MAC TECHNICAL MEMORANDUM 40

AN IMPROVED OVERLAP ARGUMENT FOR ON-LINE MULTIPLICATION

Michael S. Paterson
Michael J. Fischer
Albert R. Meyer

January 1974

This research was supported in part by the National Science Foundation under research grant GJ-34671; in part by the Artificial Intelligence Laboratory supported by the Advanced Research Projects Agency of the Department of Defense which was monitored by ONR Contract No. N00014-70-A-0362-0003, and in part by the Science Research Council of the U.K.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

CAMBRIDGE MASSACHUSETTS 02139

This empty page was substituted for a blank page in the original document.

AN IMPROVED OVERLAP ARGUMENT FOR ON-LINE MULTIPLICATION[†]

Michael S. Paterson
University of Warwick, Coventry, England

and

Michael J. Fischer and Albert R. Meyer
Massachusetts Institute of Technology, Cambridge, Massachusetts

ABSTRACT

A lower bound of cNlogN is proved for the mean time complexity of an on-line multitape Turing machine performing the multiplication of N-digit binary integers. For a more general class of machines which includes some models of random-access machines, the corresponding bound is cNlogN/loglogN. These bounds compare favorably with known upper bounds of the form $cN(\log N)^k$, and for some classes the upper and lower bounds coincide. The proofs are based on the "overlap" argument due to Cook and Aanderaa.

[†]Much of this work was carried out at the University of Warwick with partial support from the Science Research Council of the U.K. It was also supported in part by the National Science Foundation under research grant GJ-34671 to MIT Project MAC, and in part by the Artificial Intelligence Laboratory, an MIT research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research contract number N00014-70-A-0362-0003.

1. INTRODUCTION

A challenging problem in the field of computational complexity is to prove lower bounds on the computing time for naturally defined algorithms executed by realistically powerful machinery. For a serial machine whose task is to map an input string to an output string, a trivial lower bound for many mappings is the number of steps required to read the input string. There are a number of combinatorial techniques, involving for example crossing sequences [cf.6, §10,4], which are adequate to derive nontrivial lower bounds but only for rudimentary machines such as single-tape Turing machines. The powerful diagonalization techniques are of use only for an input/output mapping sufficiently structured to encode machine computations.

In this paper we expound and develop further the "overlap" argument introduced by Cook and Aanderaa [3], which establishes a nonlinear lower bound on the time required by a very general class of machines to perform multiplication of binary integers. (A similar argument has been used again recently by Aanderaa [1].) Our contribution relative to [3] is firstly that the main line of proof is somewhat shortened and simplified, secondly that the lower bound in [3] is increased by a factor loglogn and

is shown to hold for the average rather than just the worst case, and thirdly a new observation is made which yields an even stronger result for the case of multitape Turing machines. In some cases we show our new results to be optimal to within a constant factor by exhibiting suitable multiplication algorithms.

Our main results are lower bounds for "on-line" multiplication. A mapping from an input string to an output string is said to be carried out on-line if for all n the n^{th} output symbol is printed after the n^{th} and before the $(n+1)^{st}$ input symbol is read*. For on-line multiplication the multiplicands are given in binary, least significant digit first, and each input symbol encodes the two corresponding input digits. We may as well assume that, for N-digit arguments, only the least significant N digits of the product are to be produced.(The remaining digits may be obtained, if desired, by concatenating N zeros to the arguments.) On-line multiplication is of course possible, though a naive implementation may take time at least proportional to n between the $(n-1)^{st}$ and n^{th} digits, with therefore a time of order N^2 for an N-digit product. We show here that the minimum average computation time

---

*For technical convenience, we use this strong form of the definition which prohibits an output from being produced too soon. It is not a serious restriction for two reasons: For binary multiplication, the i^{th} digit of the product cannot be determined until the i^{th} digits of the two inputs have been read except when both numbers are even; hence a machine can take advantage of the weaker definition for at most a quarter of all possible inputs, changing our mean-time bounds by only a constant factor. Secondly any BAM may be modified without time loss to obey the strong definition by adding a two-headed linear tape to serve as an output buffer. This does not affect any of our lower bounds. (Cf. [5].)

for on-line multiplication is bounded below and above by functions of the form $N(\log N)^k$, where for the lower bounds k is approximately 1 and for the upper bounds k is approximately 1 or 2 depending on the class considered. The exact results are given in Sections 6,7 and 8. For further background and motivation the reader is referred to [3].

2. MACHINE MODELS

In the class of machines to which our proofs apply, we wish to include not only the familiar multitape Turing machine but also Turing machines with tapes of higher dimension and some suitably tame "random access machines". We shall have to exclude iterative arrays and other machines with unlimited parellelism since these are able to do multiplication in "real-time" [2]. Our definitions follow [3] fairly closely, with minor differences in order to simplify the notation and proof or to take fuller advantage of the power of the proof technique. The reader is assumed to have experience with the basic definitions and techniques of automata theory [6].

A bounded activity machine (BAM) has a deterministic finite-state control which operates with a one-way read-only input tape, a one-way write-only output tape and a storage structure. The storage structure is a countable set of locations each of which can hold a binary value. The store is accessed and modified by a finite, fixed number of work heads whose moves are specified by a finite set of shifts $\varphi_1,\ldots,\varphi_p$. For each i, φ_i is a map from the set of locations into itself, and a head at some location x may be moved in one step to the location $\varphi_i(x)$.

A complete step of the BAM is described as follows. Depending on the state of the finite control, the input tape may be advanced one symbol and precisely one work head must be "moved" by one of the shifts. Then, depending on the control state, the new input symbol (if any), and the value stored in the storage location to which the head is moved, a new value may be stored, an output symbol may be given, and a new control state is entered. Thus, for each given symbol from the input tape or a storage location, there is a unique step at which it is read, and the definition prevents it from being reread later. Moreover, only one work tape symbol is read per step, so we may speak of "the work symbol read at step s".

Various restrictions in this definition, such as binary storage, one head move per step, and the lack of dependence of the new step on the old storage values, are introduced to simplify the exposition and cause a time penalty of at most a constant factor compared with more versatile machines.

We shall say that a computation is real-time if it is on-line and each input symbol is read a fixed number of steps after the previous input. Note that we shall not require the store to be initially "empty" except for the special class of "uniform" machines defined below.

It is easy to design a BAM which can multiply in real time. A suitable storage structure is based on an infinite binary tree, traversed by a single head which takes left or right branches depending on the input

digits. The correct output is either already stored in the location of the tree at the start or else is encoded in the structure itself in an obvious way. For example the structure may have a transformation ψ such that for all x, either $\psi(x) = x$ or else $\psi(\psi(x)) = x$ and $\psi(x) \neq x$. Which alternative holds can be determined for any location by a sequence of a few steps.

Two classes introduced by Cook and Aanderaa ([3]) to evade such an oracular construction are the polynomial-limited and uniform machines defined below. We also add two further classes.

(i) Polynomial-limited.

A storage structure is polynomial-limited if there are constants c,d such that for all locations x and for all t, the number of locations accessible from x in t steps is no greater than ct^d. A BAM with such a structure is a polynomial-limited machine.

(ii) Uniform.

A storage structure is uniform if for each pair of locations x,y, there is a permutation f such that $f(x) = y$ and for each shift φ_i of the structure, $f \circ \varphi_i = \varphi_i \circ f$. A BAM with a uniform structure which is initially "empty" (i.e. each location has the same initial value) is a uniform machine. The reader is referred to [3] for further discussion of these and other classes.

With a suitable form of definition, Turing machines, even with multiple heads and multidimensional tapes, satisfy both restrictions.

Our main result holds for machines satisfying either restriction. The BAM described above which multiplies in real-time satisfies neither.

(iii) One-dimensional multihead multitape Turing machines.

We can obtain a stronger result than for classes (i) and (ii) if we restrict the tapes to be linear, i.e. one - dimensional.

(iv) Oblivious machines.

For this class we turn our attention from the storage structure, which may be arbitrary, to the form of the finite state control or "program". The (single) storage location accessed at each step defines the storage sequence for any computation, and this depends in general on the input. A machine is oblivious if for input sequences of a given length the storage sequence is fixed, i.e. independent of the input symbols. Naturally, the control state and the values inscribed in the store can, and in general do, depend on the input symbols; it is just the movement of heads which is invariant. Our interest in oblivious machines is two-fold. Firstly the restriction permits a very simple proof of an improved lower bound, and secondly it happens that almost all the algorithms proposed or used for multiplication are oblivious or can be made oblivious at the cost of only a constant factor in time.

In section 7 we shall retrospectively consider other classes of machines to which the proof techniques applies.

3. RETRORSE FUNCTIONS

Informally, a function from an input string to the output is retrorse* if the output values in any segment depend very heavily on the input values of the immediately preceding segment, and so the function evaluator needs to "turn back" to the previous input segment. On-line multiplication will be shown to be very retrorse.

We define $K_N = \sum_{2^i < N} 2^{2^i}$, so the binary expansion of K_N has a "1" in position i iff i is a power of two, where the positions are numbered starting with 0 at the right (lower-order) end. The usefulness of K_N is that multiplication of N-digit numbers by K_N is extremely retrorse, and the main proof is simpler than for two-input multiplication. Our first theorem provides a lower bound on the average time for on-line multiplication of an N-digit integer by K_N and hence also on the worst-case time for on-line multiplication. In the second theorem we show that the same bound holds for the average time for general on-line multiplication.

Figure 1 represents the multiplication of K_N by an N-digit number X with result Z. It is drawn in the conventional way with least significance to the right. R and M are non-negative integers, and we shall always take R to be a power of two, 2^r. W represents the subfield of Z consisting of bits $X_{M+R-1}\cdots X_{M+1}X_M$, and Y represents the subfield of Z consisting of bits $Z_{M+2R-1}\cdots Z_{M+R+1}Z_{M+R}$. We will at times think of W and Y as R-bit integers

* We choose not to follow Cook and Aanderaa in their choice of "complex" to describe these functions.

in the range 0 to 2^R-1. To say that W assumes a value i means that we imagine placing the binary representation of i into the W subfield of X. This in turn causes Z to change, for Z always means the product $K_N \cdot X$, and that in turn affects the value of Y. We investigate the dependence in this way of Y upon W.

The way in which Y varies with W of course depends on the remaining bits of X, other than those in W, which we denote by $X \setminus W$. As a number, $X \setminus W$ is the value of the binary string obtained by setting the W-field of X to zero.

For some particular fixed value of $X \setminus W$, let W range through all possible values 0, 1,..., 2^R-1, so $X_i = X \setminus W + i \cdot 2^M$, $0 \le i \le 2^R-1$. Let Z_0, Z_1,... and Y_0, Y_1,... be the corresponding values of Z and Y, that is, $Z_i = K_N \cdot X_i$, and Y_i is the Y-field of Z_i.

If $i < j$, then

$$Z_j - Z_i = (X_j - X_i) \cdot K_N = (j-i) \cdot 2^M \cdot K_N.$$

Since $K_N = K_{2R} + 2^{2R} \cdot \overline{K}$ for some integer $\overline{K}$, we have

$$Z_j - Z_i \equiv (j-i) \cdot 2^M \cdot K_{2R} \pmod{2^{M+2R}}.$$

Now suppose $Y_i = Y_j$. Then

$$Z_j - Z_i \equiv a \cdot 2^M \pmod{2^{M+2R}}$$

for some integer a, $|a| < 2^R$, and hence

$$(j-i) \cdot K_{2R} \equiv a \pmod{2^{2R}}.$$

Since $R = 2^r$,

$$2(2^R-1) \geq K_{2R} = 2^{2^r} + 2^{2^{r-1}} + \ldots + 2^{2^0} \geq 2^R.$$

By the right hand inequality, $(j-i)\cdot K_{2R} \geq 2^R > a$, and hence for the congruence to hold, we must have

$$(j-i)\cdot K_{2R} \geq a+2^{2R} > 2^{2R} - 2^R.$$

From the left-hand inequality for K_{2R},

$$j-i > \frac{1}{2}\cdot 2^R.$$

Hence, for any i, there is at most one $j > i$ such that $Y_i = Y_j$, and we have proved:

Lemma 1. For fixed values of M, R, N and X↘W, each value of Y can arise from at most two values of W.

4. OVERLAP

This concept is the basis for a very elegant counting argument introduced in [3]. It has recently been put to use again by Aanderaa [1]. The motivation for its definition comes from the computation of very retrorse functions. The obvious way in which information about a previous input segment can be obtained is by revisiting locations which were visited when that segment was being read. Overlap is defined in terms only of the storage sequence defined previously. If two successive accesses to the same location ℓ occur at steps s_1 and s_2 $(s_1 < s_2)$, then the pair (s_1, s_2)

is called an overlap pair, ℓ is called the overlap location of s_2, and the value stored in ℓ at step s_1 and referenced at step s_2 is called the overlap value of s_2. The total overlap $\Omega = |\{(s_1,s_2) \mid (s_1,s_2)$ is an overlap pair$\}|$. Clearly the total time $T \geq \Omega$, since each step s is the second component of one or zero overlap pairs depending on whether the location accessed at step s has been accessed before or not.

Let C_1, C_2 be disjoint contiguous time intervals during a computation. We define overlap(C_1, C_2) to be the number of overlap pairs (s_1, s_2) for which $s_1 \in C_1$ and $s_2 \in C_2$.

Without loss of results we assume $N = 2^n$. For any $i = 0,\ldots,n$, which we call the level, define $R_i = 2^i$, and if $S = S_{N-1}S_{N-2}\ldots S_1S_0$ is any string of length N, we partition S into contiguous blocks $S_{i,2^{n-i}-1}$, $\ldots$, $S_{i,1}$, $S_{i,0}$ of length R_i, where $S_{i,j} = S_{(j+1)R_i-1}\ldots S_{j\cdot R_i}$, $0 \leq j \leq 2^{n-i}-1$. If X is the input string, the time interval $C_{i,j}$ starts as the rightmost digit of $X_{i,j}$ is read and continues until the rightmost digit of $X_{i,j+1}$ is about to be read (or until the computation ends if there is no such j+1).

Let $t_{i,j}$ be the length of time of $C_{i,j}$. Clearly the total time of the computation $T = \sum_j t_{i,j}$ for each level i. We define $\omega_{i,j}$ = overlap $(C_{i,j}, C_{i,j+1})$ for all suitable i,j, and also

$$\omega_i = \sum_j \omega_{i,2j}$$

for any i.

Lemma 2. Total overlap $\Omega = \sum_i \omega_i$

Proof. Let (s_1,s_2) be an overlap pair and let i be the least level such that $s_1,s_2 \in C_{i,j}$ for some j. $C_{i,j}$ is the concatenation of the two intervals $C_{i-1,2j}$ and $C_{i-1,2j+1}$ at the next lower level. By our choice of i, $s_1 \in C_{i-1,2j}$ and $s_2 \in C_{i-1,2j+1}$, so (s_1,s_2) contributes to $\omega_{i-1,2j}$ and hence to ω_{i-1}. Suppose it contribues to $\omega_{i',2j'}$. Then $i' \leq i-1$, for s_1 and s_2 belong to the same interval for each level above i-1. $i' \geq i-1$ since if (s_1,s_2) contributes to $\omega_{i',2j'}$, then both s_1 and s_2 are in the same block $C_{i'+1,j'}$ at level i'+1. Hence i' = i-1, and it is clear that j' = j. We conclude that each overlap pair contributes exactly once to exactly one ω_i and hence exactly once to $\sum_i \omega_i$. □

5. COMPUTATIONS WITH SMALL OVERLAP

We consider an on-line computation of some machine $\mathfrak{M}$ from input X to output Z. As before, M and R are fixed numbers, W is the length R subword of X, $X_{M+R-1}\cdots X_M$, and Y is the length R subword of Z, $Z_{M+2R-1}\cdots Z_{M+R}$. Throughout this section, X\W, M and R remain fixed, and we explore how the value of Y changes as the value of W is varied. Unlike the previous sections, Z now represents the output of $\mathfrak{M}$ on input X.

Giving a particular value to W completely determines the computation of $\mathfrak{M}$. Let s_0 be the step which advances the input head onto the first symbol of W, let s_1 be the step which moves the input head off of the last symbol of W, and let s_2 be the step which reads the next input symbol after producing Y. Define interval C_W to be the steps from s_0 to s_1-1, C_Y the steps from s_1 to s_2-1, and let t_W and t_Y be the lengths of time associated with C_W and C_Y, respectively. T is the total time of the computation, and we let $\omega = \text{overlap}(C_W, C_Y)$. This notation is illustrated in Figure 2.

As W varies, so do ω, t_Y, t_W, T and Y. Let $Q(\hat{\omega}, \hat{t}, \hat{T})$ be the total number of different Y values yielded by those W such that $\omega \leq \hat{\omega}$, $t_Y \leq \hat{t}$, and $T \leq \hat{T}$. If $\mathfrak{M}$ is computing a retrorse function, $Q(\hat{\omega}, \hat{t}, \hat{T})$ must be large, and we will use this fact to deduce the constraints on $\hat{\omega}$, $\hat{t}$, and $\hat{T}$ that eventually lead to our lower bound on T.

Our upper bounds on Q depend on the kind of machine, but all are obtained using the same general method. For a given value of W, we observe the computation during the interval C_Y and we record in a suitable way information about W that affects the computation in C_Y, such as the state of the control and the positions of the heads at step s_1 (the beginning of C_Y), the steps during C_Y at which overlap with C_W occurs (W-overlap steps), and the overlap value for each W-overlap step. For each class of machines, enough information will be recorded to ensure the validity of:

Condition 1. Let w and w' be two values for W and y and y' be the corresponding values of Y. If the information recorded for w and w' is the same, then y = y'.

It follows immediately from Condition 1 that the number of different possible information records obtained from values of W for which ω, t_Y, and T are bounded respectively by $\hat{\omega}$, $\hat{t}$, and $\hat{T}$ is an upper bound on $Q(\hat{\omega}, \hat{t}, \hat{T})$,

Lemma 3. There exists a constant C depending only on $\mathfrak{M}$ such that, for $\hat{T} > 1$ and $\hat{\omega} \le \hat{t} \le \hat{T}$:

(a) $Q(\hat{\omega}, \hat{t}, \hat{T}) \le \hat{T}^C \cdot 2^{\hat{\omega}} \cdot \binom{\hat{t}}{\hat{\omega}}$ if $\mathfrak{M}$ is polynomial-limited or uniform;

(b) $Q(\hat{\omega}, \hat{t}, \hat{T}) \le \hat{T}^C \cdot 2^{\hat{\omega}}$ if $\mathfrak{M}$ is a one-dimensional multihead multitape Turing machine;

(c) $Q(\hat{\omega}, \hat{t}, \hat{T}) \le C \cdot 2^{\hat{\omega}}$ if $\mathfrak{M}$ is oblivious.

Proof.

(a) Case 1: $\mathfrak{M}$ is polynomial-limited. The information record consists of the state of the control, the position of each head at time s_1, a subset $\theta = \{t_1, \ldots, t_{\hat{\omega}}\}$ of the integers from 0 through $\hat{t}-1$, and a binary sequence v of length ω. θ is chosen to include the times, relative to s_1, of all the W-overlap steps. The i^{th} bit of v equals the symbol referenced at time $s_1 + t_i$, which will be the overlap value if $s_1 + t_i$ is a W-overlap step. This insures that Condition 1 is satisfied.

The total number of such records is clearly at most $c \cdot 2^{\hat{\omega}} \cdot \binom{\hat{t}}{\hat{\omega}} \cdot H^c$, where H is the number of possible positions for each head at step s_1. Since $\mathfrak{M}$ is polynomial-limited, $H \le t_W^d \le T^d \le \hat{T}^d$, yielding the bound

of part (a).

(a) Case 2: $\mathfrak{M}$ is uniform. This case is exactly like Case 1 except that we do not record the actual head position at time s_1, for the number of possible positions is too large. Rather, we record for each head h the step of C_Y (if any) at which h first visits a square ℓ visited prior to step s_0 together with the time of the first visit to ℓ (which uniquely specifies ℓ). Call such a location ℓ filled. (In the case of more than one head, a small amount of additional information must be recorded to account for the possible interactions among the heads before revisiting a filled location. This argument is presented in more detail in [3].)

If the head h never visits a filled location during C_Y, then because of uniformity the symbols read and written by the head can be uniquely determined from the overlap steps and values, without knowing the position of h at time s_1. On the other hand, if h does visit a filled location ℓ, then the step of C_Y at which ℓ is visited together with ℓ itself uniquely determine the position of the head at step s_1, again by the uniformity condition. ℓ can be specified by the time of its first visit, so there are at most $t_Y \cdot T \leq \hat{T}^2$ different starting positions of a single head h which must be distinguished.

(b) $\mathfrak{M}$ is a one-dimensional multihead multitape Turing machine. $\mathfrak{M}$ is a special case of a polynomial-limited machine, so we may record the state and starting head positions as in (a), Case 1. However, the positions at

which overlap will occur may be specified much more succinctly, for the squares visited during C_W by each head form an interval. Thus, only the endpoints need be named. Since at most 2T locations on a linear tape can be reached in T steps by a given head, there are at most $2T^2$ possible intervals per head. As before, a binary sequence of length ω is sufficient in which to record the overlap values. Thus, the total number of such records is at most $c \cdot H^c \cdot (2\hat{T}^2)^c \cdot 2^{\hat{\omega}}$, where H is as in (a), Case 1.

(c) $\mathfrak{M}$ is oblivious. The positions of the heads at each step are independent of W, so only the state of the control at step s_1 and the values of the overlap locations need to be recorded, giving the bound $C \cdot 2^{\hat{\omega}}$. □

We finish our preparations for the main proof with a combinatorial lemma. By way of motivation, let $\mathfrak{M}$ be a machine that multiplies on-line and let $\hat{\omega}$, $\hat{t}$, and $\hat{T}$ be bounds on ω, t_Y, and T respectively such that $Q(\hat{\omega}, \hat{t}, \hat{T}) < 2^{3R/4}$. By Lemma 1, all but $2 \cdot 2^{3R/4}$ values of W, a vanishing fraction of the 2^R values, cause one of the three bounds to be exceeded.

This gives an implicit lower bound on ω in terms of t_Y and T which says in effect that if T is small, then the total overlap Ω is large, which implies that T is large. Hence, T must be large on the average.

Lemma 4. Let C, R, a be positive constants such that $\log a > 2(C + 3)$. If $0 < \omega \leq t$ and $\omega + t/a + \log T \leq R/(2 \log a)$, then $T^C \cdot 2^{\omega} \cdot \binom{t}{\omega} < 2^{3R/4}$. (All logarithms in this paper are taken to base 2.)

Proof. For any $p \geq q > 0$,

$$\binom{p}{q} \leq \frac{p^q}{q!} < \left(\frac{pe}{q}\right)^q$$

by Stirling's formula. Assume the hypothesis, so $\omega < R/(2 \log a)$, $t < aR/(2 \log a)$, and $\log T < R/(2 \log a)$.

$$T^C \cdot 2^\omega \cdot \binom{t}{\omega} < T^C \cdot \binom{2t}{\omega}$$

which is monotonic increasing in ω, t and T since $\omega \leq t$. So

$$\begin{aligned} T^C \cdot \binom{2t}{\omega} &< 2^{CR/(2 \log a)} \cdot \binom{aR/\log a}{R/(2 \log a)} \\ &< 2^{CR/(2 \log a)} \cdot (2ae)^{R/(2 \log a)} \\ &< 2^{3R/4}. \quad \square \end{aligned}$$

6. MAIN RESULTS AND PROOFS

Theorem 1. There is a constant C such that for any BAM $\mathfrak{M}$ which for all N multiplies N-digit numbers by K_N on-line, the mean time T(N) over all numbers of length N satisfies the following bounds for all sufficiently large N.

(i) If $\mathfrak{M}$ is polynomial-limited or uniform,

$$T(N) > CN \log N/\log \log N;$$

(ii) If $\mathfrak{M}$ is a one-dimensional multihead multitape Turing machine or is oblivious,

$$T(N) > CN \log N.$$

Proof. Suppose first that M is polynomial limited or uniform. We may assume that $\log\log N > 2(C + 3)$ where C is as in Lemma 3, and define $a = \log N$. We consider again the situation depicted in Figures 1 and 2, where $X \setminus W$ is fixed and W is allowed to vary. Applying Lemmas 1, 3(a) and 4, we deduce that the number of distinct W's for which $\omega + t/a + \log T \le R/(2 \log a)$ is less than $2 \cdot 2^{3R/4}$. Hence certainly,

$$\underset{W}{\text{mean}} \quad (\omega + t/a + \log T) > R/(3 \log a) \quad \text{for } R \ge 16.$$

Since this inequality holds for all values of $X \setminus W$,

$$\underset{X}{\text{mean}} \quad (\omega + t/a + \log T) > R/(3 \log a)$$

where the mean is taken over all N-digit numbers. If we assume that $\underset{X}{\text{mean}}\ T < N^2$ then

$$\underset{X}{\text{mean}} \log T \le \log \underset{X}{\text{mean}}\ T < 2\log N$$

since the geometric mean is less than or equal to the arithmetic mean. Now use the identity: $\text{mean}(A + B) = \text{mean}(A) + \text{mean}(B)$. Therefore

$$\begin{aligned} \underset{X}{\text{mean}} (\omega + t/a) &> R/(3 \log a) - 2 \log N \\ &> R/(4 \log a) \end{aligned}$$

provided that $R > 24 \cdot (\log N) \cdot (\log \log N)$.

Now we suppose $W = X_{i,2j}$ and $Y = Z_{i,2j+1}$ for some i,j, so that $\omega = \omega_{i,2j}$, $t = t_{i,2j+1}$ and $R = R_i = 2^i$. Taking the intervals in pairs by summing over j, the previous inequality gives

$$\begin{aligned} \underset{X}{\text{mean}}\ \omega_i + \underset{X}{\text{mean}} \sum_j t_{i,2j+1} / a &= \sum_j \underset{X}{\text{mean}} (\omega_{i,2j} + t_{i,2j+1} / a) \\ &> \sum_j R_i/(4 \log a) = N/(8 \log a). \end{aligned}$$

If we assume that $\underset{X}{\text{mean}}\ T < N \log N/(16 \log\log N)$, then since $\sum_j t_{i,2j+1} < T$,

we have $\underset{X}{\text{mean}}\ \omega_i > N/(8 \log a) - N/(16 \log\log N)$

$= N/(16 \log\log N)$.

Since this inequality holds for all i such that $i < \log N$ and

$2^i = R_i > 24 \cdot (\log N) \cdot (\log\log N)$, we conclude that

$$\underset{X}{\text{mean}}\ T > \underset{X}{\text{mean}}\ \Omega = \sum_i \underset{X}{\text{mean}}\ \omega_i$$

$$\geq [\log N - \log(24(\log N)(\log\log N))] \cdot N/(16 \log\log N)$$

$$\geq N \cdot \log N/(17 \log\log N)$$

provided N is sufficiently large. Thus, we have proved case (i).

The proof for case (ii) is somewhat simpler. We can easily show that if $\omega + 2C \cdot \log T \leq R/2$ then $T^C \cdot 2^\omega < 2^{3R/4}$. From this we deduce in a similar way to case (i) that

$$\underset{X}{\text{mean}}\ (\omega + 2C \cdot \log T) > R/3.$$

If $\underset{X}{\text{mean}}\ T < N^2$ and $R_i > 48 \cdot C \cdot \log N$ then $\underset{X}{\text{mean}}\ \omega_{i,2j} > R_i/4$ and $\underset{X}{\text{mean}}\ \omega_i > N/8$,

so

$$\underset{X}{\text{mean}}\ T > \underset{X}{\text{mean}}\ \Omega = \sum_i \underset{X}{\text{mean}}\ \omega_i > N \cdot (\log N)/9.$$

This proves case(ii). Of course a proof solely for oblivious machines would be very easy since ω_{ij}, t_{ij}, T, etc. are independent of X. □

We can immediately extend this proof, removing the dependence on K_N. We show that nearly all numbers as multipliers yield a function nearly as retrorse as multiplication by K_N. In the situation of Figure 1 and Lemma 1, let us replace K_N by an arbitrary N-digit number K_N^*.

Lemma 5. For any h, $0 < h < 2^R$, if for some i, $Y_i = Y_{i+h}$, then K_{2R}^* must have one of at most 2^{R+1} possible values.

Proof. As in the proof of Lemma 1, if $Y_i = Y_{i+h}$, then

$$h \cdot K_{2R}^* \equiv a \pmod{2^{2R}}$$

for some a, $|a| < 2^R$.

Let $d = \gcd(h, 2^{2R})$. Then $d|a$, so $a = kd$, where $|kd| < 2^R$. Also, $d|2^{R-1}$ by definition of d and the fact that $h < 2^R$. Hence,

$$k \in \left\{-\frac{2^R}{d}+1, -\frac{2^R}{d}+2, \ldots, -1, 0, 1, \ldots, \frac{2^R}{d}-1\right\}$$

so there are $(2 \cdot 2^R/d) - 1$ such k's.

By elementary number theory, there are exactly d values of K_{2R}^* in the range $0 \le K_{2R}^* < 2^{2R}$ which satisfy

$$hK_{2R}^* \equiv kd \pmod{2^{2R}}.$$

Hence, there are at most

$$d \cdot \left(\frac{2 \cdot 2^R}{d} - 1\right) < 2^{R+1}$$

values of K_{2R}^* for which $Y_i = Y_{i+h}$. □

From this lemma, it follows at once that at least half of all possible values of K^*_{2R} have the property that for all h, $0 < h \leq 2^{R-2}$, and for all i, $Y_i \neq Y_{i+h}$. Hence, for these values of K^*_{2R}, at most 4 different W's yield the same Y. Therefore the proof of Theorem 1 can be followed very closely except that "mean" is replaced throughout by "mean mean". Thus we have

$$\overset{X}{\qquad} \qquad\qquad \overset{K^*_N \quad X}{\qquad}$$

shown:

Theorem 2. There is a constant c such that for any BAM $\mathfrak{M}$ which performs on-line multiplication, the mean time, T(N), for pairs of N-digit numbers satisfies the following bounds for all sufficiently large N.

(i) If $\mathfrak{M}$ is polynomial-limited or uniform,

$$T(N) > c\ N \log N/\log \log N;$$

(ii) If $\mathfrak{M}$ is a multitape Turing machine or is oblivious,

$$T(N) > c\ N \log N.$$

We know of no direct implication between Theorems 1 and 2.

7. EXTENSIONS

In this section we shall outline some of the ways in which the classes already considered can be extended while remaining susceptible to the same proof methods.

A simple "random-access" machine could be modelled by a BAM with a storage structure based upon some sort of binary tree, so that location are "addressed" by binary sequences and accessed in time

proportional to the address length. Such a structure is of course exponentially, rather than polynomially, limited. However we recall that the latter property is used in the proof only to allow head positions to be specified just before a new input symbol is to be read. The proof goes through just as before therefore, provided that the tree structure is used in such a way that the heads are returned to the root before each input is read, for example if all the "random-accessing" is accomplished by a subroutine.

An alternative approach to a random-access store is the structure based on the free group on two generators, a,b, with the four shifts being left multiplication by a, a^{-1}, b, b^{-1}. This can be operated as a quite serviceable random-access store and is of course uniform.

A point of merely technical interest is that the same bounds may be easily proved when the polynomial limited class is extended by replacing "ct^d" in the definition by "$c2^{t^\epsilon}$" for any $\epsilon < 1$. Unfortunately we know of no natural class of machines which takes advantage of this extension.

Finally we show that without impairing the proof for any of the four classes of machines we may add "oracles", and indeed more, in the following way. The BAMs are extended by allowing an infinite number of states in the control subject only to the restriction that just a finite number of them may read the input tape. An "oracle", which may even be non-recursive, could be invoked with such a machine to read a sequence

of storage locations and put the result of applying its oracular function in some other sequence of locations. This would take just the number of steps required to access the locations. The proofs are unaffected by this relaxation. A simple example which emphasizes the importance to our proof of the on-line restriction is a multitape Turing machine with an oracle to perform (off-line) multiplication in linear time. The $cN \cdot \log N$ lower bound applies even to this machine.

8. UPPER BOUNDS

An important technique for establishing upper bounds for on-line multiplication is given by M. Fischer and Stockmeyer [4]. Their construction shows that, for a wide range of machine classes including multitape Turing machines, oracle Turing machines, and oblivious machines, given any off-line (i.e. unrestricted) multiplication machine with time complexity $T(N)$, where T satisfies $T(2N) \geq 2T(N)$, an on-line machine can be produced with time complexity no greater than $c \cdot T(N) \cdot \log N$.

A slight extension of their methods shows that on-line multiplication of N-digit integers, where one of the numbers has at most $\log N$ "1"-digits, can be performed in time $O(N \cdot \log N)$. In particular, there is a Turing machine for on-line multiplication by K_N with complexity $O(N \cdot \log N)$, matching the bound of Theorem 1 (ii).

General on-line multiplication algorithms may be obtained by applying the Fischer-Stockmeyer result to the off-line algorithm of Schönhage-Strassen [7], which on a Turing machine has complexity $O(N \cdot \log N \cdot \log\log N)$.

With the facility of constructing and rapidly accessing a multiplication table for logN digit numbers such as is provided by a random-access machine, the Schönhage-Strassen off-line multiplication algorithm can be performed in time $O(N \cdot \log N)$.

In Figure 3 we set out some of the upper bounds derived from the above results for three classes of machines. Constant factors are omitted and underlining denotes that a lower bound of the same order has been demonstrated in previous sections. The first class is multitape Turing machines with one-dimensional tapes; the second is BAMs with an infinite number of states under the restriction on input states given in Section 7; the third class is either version of "random-access" machine described in Section 7. With the uniform structure based on the free group on two generators, it is easy to simulate Turing machines and stores with "random-access". BAMs with the binary tree structure and the restriction on head positions given in Section 7 are also sufficiently powerful to allow a fast implementation of the required algorithms, though the programming techniques needed are less straightforward.

9. CONCLUSION

In this paper we have described a powerful counting argument based on the notion of "overlap" and have investigated the extent and limitations of its applicability. Overlap arguments are applicable only under the

on-line restriction, but in many cases they can lead to complexity bounds which are optimal within a constant factor.

An important objective for future research is to obtain nontrivial lower bounds without the severe restriction to on-line computation. Such results, even for oblivious machines or combinational circuits, would constitute a significant advance.

REFERENCES

1. S.O. Aanderaa, On k-tape versus (k+1)-tape real-time computation, this volume.

2. A.J. Atrubin, A one-dimensional real-time iterative multiplier, IEEE Trans. Electronic Computers EC-14 (1965), 394-399.

3. S.A. Cook and S.O. Aanderaa, On the minimum computation time of functions, Trans. Amer. Math. Soc. 142 (1969), 291-314.

4. M.J. Fischer and L.J. Stockmeyer, Fast on-line integer multiplication, Proc. 5th ACM Symposium on Theory of Computing (1973), 67-72.

5. P.C. Fischer, A.R. Meyer and A.L. Rosenberg, Real-time simulation of multihead tape units, Jour. ACM 19 (1972), 590-607.

6. J.E. Hopcroft and J.D. Ullman, Formal Languages and their Relation to Automata, Addison-Wesley, Reading, Mass., 1969.

7. A. Schönhage and V. Strassen, Schnelle Multiplikation grosser Zahlen, Computing 7 (1971), 281-292.

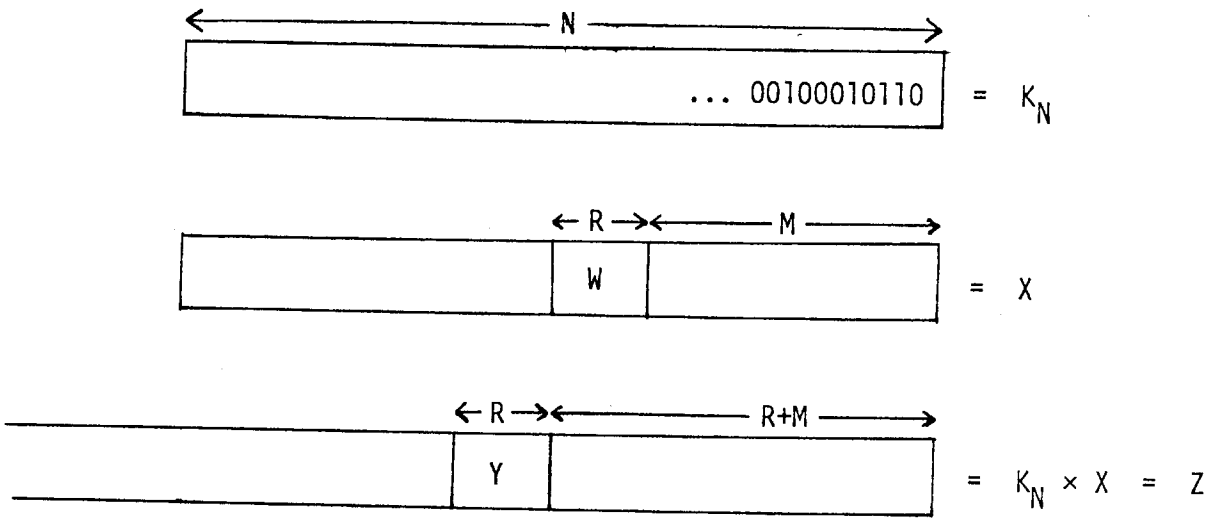


Figure 1.

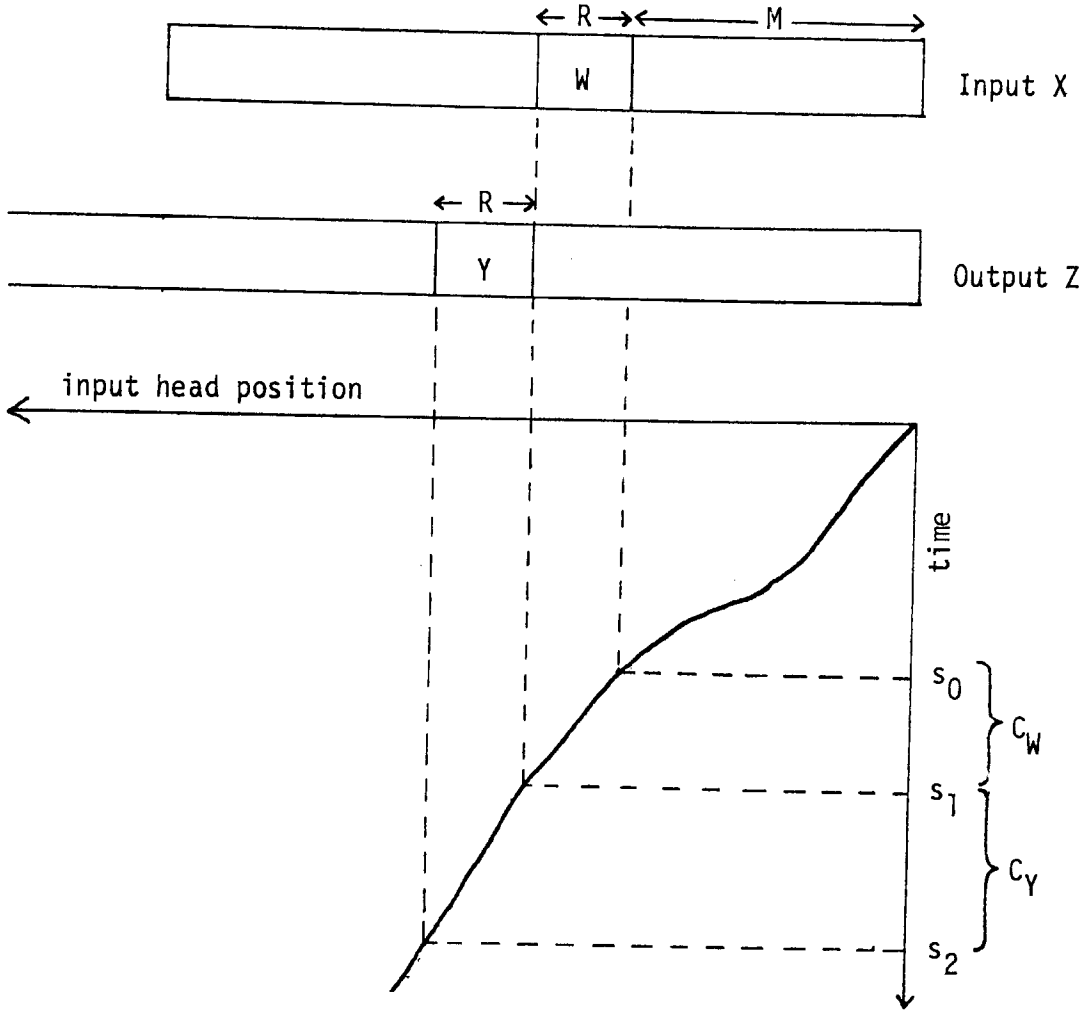
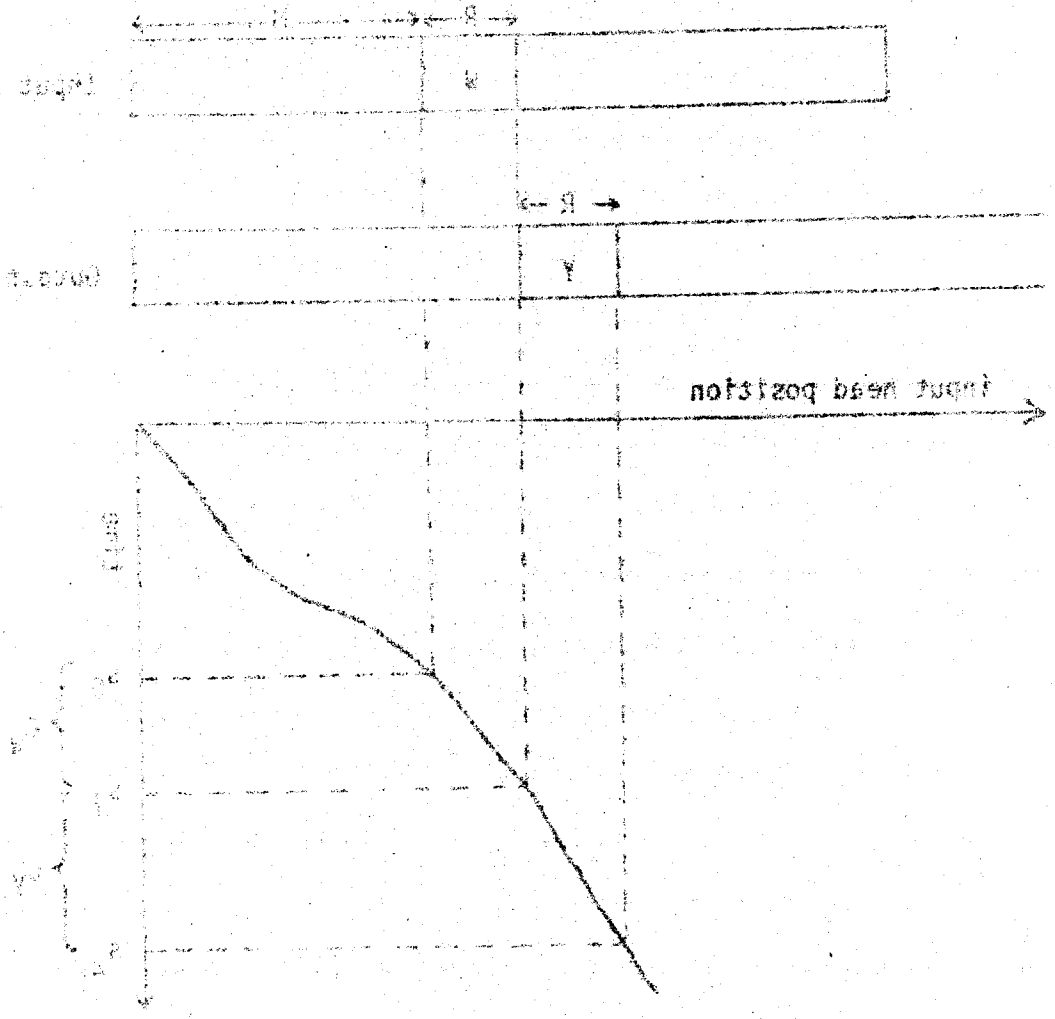


Figure 2.

	Multitape TM	Oracle TM	"RAM"
On-line multiplication	$N \log^2 N \log\log N$	$N \log N$	$N \log^2 N$
On-line mult. by K_N	$N \log N$	$N \log N$	$N \log N$
Off-line multiplication	$N \log N \log\log N$	N	$N \log N$

Figure 3.



BIBLIOGRAPHIC DATA SHEET	1. Report Nos: GJ34671 + N00014-70-A-0362-0003 MAC TM-40	2.	3. Recipient's Accession No.
4. Title and Subtitle An Improved Overlap Argument for On-Line Multiplication			5. Report Date: Issued January 1974 6.
7. Author(s) Michael S. Paterson, Michael J. Fischer + Albert R. Meyer			8. Performing Organization Rept. No. MAC TM-40
9. Performing Organization Name and Address PROJECT MAC; MASSACHUSETTS INSTITUTE OF TECHNOLOGY: 545 Technology Square, Cambridge, Massachusetts 02139			10. Project/Task/Work Unit No. 11. Contract/Grant Nos: GJ34671 + N00014-70-A-0362-0003
12. Sponsoring Organization Name and Address Office of Naval Research Department of the Navy Information Systems Program Arlington, Va 22217 Associate Program Director Office of Computing Activities National Science Foundation Washington, D. C. 20550			13. Type of Report & Period Covered: Interim Scientific Report 14.

15. Supplementary Notes

Much of this work was carried out at the University of Warwick with partial support from the Science Research Council of the U. K.

16. Abstracts

A lower bound of cNlogN is proved for the mean time complexity of an on-line multitape. Turing machine performing the multiplication of N-digit binary integers. For a more general class of machines which includes some models of random-access machines, the corresponding bound is cNlogN/loglogN. These bounds compare favorably with known upper bounds of the form $cN(\log N)^k$, and for some classes the upper and lower bounds coincide. The proofs are based on the "overlap" argument due to Cook and Aanderaa.

17. Key Words and Document Analysis. 17a. Descriptors

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Unlimited Distribution Write Project MAC Publications	UNCLASSIFIED	30
	20. Security Class (This Page) UNCLASSIFIED	22. Price