, researchers considered restricted models of computation or restricted versions of lower bounds. Here are a few examples:

* Research supported by National Science Foundation Grant MCS-8303139.

0019-9958/84 $3.00

— Straight line programs
— Decision tree model
— Monotone circuits
— On-line computation
— Time-spece trade-offs: lower bounds on time given upper bounds on space
— Fooling automata: showing that certain automata cannot accept certain languages.

As a result of restricting the problem, it has been possible to prove some lower bounds in these restricted models. However, even in these cases results are nontrivial and many open problems exist.

There are several reasons why we try to prove restricted lower bounds. Sometimes the attempts to prove restricted lower bounds bring about the discovery of interesting techniques that may be useful elwewhere. The study of restricted lower bounds enables us to determine how far our current techniques and their refinements can bring us. It helps us to identify the simplest problems, where current techniques fail. Once these problems are identified, the new ideas which will lead to their solution may be useful in proving general lower bounds.

We prove here two lower bounds for restricted models of computation. In both cases we consider *on-line* computation. By on-line we mean having an additional *one-way* input tape. We also restrict the storage used. Our two results are:

(1) Two tapes versus one for nondeterministic machines—an $\Omega(n \log n)$ lower bound. (Previously, we knew only that no real-time simulation exists [4]. More recently [5] we proved an $\Omega(n \log \log n)$ lower bound.)

(2) Tapes versus pushdown stores (for deterministic machines)—an $\Omega(n \log^{1/(k+1)} n)$ lower bound. (No lower bound was previously known.)

The first result refines and extends the use of the crossing sequence argument. The second result extends the use of the information theoretic approach to proving lower bounds.

Two Tapes versus One for Nondeterministic Machines

It is known that a nondeterministic machine with two tapes can simulate a k-tape linear-time machine in real time [3]. So, for nondeterministic machines, the only question left concerning the influence of the number of tapes on the computing power between one- and two-tape machines. (Recall that the input tape is not counted.)

The currently best upper bound for simulating a real-time two-tape machine by a one-tape machine is (the tivial) $O(n^2)$. In [4] we showed that two tapes are better than one. We considered the language $L = \{x \# x \mid |x| = 2^m \text{ for some } m\}$, and showed that it is accepted in real time by a two-tape (deterministic) machine and cannot be accepted in real time by a one-tape machine. Our first result is:

THEOREM 1. *Any one-tape on-line nondeterministic machine that accepts L requires $\Omega(n \log n)$ time.*

For the sake of comparison, we roughly sketch the previous result that no real-time simulation exists: we assume that L is accepted by a real-time one-tape machine M and derive a contradiction. For every $y \in L$ we arbitrarily fix an accepting computation. Then, by a simple counting argument we show the existence of $y \in L$, such that its accepting computation must use at least a linear amount of space. We now consider the accepting computation of M on y. We divide the working tape into blocks of some constant length. By the pigeon hole argument, there must be a block such that the total time it is scanned is small. Another counting implies that there are two identical crossing sequences in this block. As a result, M must accept a shorter input. This gives a contradiction, since the shorter input is not short enough. (It is shorter by a constant and cannot be of length $2 \cdot 2^k + 1$.)

The global strategy of the proof of Theorem 1 is similar. We will fool a machine M that supposedly accepts L in time $< cn \log n$ (for some c that we specify) by showing that it must accept a shorter input that is not short enough. This is achieved by finding two identical crossing sequences in between which: (1) M reads an input symbol, and (2) M does not spend much time. Finding such crossing sequences will not be as immediate as in our previous paper [4].

Proof of Theorem 1. We assume that L is accepted by an on-line nondeterministic Turing machine M that has q internal states and t different symbols for its working tape. We consider the behavior of M on inputs from L_n, the subset of L of strings of length $m = 2n + 1$, $n = 2^k$. We choose n to be large enough for some of the inequalities below to be true. For each input y in L_n we arbitrarily fix an accepting computation of M. We refer to it as *the* accepting computation of y.

During a computation M sometimes reads new input symbols. Without loss of generality, when M does not read a new input symbol its next state depends on its state and the symbol on the working tape but not on the input symbol. This can be easily achieved by having reading states (when M reads a new input symbol) and nonreading states. The latter can encode the last input symbol that M has read.

Let

$$\alpha = \max(\log 4q, \log 2t) \tag{1}$$

and fix a constant c small enough such that for sufficiently large n $(n > n_0(c))$

$$\alpha^{8c\alpha \cdot \log n + 1} < n^{1/4}/(32\alpha c \cdot \log n). \tag{2}$$

We assume that M accepts L_n in time $t(n) < cn \log n$, and derive a contradiction following several definitions and lemmas. (Note that the length of the input is actually m, but it does not matter because $m = 2n + 1$.)

For y in L_n, $y = x \# x$, consider the part of the computation of M on y until M reads the $\#$. A tape cell on the working tape of M is called an *important cell* if M read a new input symbol in at least one of the times it scanned it. Two important cells are said to be *close* if their distance is at most $2\lceil 3\alpha c \log n \rceil$. A *bunch* is a maximal sequence of important cells such that two successive elements in it are close.

Assume there are r bunches, $r \geqslant 1$. Consider the first (last) important cell of the ith bunch, $1 \leqslant i \leqslant r$. Of the $\lceil 3\alpha c \log n \rceil$ boundaries to the left (right) of it, choose the one with the shortest crossing sequence. Denote this boundary by b_{2i-1} (b_{2i}) and the crossing sequence by A_{2i-1} (A_{2i}). We call the part of the tape between b_{2i-1} and b_{2i} the ith *piece*. Let y_i be the contents of the ith piece when M reads the $\#$. For $1 \leqslant i \leqslant 2r$ let $\bar{A}_i$ be a modified version of A_i: if $A_i = (q_1, \ldots, q_l)$, then $\bar{A}_i = ((q_1, \varepsilon_1), \ldots, (q_l, \varepsilon_l))$, $\varepsilon_j \in \{0, 1\}$ for $1 \leqslant j \leqslant l$ and $\varepsilon_j = 1$ iff in between the jth and the $(j+1)$st crossing M crossed an adjacent boundary $(b_{i-1}$ or $b_{i+1})$.

The *profile of* x is the following set: $\{(\bar{A}_{2i-1}, \bar{A}_{2i}, y_i)$, $i = 1, \ldots, r$, the (important) cell scanned and the state when M reads the $\#\}$. See Fig. 1.

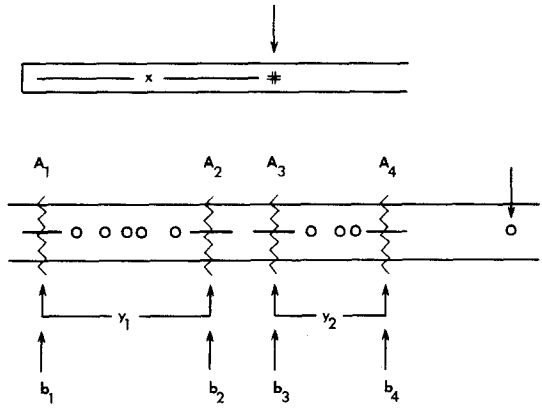

FIG. 1. The first two pieces. Circles denote important cells.

LEMMA 1. *If $x_1 \neq x_2$, then x_1 and x_2 have different profiles.*

Proof. Given an accepting computation of $y = x \# x$ in L_n we define the *special configuration* of this computation to be the triple (z, i, p), where z is the contents of the working tape, i the position of the head on the working tape, and p the state of M when it reads the $\#$. Assume C is the special configuration of the accepting computation of $x_1 \# x_1$. If x_1 and x_2 have the same profile, then there is *an* accepting computation of $x_2 \# x_2$ with the *same* special configuration C. This follows from the usual cut and paste trick and the fact that M did not read any input when it scanned the parts between b_{2i} and b_{2i+1} of the working tape. (These parts are the ones that do not appear in the profiles.) But this would imply that M must also accept $x_1 \# x_2$ (another cut and paste), and hence $x_1 = x_2$. ∎

LEMMA 2. *There is a $y = x \# x \in L_n$, such that the total length of all the pieces in the profile of x ($\sum_{i=1}^{r} |y_i|$) is at least $n/3\alpha$.*

Proof. By the choice of the boundaries b_i

$$\sum_{i=1}^{2r} |A_i| \cdot \lceil 3\alpha c \log n \rceil \leqslant t(n) \leqslant cn \log n.$$

Hence $\sum_{i=1}^{2r} |A_i| \leqslant n/3\alpha + o(n)$, and by (1) the number of possible $\{\bar{A}_i\}$'s is at most $2^{n/3+o(n)}$. If we always have $\sum_{i=1}^{r} |y_i| < n/3\alpha$, then by (1) the number of all possible $\{y_i\}$'s would be $2^{n/3}$, and the number of possible profiles would be $2^{2n/3+o(n)} < 2^n$ for n large enough. But by Lemma 1, the latter is impossible. ∎

From now on we restrict attention to the y of Lemma 2. We partition the working tape into *blocks* of $n^{1/2}$ consecutive tape cells. Each *block* we further partition into $n^{1/4}$ subblocks. (Recall, that $n = 2^k$.) A subblock is *important* if it contain an important cell. Note that the total number of complete blocks (subblocks) is at most $t(n)/n^{1/2} \leqslant cn^{1/2} \log n$ ($cn^{3/4} \log n$).

LEMMA 3. *There are at least $n^{3/4}/4\alpha$ important subblocks.*

Proof. Let Y be the union of all the pieces. A nonimportant subblock which hits Y can contain at most $2\lceil 3c\alpha \log n \rceil$ cells of Y. (At most the last $\lceil 3c\alpha \log n \rceil$ in a piece and the first $\lceil 3c\alpha \log n \rceil$ in the next piece.) So altogether nonimportant subblocks can contain at most $cn^{3/4} \log n \cdot 2\lceil 3c\alpha \log n \rceil = o(n)$ cells in Y. By Lemma 2, $|Y| \geqslant n/3\alpha$ and the important subblocks must cover at least $n/3\alpha - o(n)$ cells of Y. Hence their number must be at least $n^{3/4}/4\alpha$, for n large enough. ∎

Let S be the set of important subblocks in which M spent less than $8c\alpha n^{1/4} \log n$ time.

LEMMA 4. $|S| \geqslant n^{3/4}/8\alpha$.

Proof. Otherwise by Lemma 3, M spends at least $8c\alpha n^{1/4} \log n$ time in more than $n^{3/4}/8\alpha$ subblocks for a total $> cn \log n$. ∎

LEMMA 5. *Each subblock in S has a crossing sequence of length at most* $8c\alpha \log n$.

Proof. Obvious. ∎

Let R be the set of blocks that contain at least $n^{1/4}/(16\alpha c \log n)$ subblocks in S.

LEMMA 6. $|R| \geqslant n^{1/2}/16\alpha$.

Proof. Otherwise, the number of subblocks in S would be smaller than

$$n^{1/2}/16\alpha \cdot n^{1/4} + cn^{1/2} \log n \cdot n^{1/4}/(16\alpha c \log n) \leqslant n^{3/4}/8\alpha,$$

contradicting Lemma 4. ∎

LEMMA 7. *There exists a block B in R such that M spent at most* $16\alpha c n^{1/2} \log n$ *steps on B.*

Proof. Otherwise the time bound of M would exceed $|R|\, 16\alpha c n^{1/2} \log n \geqslant cn \log n$. ∎

We are ready to complete the proof of Theorem 1. Consider the block B. It contains at least $n^{1/4}/(16\alpha c \log n)$ subblocks in S: $B_1, B_2, \ldots$. Consider every other one: $B_1, B_3, \ldots$. There are at least $n^{1/4}/(32\alpha c \log n)$ of them. By Lemma 5, each one contains a crossing sequence of length at most $8c\alpha \log n$. By (2), two of them must be identical. Hence M accepts a shorter input (due to at least one important subblock in between them). By Lemma 7, the length of this shorter input is at least $2 \cdot 2^k + 1 - O(k2^{k/2}) > 2 \cdot 2^{k-1} + 1$ for k large enough. ∎

Remark. Theorem 1 answers affirmatively problem 2 in [4]. L can be easily accepted in time $O(n \log n)$ by an on-line deterministic one-tape Turing machine. Theorem 1 shows that even allowing the machine to be nondeterministic it still requires time $\Omega(n \log n)$. Consequently we have an interesting example where nondeterminism does not help.

TAPES VERSUS PUSHDOWN STORES

In this section we use a slightly different notion of on-line computation. Let M be a multitape (multipushdown) deterministic Turing machine that given sequences $E = e_1, e_2 \cdots$ of input symbols as an input tape produces sequences $A = a_1, a_2 \cdots$ of output symbols as an output tape. M works *on-line* if for every input E the computation of M given E proceeds in stages 1, 2,... such that for all i during the ith stage input symbol e_i is read, some computation is performed and output symbol a_i is printed. M works in *real time* if there is a constant c such that for all inputs E and all i the ith stage of the cmputation of M given E consists of at most c steps. Machine S *simulates* machine M if for all inputs E machine S given E produces the same output as machine M given E.

We use the concepts of on-line and real-time computations, Kolmogorov-complexity of strings, random strings and overlap. Their definitions can be found in [9]. It is well known that k-tape Turing machines can be simulated in real time by $2k$-pushdown store (pds) machines. On the other hand, on-line simulation of n-time bounded $(k+1)$-pds machines by k-pds (or even k-tape) machines requires time $\Omega(n \log^{1/(k+1)} n)$ [9]. Here, we observe that for the purpose of the lower bound proof in [9], k-tape machines behave like $(k+1)$-pds machines and show:

THEOREM 2. *For all k there is a k-tape Turing machine M_k that works in real time such that every k-pushdown machine S that simulates M_k on-line is $\Omega(n \log^{1/(k+1)} n)$-time bounded.*

We assume that the reader is familiar with [9].

DEFINITION OF M_k. We number the tapes of M_k from 1 to k. Each tape has two tracks. Each cell of each track can store 0 or 1. Initially, all cells store the blank symbol b. Input symbols for M_k have the form (h, d, t, a) where $h \in \{1,..., k\}$, $d \in \{\text{left, right}\}$, $t \in \{1, 2\}$ and $a \in \{\text{print 0, print 1, do nothing}\}$. Upon receiving input symbol (h, d, t, a) machine M_k moves the head on tape h one step in direction d, outputs the symbol it finds on track t of the cell it just moved to and then performs action a on that cell. For natural numbers l, the input sequence (h, left; t, do nothing) $\cdots$ (l-times) is called an *l-loop for track t of tape h.*

INPUT SEQUENCES FOR M_k

Let S be a k-pushdown machine with alphabet $\{0, 1, b\}$ that simulates M_k on-line. Let n be large enough for the analysis that follows, let

$s = \lfloor (\log n)/2 \rfloor$ and $d = \lfloor (\log n)^{1/(k+1)} \rfloor$. For convenience, assume $8k(k+1)$ divides n. Let $w = zw_1 \cdots w_k \in \{0,1\}^*$ be a random string where $|w_h| = n2^s d^h$ for $h \in \{1,\ldots,k\}$ and $|z| = n2^s$. Let G be an input sequence that makes head 1 of M_k print z from right to left on track 1 of tape 1.

For $h \in \{1,\ldots,k\}$ let C_h be an input sequence that makes head h of M_k print w_h from left to right on track 2 of tape h. Let $D_1 = C_1$ and for $i \in \{2,\ldots,k\}$ obtain D_i from D_{i-1} by inserting d commands from C_i after avery command from C_{i-1} that is already inserted in D_{i-1}. The sequence $E = D_k$ has length

$$\begin{aligned} |E| &= n2^s(d + d^2 + \cdots + d^k) \\ &= \Theta(n2^s \log^{k/(k+1)} n), \end{aligned}$$

and any subsequence E' of E of length $n'(d + d^2 + \cdots + d^k)$ that begins with a command from C_1 makes M_k write $n'd^h$ symbols from w_h on tape h for $h = 1,\ldots,k$.

Divide E into 2^s parts E_i of length $n(d + \cdots + d^k)$. The final input sequence for M_k will have the form $GE_0F_0E_1F_1\ldots$, where each F_i consists of up to $s+1$ l-loops. The choice of these l-loops depends on the behavior of the simulator.

The Choice and Effect of the l-Loops in F_i

Let I be a time interval, i.e., a sequence of steps in a computation of S, let $h \in \{1,\ldots,k\}$, and let e be a natural number. We say S *extends* pushdown h be e during I, if the number of steps in that interval when S pushes a symbol on pushdown h minus the number of steps in that interval when S pops a symbol from pushdown h is e. $\omega(I)$ denotes the internal overlap of time interval I.

We will partition the computation of the simulator S given $E_0F_0E_1F_1\cdots$ into intervals $I_{s,0}, I_{s,1},\ldots$, where interval $I_{s,j}$ lasts from the first step of the simulation of E_j to the last step of the simulation of F_j. For $0 \leqslant i < s$ and $0 \leqslant j < 2^{s-i}$ intervals $I_{i,j}$ are defined by

$$I_{i,j} = I_{i+1,2j} \cup I_{i+1,2j+1}.$$

The weight $w(I_{i,j})$ of interval $I_{i,j}$ is defined as 2^{s-i}. For all j, let $A_j = \{(a,b) \mid \text{interval } I_{a,b} \text{ ends with the last step of } F_j\}$. For each $(a,b) \in A_j$ there will be a part $F_{a,b}$ in F_j, which will be either an l-loop or empty.

Let $B \subset A_j$ and suppose that for all $(a,b) \in B$ part $F_{a,b}$ has already been defined and simulated by machine S. Let t be the last step performed so far

by S. For each $(a, b) \in A_j - B$ let $I_{a,b}(t)$ be the time interval in the computation of S that begins with the first step $I_{a,b}$ and lasts until step t. Let

$$m = n/(8(k+1)).$$

If $\omega(I_{a,b}(t)) \geqslant m \cdot w(I_{a,b})$ for all $(a, b) \in A_j - B$, then all parts $F_{a,b}$ with $(a, b) \in A_j - B$ are empty and F_j is completed. Otherwise, we pick $(a, b) \in A_j - B$ such that

$$\omega(I_{a,b}(t)) \leqslant m \cdot w(I_{a,b})$$

and define an l-loop $F_{a,b}$. For the remainder of this section, a, b, and t will be fixed and we will use the shorthand I for $I_{a,b}(t)$ and $w(I)$ for $w(I_{a,b})$.

Let E' be the portion of E that is simulated during I. Then $|E'| = nw(I)(d + \cdots + d^k)$. Partition it into k parts $E^1 \cdots E^k$ of equal length. This induces a partition of I into parts $I^1 \cdots I^k$, where each interval I^i begins with the first step of the simulation of E^i. Let $N = nw(I)$ and $N' = N/(8k(k+1))$. Following the argument in [1, 9] we say S *neglects* tape h of M_k during I if every pushdown that is extended by at least $N'd^h$ during interval I^h is extended by at least $N'd^{h+1}$ during $I^{h+1} \cdots I^k$.

Now we consider two cases:

Case 1. At least one pushdown of S extends by less than $N'd$ during I.

In this case, S behaves (for the purpose of the lower bound proof in [9]) like a $(k-1)$ pushdown machine and the arguments in [9] apply almost literally. One finds $h \in \{1,..., k\}$ such that S neglects tape h of M_k during I. $F_{a,b}$ is defined as an l-loop for track 2 of tape h of M_k that makes head h of M_k sweep over the portion u of w_h that was written on tape h during interval I^h; i.e.,

$$l = Nd^h(k-h+1)/k.$$

The simulation of $F_{a,b}$ will take S at lest $Nd^{h+1}/25k(k+1)) \geqslant c(2l)d$ steps by Lemma 2 in [9].

Case 2. All pushdowns of S extend by at least $N'd$ during I. In this case, we can treat track 1 of tape 1 of M_k as a tape with number $h = 0$ that has been neglected by S during I. $F_{a,b}$ is defined as an l-loop for track 1 of M_k with $l = N$, and the simulation of $F_{a,b}$ will take S at least $Nd/(25k(k+1)) \geqslant c(2l)d$ steps by a simple version of the proof of Lemma 2 in [9]: Let u be the contents of track 1 of the l tape cells immediately to the left of the head of tape 1 of M_k. For $1 \leqslant i \leqslant k$ let x_i be the portion of w_i that was written on tape i of M_k during I. Assuming the claim is false, one derives the contradiction

$$|u| - O(\log n) \leqslant K(u \mid x_1 \cdots x_k) \leqslant |u|/2 + O(\log n).$$

(Since all pushdowns of S extend by much during I and I has a small overlap, we can simulate M_k on I given $x_1 \cdots x_k$ and only small part of the top of the k pushdown stores. The latter is bounded by the overlap $\leqslant |u|/2$.)

The rest of the proof follows as in [9]: let $F = F_0 \cdots F_{2^s-1}$. In both cases above S is slower than M_k by the factor proportional to d $(=\log^{1/(k+1)} n)$ when it works on F. So if $|F| \geqslant |E|$ the proof is complete. If $|F|$ is small, this means that many $F_{a,b}$'s are empty. So many intervals with large overlap exist. In this case the generalized version of the overlap lemma (Lemma 1 in [9]) yields the desired lower bound.

Open Problems

(1) Can the gap between the trivial $O(n^2)$ upper bound and the lower bound of Theorem 1 be narrowed or closed?

(2) Is the $\Omega(n \log^{1/(k+1)} n)$ lower bound of Theorem 2 best possible?

(3) Are two heads on the same tape better than two (or even three heads on different tapes? (They are not better than four [7].)

(4) Can $2(k)$ tapes sometimes be better than

$$3(k+1, k+2, \ldots \text{ or even } 2k-1) \text{ pds's?}$$

(5) Find tight bounds for the simulation of $k+1$ tapes by k (or by $j < k$) tapes $k \geqslant 2$ (known upper and lower bounds are $O(n \log n)$ [6] and $\Omega(n \log^{1/k+1} n)$ [9]).

(6) Find tight bounds for the simulation of k (or 2) tapes by one tape (known upper and lower bounds: $O(n^2)$, $\Omega(n \log n)$ by Theorem 1).

(7) Can Kolmogorov–Uspenskii machines simulate storage modification machines in real time?

(8) Can Turing machines with tree-tapes simulate two-dimensional Turing machines in real time?

Note added in proof. W. Maass (1984, Quadratic lower bounds for deterministic and nondeterministic one-tape Turing machines, *in* "Proceedings 16th Annual ACM Symp. on Theory of Computing," pp. 401–408, ACM, New York) has recently improved Theorem 1 considerably. He defined another language $\hat{L}$ that is recognized by a deterministic two-tape machine in real time and proved that (1) any one-tape deterministic Turing machine that recognizes $\hat{L}$ requires $\Omega(n^2)$ time, and (2) any one-tape nondeterministic Turing machine that recognizes $\hat{L}$ requires $\Omega(n^2/\log^5 n)$ time. The second bound can be improved to $\Omega(n^2/\log^2 n)$ by slightly modifying Maass' proof. Consequently, problem (6) above is solved, and the gap mentioned in problem (1) is almost closed. It would still be interesting to close it entirely (as in the deterministic case).

M. Li (1984, On one-tape versus two stacks, Department of Computer Science, Cornell University) has independently obtained similar results: an $\Omega(n^2)$ lower bound for the deterministic case but only an $\Omega(n^{1.5})$ lower bound for the nondeterministic case.

REFERENCES

1. AANDERAA, S. O. (1974), On k-tape versus $(k-1)$-tape real time computation, *in* "Complexity of Computation" (R. M. Karp, Ed.), pp. 75–96, Amer. Math. Soc., Providence, R.I.
2. AHO, A. V., HOPCROFT, J. E., AND ULMAN, J. D. (1974), "The Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, Mass.
3. BOOK, R. V., AND GREIBACH, S. A. (1970), Quasi real time languages, *Math. Systems Theory* **4**, 97–111.
4. DŪRIŠ, P., AND GALIL, Z. (1984), Two tapes are better than one for nondeterministic machines, *SIAM J. Comput.* **13**, 219–227.
5. DŪRIŠ, P., GALIL, Z., PAUL, W. J., AND REISCHUK, R. (1983), Two nonlinear lower bounds, *in* "Proceedings, 14th Annual ACM Symp. on Theory of Computing," pp. 127–132, ACM, New York.
6. HENNIE, F. C., AND STEARNS, R. E. (1966), Two-tape simulation of multitape Turing machines, *J. Assoc. Comput. Mach.* **13**, 533–546.
7. LEONG, B. L., AND SEIFERAS, J. I. (1981), New real-time simulation of multihead tape units, *J. Assoc. Comput. Mach.* **28**, 166–180.
8. PAUL, W. J. (1981), On heads versus tapes, *in* "Proceedings, 22nd Annual IEEE Symp. on Foundation of Computer Science, IEEE Computer Society, Los Angeles," pp. 68–73.
9. PAUL, W. J. (1982), On-line simulation of $k+1$ tapes by k tapes requires nonlinear time, *in* "Proceedings, 22nd Annual IEEE Symp. on Foundation of Computer Science, IEEE Computer Society, Los Angeles," pp. 53–56.
10. REISCHUK, R. (1980), A fast implementation of a multidimensional storage into a free storage, *in* "Proceedings, 7th Int. Colloq. on Automata, Languages and Prog.," pp. 531–542, Springer-Verlag, New York.
11. SCHONHAGE, A. (1980), Storage modification machines, *SIAM J. Comput.* **9**, 490–508.
12. STOSS, H. J. (1970), k-Band-Simulation von k-kopf-Turing Machinen, *Computing* **6**, 309–317.