JOURNAL OF COMPUTER AND SYSTEM SCIENCES: 4, 50–73 (1970)

Time-Restricted Sequence Generation*

PATRICK C. FISCHER†

University of Waterloo, Waterloo, Ontario, Canada

ALBERT R. MEYER‡

AND

ARNOLD L. ROSENBERG

IBM Watson Research Center, Yorktown Heights, New York

Received July 30, 1969

The classes of sequences generated by time- and space- restricted multiple counter machines are compared to the corresponding classes generated by similarly restricted multiple tape Turing machines. Special emphasis is placed on the class of sequences generable by machines which operate in *real time*. Real-time Turing machines are shown to be strictly more powerful than real-time counter machines. A number of questions which remain open for real-time Turing machines are settled for real-time counter machines.

INTRODUCTION

In recent years much work has been done on the complexity of Turing machine computations. Parallelling this work on time- and space-restricted Turing machines, we consider similar restrictions on machines which, in place of tapes, have registers capable of holding integers which can be incremented or decremented by one and tested for zero. Although such counter machines (CM's) appear to be much simpler than Turing machines (TM's), unrestricted CM's have been shown by Minsky [8] to be as powerful as TM's. Counter machines, being susceptible to formal definition using little more mathematics than vector addition, are somewhat more tractable

* A portion of this paper was presented at the Eighth Annual Symposium on Switching and Automata Theory, Austin, Texas, October 18-20, 1967. See Reference [3].

† Research of this author was partially supported by the National Science Foundation, Grant GP-7701, and by the National Research Council of Canada, Grant A-5549.

‡ This author is currently with the Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

than restricted TM's; therefore, a number of questions which are still open for restricted TM's are settled in this paper for restricted CM's.

We confine our attention in this paper to machines as sequence generators. The main theme of the paper is a comparison of the properties of time-restricted TM and CM generators. In an earlier paper [4], we have made a similar comparison of language recognizers. Since sequence generation can be viewed as recognition of a language encoded over a one letter alphabet, a number of the results in this paper strengthen results reported in our earlier work.

Special attention is paid to sequence generators which operate in *real time*, i.e., emit a bit at every step. The work of Hartmanis and Stearns [5] and Fischer [2] indicates that real-time generation plays a fundamental role in the study of time requirements of more general computations. Moreover, a sequence generable within any computable time restriction is, by definition (cf. Section 1.1), a homomorphic image of a real-time generable sequence. This observation yields immediate extensions to a number of results reported in this paper which, for simplicity, have been stated and proved only for real-time generators.

A surprising application of the notion of real-time generability has appeared in proofs by Cobham [1] of some special cases of

CONJECTURE (Hartmanis-Stearns). Let x be a real number, $0 \leqslant x < 1$. If the infinite binary expansion of x is real-time generable by a CM, then x is either rational or transcendental.

This conjecture lends yet more motivation to the study of real-time sequence generation.

1. PRELIMINARIES

1.1. *The Models*

An n counter machine (n-CM) comprises a finite state control and n *counters*, each capable of containing any integer. The control unit of an n-CM is partitioned into *active states* (which emit outputs) and *dormant states* (which do not emit outputs). At the start of its computation, the n-CM is in a designated initial state, and all counters are set to zero. Each step in a CM computation is uniquely determined by the state of the control unit and by that subset of the counters which contain zero. The action at a step consists of adding 0, $+1$, or -1 independently to each counter and changing the state of the control unit. If the state of the CM is active, then an output is also emitted at that step. The sequence generated by the CM is then the sequence of outputs emitted during the course of the computation. Note that CM's operate autonomously. Thus, each CM computes a unique sequence, and each computation is infinite in that no "halting" mechanism is assumed.

More precisely, an n-CM $\mathbf{M}$ is specified by

(1) a finite set Q of *states* which is partitioned into a set Q_a of *active* states and a set Q_d of *dormant* states;

(2) a designated *initial* state $q_0 \in Q$;

(3) a *machine function*[1] M,

$$M : Q \times \{0, 1\}^n \to Q \times \{-1, 0, 1\}^n;$$

(4) an *output function* ω,

$$\omega : Q_a \to \{0, 1\}.$$

Let Z denote the set of integers. For any $z \in Z$, define

$$sg(z) = \begin{cases} 0 & \text{if } z = 0, \\ 1 & \text{otherwise.} \end{cases}$$

Extend the function sg to Z^n by

$$sg(\langle x_1, \ldots, x_n \rangle) = \langle sg(x_1), \ldots, sg(x_n) \rangle.$$

For $\langle x_1, \ldots, x_n \rangle$ and $\langle y_1, \ldots, y_n \rangle$ in Z^n, define

$$\langle x_1, \ldots, x_n \rangle + \langle y_1, \ldots, y_n \rangle = \langle x_1 + y_1, \ldots, x_n + y_n \rangle.$$

A *configuration* of the n-CM $\mathbf{M}$ is an element of $Q \times Z^n$. If $C = (s, \langle x_1, \ldots, x_n \rangle)$ is a configuration of the n-CM, then the state of $\mathbf{M}$ in configuration C is $\sigma(C) = s$. The *successor* of configuration C is the configuration

$$\Sigma(C) = (s', \langle y_1, \ldots, y_n \rangle)$$

where

$$M[s, sg(\langle x_1, \ldots, x_n \rangle)] = (s', \langle z_1, \ldots, z_n \rangle)$$

and

$$\langle y_1, \ldots, y_n \rangle = \langle x_1, \ldots, x_n \rangle + \langle z_1, \ldots, z_n \rangle.$$

The *computation* of the n-CM $\mathbf{M}$ is the (infinite) sequence of configurations $C_0, C_1, C_2, \ldots$ where $C_0 = (q_0, \langle 0, \ldots, 0 \rangle)$, and for each $i > 0$ $C_i = \Sigma(C_{i-1})$. The *state-sequence* generated by $\mathbf{M}$ is the sequence $s_0, s_1, \ldots$ where, for each $i \geq 0$, $s_i = \sigma(C_i)$.

The *sequence generated* by $\mathbf{M}$ is the binary sequence $\alpha = a_0, a_1, a_2, \ldots$ where the a_i are specified as follows:

[1] For any set S, we denote by S^n the set of n tuples of elements of S.

Let $q_0, q_1, \ldots$ be the state-sequence generated by **M**.

(1) Let $t(0) = \mu k[q_k \in Q_a]$;[2]

(2) For each $j > 0$, let

$$t(j) = \mu k[k > t(j-1) \,\&\, q_k \in Q_a].$$

Then, for each $j = 0, 1, 2, \ldots$, $a_j = \omega(q_{t(j)})$.[3]

Let T be any increasing function from the nonnegative integers into the nonnegative integers. We say that **M** operates *within time* T if $t(k) \leqslant T(k)$ for all k. **M** operates *in time* T if $t(k) = T(k)$ for all k.

Similar definitions can be made for Turing machines. We refer the reader to Hartmanis and Stearns [5] for details.

For each such function T and positive integer m, we let $\mathbf{C}_T(m)$ denote the class of sequences generable within time T by m-CM's, and let $\mathbf{C}_T = \bigcup_{m=0}^{\infty} \mathbf{C}_T(m)$. We let $\mathbf{T}_T(m)$ and $\mathbf{T}_T$ denote the corresponding classes for TM's.

Of particular interest is the case $T(n) = n$. Automata which operate in time $T(n) = n$ are said to operate in *real time*, and the sequence emitted by such an automaton is said to be *real-time generable*. Obviously, an automaton which operates in real time has no accessible dormant states.

Two automata are said to be *time-equivalent* if they generate the same sequence and operate in the same time.

1.2. *Preliminary Results*

We state, without proof, a number of results from [4] which will be useful in the sequel.

(a) *Space Complexity*

We say an m-CM (m-TM) *operates in space* S [S being a nondecreasing function from the nonnegative integers to the nonnegative integers] if $S(n)$ is the largest integer, in magnitude, assumed by any counter (respectively, is the number of squares of tape visited) until the n-th element of the generated sequence is emitted.

PROPOSITION 1.1. *Let* **M** *be an* m-CM *which operates in space* S. *Then*:

(i) *There is a constant* c *such that* **M** *operates within time* $T(n) \leqslant cnS(n)^m$.

(ii) *For any real* $\epsilon > 0$, *one can find a time-equivalent* CM **M**′ *which operates in space* $S'(n) \leqslant [S(n)^\epsilon]$.[4]

(iii) *One can find an equivalent* 1-TM *which operates in space* $S'(n) \leqslant \log S(n)$.

[2] μ is the minimalization operator of recursive function theory. For any propositional $P(x)$, $\mu y[P(y)]$ is the least integer n such that $P(n)$, if any exists; it is undefined otherwise.

[3] Note that the sequence α can be null, finite, or infinite. We assume in the sequel that all sequences generated are infinite.

[4] For any real number $x > 0$, $[x]$ denotes the integer part of x.

(b) *Variants of the Model*

It will, on occasion, be useful to consider certain variants of our CM model. We now enumerate some variants which do not affect the operation time of a CM.

PROPOSITION 1.2. *Given an* m-CM *with the ability to alter the contents of each counter independently by any integer between* $+c$ *and* $-c$ *in a single step (for some fixed integer* c*), one can effectively find a time-equivalent (ordinary)* m-CM.

PROPOSITION 1.3. *Given an* m-CM, *one can effectively find a time-equivalent* m-CM *which*

(i) *stores only nonnegative integers in its counters*;

(ii) *alters its counters at most every* c *steps, for some integer* $c > 0$;

(iii) *alters at most one counter per step.*

For convenience we shall generally make use of Proposition 1.3(i) without explicit note.

1.3. *Basic Necessary Conditions*

In this section we develop two basic tools for showing that a sequence is not real-time generable by an m-CM. No comparable tools for TM's are as yet known.

Let $\alpha = a_0, a_1, a_2, \ldots$ be an (infinite) binary sequence. For any integer $k > 0$, a *k slice* of α is a subsequence of the form $a_i, \ldots, a_{i+k-1}$, i.e., a contiguous length k subsequence.

LEMMA 1.1. *Let* α *be a sequence real-time generable by an* m-CM $\mathbf{M}$. *There is a constant* c *such that the number of distinct* k *slices of* α *cannot exceed* $c \cdot k^m$.

Proof. Say that $\mathbf{M}$ has q states. We call two configurations of $\mathbf{M}$ *k equivalent* if they differ only in coordinates where both contain integers exceeding k.

Clearly, if $\mathbf{M}$ is started in either of two k-equivalent configurations, the next k steps of the computation will be indistinguishable in terms of state sequence, hence, in terms of output sequence.

Since the number of k-equivalent configurations of $\mathbf{M}$ cannot exceed $q(k+2)^m$, the lemma readily follows.

As an immediate application of the lemma, we find

COROLLARY. *The sequence of successive integers in binary notation is not in* $\mathbf{C}_n$.

Obviously, the sequence has 2^k distinct k slices for every integer k.

The condition of Lemma 1.1 is not sufficient to insure membership in $\mathbf{C}_n$, even for sequences which are in $\mathbf{T}_n$. The following useful condition was suggested by A. Cobham.

LEMMA 1.2. *Let α be a sequence in $\mathbf{C}_n$. Then there is a binary word w such that α has arbitrarily long slices of the form $ww \cdots w$.*

Proof. The proof is based on the following easily verified fact: Let **M** be an m-CM with the property that, for some fixed integer k, at every step in its computation, some counter of **M** contains an integer not exceeding k. Then **M** can effectively be replaced by a time-equivalent $(m - 1)$-CM.

Therefore, let r be the least integer for which there is an r-CM **M** which real-time generates α. If $r = 0$, we are done. If $r > 0$, then for every integer h, there are configurations attained by **M** in which the contents of all counters exceed h; else **M** could be replaced by a real-time $(r - 1)$-CM. Once such a configuration is attained, **M** must exhibit the ultimately periodic behavior of a finite state machine for the next h steps. Since h was arbitrary, the lemma follows.

The well-known sequence of Thue [11] contains no slice of the form www; hence,

COROLLARY (A. Cobham). *The Thue sequence is not in $\mathbf{C}_n$.*

This latter sequence is of special interest since it is real-time generable by a TM, and the number of distinct k slices is uniformly bounded by a second degree polynomial in k.

2. STABILITY OF COMPLEXITY CLASSES

In this section we consider the stability of the time complexity classes under perturbations of the timing functions, under changes in the number of counters or tapes, and under operations on the sequences.

2.1. *Speedup Theorems*

The first main distinction between CM's and TM's is in the area of speedup. The following result follows from Hartmanis and Stearns [5] in conjunction with Fischer [2] or Meyer, Rosenberg, and Fischer [7].[6]

[5] All logarithms in this paper are to the base 2.

[6] The result does not follow from [5] alone, since our generators can, in a single step, emit at most one binary digit and not a sequence thereof.

THEOREM 2.1. *Let* **M** *be an* m-TM *which operates within time* $T(n)$, *and let* $\epsilon > 0$ *be any real number such that* $\epsilon \cdot T(n) \geqslant n$. *One can effectively find an* $(m + 3)$-TM *which is equivalent to* **M** *and which operates within time* $[\epsilon \cdot T(n)]$. *Thus,* $\mathbf{T}_T(m) \subseteq \mathbf{T}_{[\epsilon T]}(m + 3)$.

It is not known if the three additional tapes are essential.

COROLLARY. *For all* $c > 1$, $\mathbf{T}_{cn} = \mathbf{T}_n$. *That is, "linear time" coincides with real time for* TM's.

This result contrasts with the speedup theorem for CM's.

THEOREM 2.2. *Let* $\epsilon > 0$ *be any real number. Given a* CM **M** *which operates within time* $T(n) = n + E(n)$, *one can find an equivalent* CM **M**′ *which operates within time* $T'(n) = n + [\epsilon E(n)]$.

Thus, $\mathbf{C}_{n+E(n)} = \mathbf{C}_{n+[\epsilon E(n)]}$.

Theorem 2.2 follows immediately from Proposition 1.3(ii).

The following result shows that Theorem 2.2 is best possible.

THEOREM 2.3 (M. J. Fischer[7]). *For any* $\epsilon > 0$, *there is a sequence* α *in* $\mathbf{C}_{(1+\epsilon)n} - \mathbf{C}_n$.

Proof. We exhibit a sequence which satisfies the theorem. The emphasis will be on simplicity of exposition rather than on constructing the "tightest" sequence which works.

(a) *Generation of* α

The sequence α is constructed by emitting, in order, binary representations of the successive integers interspersed with strings of 0's. In particular, the binary representation of each integer m is preceded by a string of 0's of length $2^{[\log m]+1}$.

We describe a 5-CM **M** which generates α. **M** will operate in discrete stages; for each integer m, the m-th stage has 2^m phases.

At the initiation of the 0-th phase of the m-th stage, the five counters of **M** will contain the following integers:

Counter	*Contents*
A	2^{m-1}
B	0
C	0
D	2^{m-1}
E	0

[7] Private communication.

(i) Using counter D, **M** emits 2^m 0's, one per time unit. Simultaneously, **M** loads counters C and E with 2^m. Using finite state control, the roles of C and D are interchanged.

(ii) Using counters A and B, by successive divisions by 2, **M** emits the binary representation of the contents of A. Simultaneously, **M** loads C with the original contents of A. At the end of this process, counters A and B are both 0, and counter C contains the integer that was in A originally. The roles of A and C are interchanged.

(iii) Counter A is incremented by $+1$ and counter E by -1.

The next phase ensues.

At the initiation of the k-th phase of the m-th stage ($1 \leqslant k < 2^m$), the five counters of **M** will contain the following integers:

Counter	*Contents*
A	$2^{m-1} + k$
B	0
C	0
D	2^m
E	$2^m - k$

(iv) Using counters C and D, **M** emits a string of 2^m 0's, one per time unit. The roles of C and D are then interchanged.

Steps (ii) and (iii) are now performed to complete this phase.

Note that the nullity of counter E is used to signal a change of stage.

(b) $\alpha \notin \mathbf{C}_n$

Note that, for every integer m, α contains 2^m distinct m-slices. By Lemma 1.1, α is not a real-time generable by any CM.

(c) $\alpha \in \mathbf{C}_{3n}(5)$

During each of the 2^m phases of the m-th state, **M** emits $m + 2^m$ symbols. The initial string of 0's of length 2^m is emitted in real time. The remaining m symbols are emitted in fewer than 2^{m+1} steps.[8] Each phase thus takes fewer than $3 \cdot 2^m$ steps. To see this more precisely, assume that symbols $a_0, \ldots, a_r$ have been generated in at most $3r$ steps, and that symbol a_{r+1} is the first symbol to be emitted in the k-th

[8] The first division takes fewer than 2^m steps, the second fewer than 2^{m-1}, etc.

phase of the m-th stage. For each $i \in \{1,..., 2^m\}$, symbol a_{r+i} is emitted at step $3r + i < 3(r + i)$. Then, for each $j \in \{1,..., m\}$, symbol a_{r+2m+j} is emitted before step

$$3r + 2^m + \sum_{h=i}^{j} 2^{m-h+1} = 3r + 2^m + 2^{m+1}\left(\sum_{h=1}^{j} 2^{-h}\right) < 3(r + 2^m) < 3(r + 2^m + j).$$

Thus, $\mathbf{M}$ operates within time $T(n) = 3n$. Theorem 2.2, therefore, implies that $\alpha \in \mathbf{C}_{(1+\epsilon)n}$.

Thus, for counter machines, one obtains the same "partial" speedup for generation as for recognition [4]. For Turing machines, Theorem 2.1 indicates a full speedup for generation, whereas Rosenberg [10] showed there is only a partial speedup for recognition.

There is an important special case in which a full speedup is attainable. This result is immediate from Proposition 1.3(ii).

PROPOSITION 2.1. *Given a* CM *which emits an output at least every c steps for some integer* $c > 0$, *one can find an equivalent* CM *which operates in real time.*

2.2. *The Real-Time Hierarchy*

In 1963, Rabin [9] proved that there were languages real-time recognizable by 2-tape TM's but not by 1-tape TM's. For generation, it is not known if any number of tapes yields more real-time computing power than one tape.

Open Problem. Is there an $m > 1$ such that $\mathbf{T}_n(m) - \mathbf{T}_n(1) \neq \emptyset$?

For counter machines, the current state of knowledge is more complete. A simple proof was presented in [4] to the effect that, for every m, there is a language which is real-time recognizable by an $(m + 1)$ — CM but by no m — CM. We now extend this result to CM generators (which can be viewed as recognizers for languages for a unary alphabet).

THEOREM 2.4. *For every integer* $m > 0$, $\mathbf{C}_n(m)$ *is properly included in* $\mathbf{C}_n(m + 1)$.

Proof. We first remark that $\mathbf{C}_n(1) = \mathbf{C}_n(0)$ is the set of sequences generable by a finite state machine. Since each such sequence is ultimately periodic, it is an elementary excercise to find sequences in $\mathbf{C}_n(2) - \mathbf{C}_n(1)$.

The reader will easily verify, for example, that the sequence

$$\alpha = a_0, a_1, ...$$

such that $a_i = 1$ iff i is a perfect square suffices. The difficulty sets in when the m of the theorem strictly exceeds 1.

We now exhibit, for any given integer $m \geqslant 2$, a sequence $\alpha \in \mathbf{C}_n(m+1) - \mathbf{C}_n(m)$. Since weak inclusion of the sets is immediate, this demonstration, combined with the above remarks will complete the proof.

(a) *Generation of* α [9]

Let **M** be a real-time $(m+1)$ — CM with counters labelled $A_0, \ldots, A_m$. **M**'s computation is composed of alternating phases, an *output* phase, and an *update* phase. The initiation of the output phase is signalled by A_0 containing 0 and that of the update phase by A_1 containing 0. (Both cannot occur simultaneously.)

Unless otherwise indicated, **M** emits a 0 at every step.

(i) Initialization: **M** adds $+1$ to A_1 and proceeds to the output phase.

(ii) Output phase: **M** increments A_0 by 1 at each step, while simultaneously decrementing all other counters, repeating until A_1 contains 0, at which point the update phase is entered. **M** emits a 1 at precisely those steps when at least one of the counters contained a 0 on the previous step.

(iii) Update phase: Let j be the largest integer for which A_j does not contain 0. If $j = 0$, set j to 1. **M** first adds $+1$ to the current A_j. Then **M** simultaneously adds -1 to A_0 and $+1$ to $A_1, \ldots, A_j$, repeating until A_0 contains 0. The output phase is then reentered.

(b) $\alpha \in \mathbf{C}_n(m+1)$

By definition, **M** generates α in real time.

(c) $\alpha \notin \mathbf{C}_n(m)$

In order to show that $\alpha \notin \mathbf{C}_n(m)$, we must analyze the structure of α.

(i) Assume that at the beginning of an output phase, the counters of **M** contain integers $0, X_1, \ldots, X_m$, respectively, where $X_1 > 0$, and $X_1 \geqslant X_i$ for $i \in \{2, \ldots, m\}$.

During this phase, **M** emits a slice of α of length $X_1 + 1$ which contains 1's precisely in positions $1, X_1 + 1, \ldots, X_m + 1$.[10] This phase ends with the counters of **M** containing the integers $X_1, 0, X_2 - X_1, \ldots, X_m - X_1$, respectively.

(ii) Now assume that, at the beginning of an update phase, the counters of **M** contain integers $X_1, 0, X_2 - X_1, \ldots, X_m - X_1$, respectively, with some $X_j - X_1 \neq 0$ while $X_k = X_1$ for $k > j$.

During this phase, **M** emits a slice of α of length $X_1 + 1$ which consists entirely of 0's. The update phase ends with the counters of **M** containing the integers

[9] For this proof we revert to the original definition of CM which permits negative integers in the counters.

[10] Of course some X_i may equal some X_j, so some of these $m+1$ positions may be the same.

$0, X_1, X_2, \ldots, X_{j-1}, X_j + 1, 0, \ldots, 0$, respectively. In the case that all $X_i = X_1$ for $i > 1$, it ends with the counters of **M** containing $0, X_1 + 1, 0, \ldots, 0$, respectively.

Note that, in effect the integers $X_2, \ldots, X_m$ are treated as the base $X_1 + 1$ representation of an integer, with X_2 being the high order digit. Thus, with each increment of this integer, carries propogate "to the left." The set of integers of length $m - 1$ in base $X_1 + 1$ is exhausted when $X_1 = X_2 = \cdots = X_m$. At this point, X_1 is incremented, and **M** begins to enumerate the $(m - 1)$-digit numbers in base $X_1 + 2$.

With this information about the structure of α, we can resume our argument.

For each integer $k > 1$, consider the set of (k, m)-*words*, i.e., length $k - 1$ binary words containing at most $(m - 1)$ 1's. For every k, the number of (k, m)-words exceeds the binomial coefficient $\binom{k-1}{m-1}$. Moreover, for every (k, m)-word w, α contains a slice $0^k 1 w 1 0^k$.[11]

Now, let $r > 2m$ be any integer divisible by 4, and let k be an integer in the range $r/2 \leqslant k \leqslant 3r/4$. Each (k, m)-word w appears in $r - k$ distinct r-slices of α; namely,

$$0^i 1 w 1 0^{r-k-1-i} \qquad (i = 0, \ldots, r - k - 1).$$

Thus, the number of distinct r slices of α must exceed

$$\sum_{k=r/2}^{3r/4} (r - k) \binom{k-1}{m-1}$$

which, in turn, exceeds

$$\sum_{k=r/2}^{3r/4} (r - 3r/4) \binom{r/2 - 1}{m - 1}$$

which is no smaller than

$$(r/4)(r/4)(r/2 - m + 1)^{m-1}/(m - 1)!$$

This last expression is a polynomial in r of degree $m + 1$ with positive leading coefficient and, therefore, exceeds $c \cdot r^{m+1}$ for some fixed $c > 0$.

Since, for sufficiently large r, $r^{m+1} > d \cdot r^m$ for any fixed d, the proof is now completed by appealing to Lemma 1.1.

Thus, the addition of a single counter increases the real-time computing power of CM's. Unfortunately, the present proof sheds no light on the corresponding problem for TM's. It would appear that new notions are needed to settle that open problem.

2.3. *Operations on Sequences*

In this section we consider the effect of operations on sequences on the classes $\mathbf{T}_n(m)$ and $\mathbf{C}_n(m)$. Another basic difference between TM's and CM's will emerge from this study.

[11] We let 0^k denote a string of k 0's.

(a) *Compression Operations*

Let f be any function $f: \{0, 1\}^k \to \{0, 1\}$ for some k. A sequence $\beta = b_0, b_1, \ldots$ is an f *compression* of a sequence $\alpha = a_0, a_1, \ldots$ if for each nonnegative integer i, $b_i = f(a_{ik}, a_{ik+1}, \ldots, a_{(i+1)k-1})$.

THEOREM 2.5. *Let α be any sequence in $\mathbf{C}_n(m)$ [respectively, in $\mathbf{T}_n(m)$] for some m. Then every f compression of α is also in $\mathbf{C}_n(m)$ [respectively, in $\mathbf{T}_n(m)$].*

The proof is immediate since one can obviously construct a generator for any f compression of α which would emit symbols at a constant rate.

(b) *Composite Sequences*

As before, let f be an arbitrary function mapping $\{0, 1\}^k \to \{0, 1\}$. The sequence $\beta = b_0, b_1, \ldots$ is an f *composite* of the k sequences $\alpha_i = a_{i0}, a_{i1}, \ldots$ $(1 \leqslant i \leqslant k)$ if, for every integer $j \geqslant 0$, $b_j = f(a_{1j}, a_{2j}, \ldots, a_{kj})$.

The following result is obvious:

PROPOSITION 2.2. *If the k sequences $\alpha_1, \ldots, \alpha_k$ are each in $\mathbf{C}_n(m)$ [respectively, in $\mathbf{T}_n(m)$] for some integer m, then any f composite of the sequences is in $\mathbf{C}_n(km)$ [respectively, in $\mathbf{T}_n(km)$].*

In this subsection we show that for TM's Proposition 2.2 can be significantly improved, while for CM's it cannot.

THEOREM 2.6. *If the k sequences $\alpha_1, \ldots, \alpha_k$ are each in $\mathbf{T}_n(m)$ for some integer m, then any f-composite of the sequences is in $\mathbf{T}_n(m + 4)$.*

Proof. By Theorem 2.4, it will suffice to show that, for some integer $c > 0$, the composite sequence is in $\mathbf{T}_{cn}(m + 1)$. This, then, is our goal.

For $i = 1, \ldots, k$, let the sequence α_i be generated by the real-time m-TM $\mathbf{M}_i$. We construct an $(m + 1)$-TM $\mathbf{M}$ which will generate the f-composite sequence β.

(a) *Structure of* $\mathbf{M}$

$\mathbf{M}$ will have $m + 1$ tapes, denoted $T_0, \ldots, T_m$, each having k tracks.[12] Tape T_0 will be used to control the generation of β. For $i = 1, \ldots, k$, the i-th channels of tapes $T_1, \ldots, T_m$ will be used to simulate $\mathbf{M}_i$.

(b) *Operation of* $\mathbf{M}$

$\mathbf{M}$ will operate in discrete stages, each comprising $k + 1$ phases. Roughly speaking, during the r-th stage, $\mathbf{M}$ will generate 2^r symbols from each α_i, hence, an equal number from β.

[12] The reader who wishes to avoid the anthropomorphic notion of tracks can read this as saying that the symbols written on T_0 are from an alphabet which is a cross product of k smaller alphabets.

Let us assume that at the beginning of the r-th stage, $\mathbf{M}$ has the following information available:

(i) a segment of length 2^{r-1} is marked off on tape T_0;

(ii) for $i = 1,\ldots, k$, the i-th channels of tapes $T_1,\ldots, T_m$ contain the same information as would the tapes of $\mathbf{M}_i$ after $2^r - 1$ steps of its computation, and the squares on which $\mathbf{M}_i$'s heads would reside are marked;

(iii) for $j = 1,\ldots, m$, the head on tape T_j resides on the square it initially occupied, which is also marked. The r-th stage now proceeds as follows:

Phase 0. Using the marked segment of length 2^{r-1}, $\mathbf{M}$ uses tapes T_0 and T_1 to mark off a segment of length 2^r on tape T_0.

Phase i ($1 \leqslant i < k$). $\mathbf{M}$ positions heads $1,\ldots, m$ on the squares marked on channel i of each tape. $\mathbf{M}$ then simulates $\mathbf{M}_i$ for 2^r steps, recording the symbols $\mathbf{M}_i$ would emit on channel i of tape T_0. The end of this simulation period is signalled by the head on T_0 reaching the end of the marked segment of length 2^r. The head on T_0 then backspaces to the beginning of the segment, and the other heads backspace to their initial squares. Phase $i + 1$ is now entered.

Phase k. Phase k is almost identical to phase k $-$ 1. However, instead of recording the symbols $\mathbf{M}_k$ would emit, $\mathbf{M}$ combines these symbols with the stored output from the preceding phases and emits the next 2^r-slice of β.

The reader should be able to fill in the details about $\mathbf{M}$'s operation.

(c) *Timing Estimates*

We estimate the time required for the r-th stage.

(i) Phase 1 clearly requires at most 2^r steps.

(ii) Since, by the end of the $(r - 1)$-th stage, $2^r - 1$ symbols of each α_i have been generated, at most $2^r - 1$ steps are required to position the heads of $\mathbf{M}$ in each phase. Each $\mathbf{M}_i$ is then simulated for 2^r steps. Finally, the heads are backspaced at most 2^{r+1} squares. Thus, the r-th stage, during which a 2^r-slice of β is generated, requires at most $(4k + 1) \cdot 2^r$ steps. Therefore, $\mathbf{M}$ operates within time $T(n) = 8k \cdot n$. (The additional factor of 2 compensates for the fact that phases 0 $-$ (k $-$ 1) produce no output.)

Since β is thus in $\mathbf{T}_{8kn}(m + 1)$, the theorem follows.

The reader will easily verify that Theorems 2.5 and 2.6 and Proposition 2.2 remain valid if the finite function f is replaced by a generalized sequential machine mapping.

In contrast to Theorem 2.6, we find that the CM complexity classes are sensitive to even simple operations on sequences, such as OR-ing.

THEOREM 2.7. *For any integers $k, m > 0$, there exist m sequences $\alpha_1, \ldots, \alpha_m$, each in $\mathbf{C}_n(k+4)$, such that the sequence $\beta = \bigvee_{i=1}^{m} \alpha_i$ is not in $\mathbf{C}_n(km)$.*[13]

Proof. For $i = 1, \ldots, m$, we describe a $(k+4)$-CM $\mathbf{M}_i$ which generates α_i in real time. Let us focus on a specific $\mathbf{M}_i$.

For each integer r, let p_r denote the r-th prime, and, for any integer x, let $\pi_r(x)$ be the exponent of p_r in the prime factorization of x.

(a) *General Description of α_i*

In rough terms, $\mathbf{M}_i$ will generate a sequence of the form $z_1 1 w_1 z_2 1 w_2 \ldots$ where, for each integer q, z_q is a sequence of $2kq$ 0's, and w_q is a binary word of length $q+1$ having 1's precisely in positions $\pi_{(i-1)k+1}(q), \ldots, \pi_{ik}(q)$.

(b) *Generation of α_i*

$\mathbf{M}_i$ will have $k+4$ counters: two process-control counters, P_1 and P_2, two computational counters, C_1 and C_2, and k generating counters, $G_1, \ldots, G_k$.

$\mathbf{M}_i$ will operate in discrete stages, each stage comprising an update phase and an output phase. During the update phase of stage q, $\mathbf{M}_i$ will emit $z_q 1$; during the output phase, it will emit w_q. This is accomplished as follows.

At the initiation of stage q ($q = 1, 2, \ldots$) counters P_1 and C_1 will each contain q, and all other counters will be empty.

(i) *Update Phase.* Throughout the update phase, $\mathbf{M}_i$ emits a 0 at each step. At every $(2k)$-th step of the update phase, $\mathbf{M}_i$ adds -1 to P_1 and $+1$ to P_2. The phase terminates when P_1 is empty; then the output phase begins, and $\mathbf{M}_i$ emits a 1.

While $\mathbf{M}_i$ is counting in its control counters, it stores the integer $\pi_j(q)$ in counter G_j for $j = (i-1)k+1, \ldots, ik$. This is accomplished by successive divisions of q by $p_{(i-1)k+1}, \ldots, p_{ik}$, using counters C_1 and C_2. Since computing $\pi_j(q)$ requires fewer than $p_j q/(p_j - 1) \leq 2q$ steps, all this computing can be done in fewer than $2kq$ steps; hence, it can be completed during the update phase.

(ii) *Output Phase.* When the output phase begins, counter P_1 is empty, and P_2 contains q; one of C_1 and C_2 is empty (say C_1; by a change of name this can be assumed), and the other contains an integer less than q; and each G_j contains the integer $\pi_j(q)$.

At each step of the output phase: $\mathbf{M}_i$ adds $+1$ to P_1 and C_1 and adds -1 to P_2; it adds -1 to C_2 until C_2 is empty; it adds -1 to each G_j until that G_j is empty.

At each step an output is emitted. This output is a 0 unless some G_j had become empty at the previous step, in which case a 1 is emitted. When P_2 is empty, $\mathbf{M}_i$ adds $+1$ to P_1 and to C_1, emits a 0, and enters stage $q+1$.

[13] β is the elementwise OR of the α_i.

(c) *The Sequence β*

For any integer s, and for any s tuple of integers $\langle y_1, \ldots, y_s \rangle$, there is obviously an integer x such that, for $i = 1, \ldots, s$, $\pi_i(x) = y_i$. Therefore, the sequence β is of the form $z_1 1 w_1 z_2 1 w_2, \ldots$ where each z_q is a sequence of $2kq$ 0's and each w_q is a binary word of length $q + 1$ having 1's precisely in positions $\pi_1(q), \ldots, \pi_{km}(q)$.

The sequence β must, then, contain every slice of the form $0^q 1 w 1 0^q$ where w is a binary word of length not exceeding q containing at most $km - 1$ 1's. The Theorem now follows by repeating the argument of Theorem 2.4.

By clever programming the constant 4 of the Theorem can undoubtedly be reduced, but it can probably not be set to 0.

3. Real-Time Countable Functions

The notion of a real-time countable function was introduced by Yamada [12] as a convenient way to describe real-time generable sequences. We shall extend the work of Yamada both by generalizing a number of his results to CM generators as well as TM's, and by developing a general technique for finding and identifying real-time countable functions.

One can associate with any sequence $\alpha = a_0, a_1, \ldots$ containing infinitely many 1's a strictly increasing recursive function f as follows:

$$f(0) = \mu y[a_y = 1]$$

$$f(x + 1) = \mu y[y > f(x) \,\&\, a_y = 1].$$

Thus, α is a *characteristic sequence* for the range of f. If α is real-time generable, then f is termed *real-time countable*, and the real-time generator for α is called a *counter for f*.

It is essential to note that a counter for f does not *compute* f in the conventional sense of the word. We shall say that an n-CM computes a function f under the following conditions. If the n-CM is started with an integer x in counter 1 and all other counters empty, it will enter a designated state with $f(x)$ in counter n and all other counters empty. The n-CM is said to compute f within (respectively, in) time T if, for any x, at most (respectively, exactly) $T(x)$ steps elapse between the time the n-CM is started with x in counter 1 to the time it enters the designated state. Similar conventions apply when the counter for f is a TM.

Thus, real-time countable functions should not be confused with functions which can be computed within slowly growing time functions. In fact, any recursive function can be majorized by some real-time countable function, so there are arbitrarily large real-time countable functions which must take arbitrarily long to compute. [The time required to compute $f(x)$ cannot, by definition, be less than $f(x)$.] Rather,

real-time countability is related to the property that a function can be computed in time proportional to its own values. We now formalize this property.

A function $f(x) \geqslant x$ is *self-computable* if there is an integer $k > 0$ such that, for all x, $f(x)$ is computable within time $k \cdot f(x)$.

The definition of real-time countability readily yields

PROPOSITION 3.1. *Every real-time* CM (TM) *countable function is self-computable on a* CM (*respectively*, TM).

The remainder of this section is devoted to proving a partial converse to Proposition 3.1. This converse will provide a powerful tool for establishing the real-time countability of functions. It can be used to simplify several known proofs and settle certain open questions.

We now lay the groundwork for a partial converse to Proposition 3.1.

We say a function $f(x) \geqslant x$ is *strongly self-computable* if there is an integer $k > 0$ such that, for all x, $f(x)$ is computable *in* time $k \cdot f(x)$.

The following lemma is of central importance in the sequel.

LEMMA 3.1. *A function $f(x) \geqslant x$ is self-computable on a* CM (TM) *if, and only if, it is strongly self-computable on a* CM (*respectively*, TM).

Proof. Sufficiency being self-evident, we establish necessity. Let **M** be an n-CM which computes f within time $k \cdot f(x)$ for some integer $k > 0$. Consider the $(n + 2)$-CM **N** which operates as follows. A computation by **N** begins with an integer x in counter 1 and all other counters empty.

(a) **N** simulates **M** on its first n counters, adding 1 to counter $n + 1$ for each step of the computation. After some number $t \leqslant k \cdot f(x)$ steps, when **M** would halt, **N** has the following configuration:

(i) counters $1,\ldots, (n - 1)$ are empty;

(ii) counter n contains $f(x)$;

(iii) counter $n + 1$ contains $t \leqslant k \cdot f(x)$;

(iv) counter $n + 2$ is empty.

(b) **N** now begins decrementing counter n, while handling counters $n + 1$, $n + 2$ in the following way. For each decrement from counter n, a 1 is added to counter $n + 2$. Similarly, for each such decrement, k is subtracted from counter $n + 1$ until that counter is emptied. Once counter $n + 1$ has been depleted, subtractions from that counter are replaced by additions to it. (Thus, we are acting as though counter $n + 1$ had contained $-t$ and we had consistently added k to it.) At the end of this process, a total of $t + k \cdot f(x)$ steps have elapsed, and **N** has the following configuration:

(i) counters 1,..., n are empty;

(ii) counter $n + 1$ contains $k \cdot f(x) - t$;

(iii) counter $n + 2$ contains $f(x)$.

(c) **N** finally empties counter $n + 1$ and halts in a designated state. Thus, after precisely $k \cdot f(x) + t + k \cdot f(x) - t = 2k \cdot f(x)$ steps, **N** has arrived at the following configuration:

(i) counters 1,..., $n + 1$ are empty;

(ii) counter $n + 2$ contains $f(x)$.

f is thus shown to be strongly self-computable. Since a TM tape can readily act as a counter with no time loss, if **M** is an n-TM, **N** can be taken to be an $(n + 2)$-TM. The lemma thus holds for TM's as well.

The following lemma is immediate by Proposition 1.3(ii) and the proof techniques of Theorem 2.2.

LEMMA 3.2. *Let* **M** *be an* m-CM *which computes a function* f *in time* $k \cdot f$ *for some integer* $k > 0$. *For any integer* $c > 0$, *one can effectively obtain from* **M** *an* m-CM **N** *which operates as follows*:

For any integer $x \geqslant 0$, *when* **N** *is started with* $[x/c]$ *in its first counter, all other counters empty, and* $x - c[x/c]$ *in finite state memory,* **N** *will enter a designated state in* $[k \cdot f(x)/c]$ *steps with* $[f(x)/c]$ *in its last counter, all other counters empty, and* $f(x) - c[f(x)/c]$ *in finite state memory.*[14]

We refer to the function which **N** computes as $f^{(c)}$.

Given a function f, let $\Delta f(x) = f(x + 1) - f(x)$. If f is increasing, Δf obviously takes on only positive values.

THEOREM 3.1. *Let* f *be a strictly increasing function. If* Δf *is self-computable on a* CM (TM) *then* f *is real-time countable on a* CM (*respectively*, TM).

Proof. The TM version follows directly from the CM construction given. Let **M** be an m-CM which computes Δf in time $k \cdot \Delta f$ for some integer $k > 0$. By Lemma 3.1, such an **M** exists; moreover, by Lemma 3.2 we can obtain, from **M**, another m-CM **N** which computes the function $\Delta f^{(2k)}$. We now construct an $(m + 2)$-CM **P** which is a real-time counter for f.

(a) *Structure of* **P**

P will have two counters P_1 and P_2 to control the process of counting f, and it will have m counters C_1,..., C_m with which to simulate **N**. It will further have two bounded registers in finite state memory, which can hold any integer not exceeding $2k$.

[14] For any real number x, $\lceil x \rceil$ denotes the least integer $n > x$.

(b) *Operation of* **P**

P will operate in discrete stages. For each integer q, at the beginning of stage q, **P** will have the integer $[q/2k]$ in counters P_1 and C_1, 0 in all other counters, and $q - 2k[q/2k]$ in finite memory.

While emitting 0's at every step, **P** simulates **N** on counters $C_1, \ldots, C_m$ until **N** would complete its computation. At this point, P_1 and P_2 are unchanged, $C_1, \ldots, C_{m-1}$ are empty, C_m contains $[\Delta f(q)/2k]$, and $[k \cdot \Delta f(q)/2k] = \lceil \Delta f(q)/2 \rceil$ steps have elapsed.

P now empties C_m, subtracting 1 every k time units, and simultaneously transfers the contents of P_1 to both P_2 and C_1. The reader will readily see that, when C_m is finally emptied, **P** can emit a 1 after an additional delay of fewer than k steps and make the necessary adjustments for the next stage of the process.

We leave it to the reader to fill in the minor details of **P**'s operation.

Thus, the q-th stage of **P**'s computation takes precisely $\Delta f(q)$ steps. Therefore, inductively, if **P** emitted a 1 at time $f(q)$, it would again emit a 1 at time $f(q) + \Delta f(q) = f(q + 1)$. That is to say, **P** is a real-time counter for f.

Theorem 3.1 and Proposition 3.1 immediately yield

COROLLARY. *Let f be an increasing function. If Δf is real-time countable, then f is also.*

A second corollary of Theorem 3.1 and Lemma 3.1 has wide application.

COROLLARY. *Let f be an increasing self-computable function such that, for some real number $a \geqslant 2$ and for all integers x, $f(x + 1) \geqslant a \cdot f(x)$. Then f is real-time countable.*

Proof. By hypothesis, $f(x)$ is computable within time $k \cdot f(x)$ for some integer $k > 0$. Thus, the obvious method of computing $\Delta f(x)$, namely computing $f(x)$, then $f(x + 1)$, then computing their difference, requires at most $3k \cdot f(x + 1)$ steps.

Now, since $f(x + 1) \geqslant a \cdot f(x)$, it is clear that

$$\Delta f(x) = f(x + 1) - f(x) \geqslant f(x + 1) - f(x + 1)/a = ((a - 1)/a) f(x + 1),$$

whence, $f(x + 1) \leqslant (a/(a - 1))\, \Delta f(x)$.

Therefore, we have (i) $\Delta f(x)$ is computable within time $\lceil (3ka/(a - 1)) \rceil\, \Delta f(x)$, and (ii) $\Delta f(x) \geqslant (a - 1) f(x) \geqslant f(x) \geqslant x$. Thus, Δf is self-computable, and the result follows by Lemma 3.1 and Theorem 3.1.

In order to gauge the utility of the previous results, we note that many familiar arithmetic functions are self-computable (and, in fact, real-time countable on a CM). For example, all polynomials with nonnegative integer coefficients are real-time countable on a CM, as are $f(x) = c^x$ for any positive integer c and $g(x) = x!$. With such common functions as starting points, one can infer the self-computability and/or real-time countability of other functions from the folloming closure properties:

PROPOSITION 3.2. *If f and g are self-computable on a* CM (TM), *then so are the following functions*:

(a) $h_1(x) = f(x) + g(x)$;

(b) $h_2(x) = f(x) \cdot g(x)$;

(c) $h_3(x) = f(x)^{g(x)}$;

(d) $h_4(x) = f(x)!$;

(e) $h_5(x) = f(g(x))$;

(f) $h_6(x) = \sum_{i=0}^{x} f(i)$;

(g) $h_7(x) = \prod_{i=0}^{x} f(i)$.

Proof. It is a straightforward exercise to show that each of the noted functions is computable within time proportional to its values. For example, to compute $f(g(x))$:

(i) Compute $g(x)$ in $k_1 g(x)$ steps.

(ii) Compute $f(g(x))$ in $k_2 f(g(x))$ steps.

This computation thus takes $k_2 f(g(x)) + k_1 g(x) \leqslant (k_1 + k_2) f(g(x))$ steps, whence h_5 is self-computable.

The other proofs are similar and are left to the reader.

The techniques developed in this section afford powerful tools for establishing the real-time countability of functions. In fact, they suffice to show that every function explicitly shown by Yamada to be real-time countable on a TM is, in fact, real-time countable on a CM. However, not all functions which are real-time countable in Yamada's sense (i.e., on a TM) are real-time countable on a CM as the corollaries to Lemma 1.1 and 1.2 readily show.

As a final remark, we note that Lemma 3.2 has a rather strong implication for CM speedup. The lemma can be expanded on to show that, for sequences α in which 1's are sufficiently sparse,[15] one can obtain a full speedup for CM's. This supplements the special case of Proposition 2.1.

4. SIMULATION OF ONE MACHINE BY ANOTHER

4.1. *Simulation of* CM's *by* TM's

The following result was proved by the authors in [4].

[15] To make the notion, "sufficiently sparse," more precise, let $d_\alpha(i, j)$ be the distance between the ith and jth 1's in the sequence α. We then say that α is "sufficiently sparse" if, for some $\epsilon > 0$ and for all n, $d_\alpha(n, n+1) \geqslant (1+\epsilon) d_\alpha(n-1, n)$.

THEOREM 4.1. *Given any* k-CM *recognizer which operates within time* $T(n)$ *and space* $S(n)$, *one can effectively find, for any* $\epsilon > 0$, *an equivalent* 1-TM *recognizer which operates within time* $(1 + \epsilon)\, T(n)$ *and space* $\log S(n)$.

The "full" speedup for TM generators (Theorem 2.1) affords us the following strengthened version of Theorem 4.1.

THEOREM 4.2. *For any integer* k *and real-time countable function* T, $\mathbf{C}_{T(n)}(k) \subseteq \mathbf{T}_{T(n)}(4)$.

In other words, every k-CM can effectively be replaced by a time-equivalent 4-TM. It is an interesting open problem whether or not the number of tapes can be reduced, e.g., to one.

4.2. *Simulation of* TM's *by* CM's

We begin by presenting an upper bound on the time required by an m-CM to simulate a k-TM.

THEOREM 4.3. *Let* $\mathbf{M}$ *be a* k-TM *with* w *working symbols which operates in time* T *and space* S. *There are constants* $u > 1$ *and* v *such that for any integer* $r > 0$, *one can effectively find a* $(2r + 1)k$-CM $\mathbf{N}$, *equivalent to* $\mathbf{M}$ *which operates within time*

$$\begin{aligned} T'(n) &\leqslant v \cdot \sum_{i=1}^{n} [T(i) - T(i-1)] \cdot w^{\lceil S(i)/r \rceil} \\ &< v \cdot T(n)\, w^{\lceil S(n)/r \rceil} \\ &< u^{\lfloor T(n)/r \rfloor}. \end{aligned}$$

Proof. We prove the result for the case $k = 1$, the extension to arbitrary k being immediate.

(a) *Structure of* $\mathbf{N}$

$\mathbf{N}$ will have $2r + 1$ counters named $R_0, \ldots, R_{r-1}$ (these will simulate the right side of $\mathbf{M}$'s tape), $L_0, \ldots, L_{r-1}$ (which will simulate the left side of $\mathbf{M}$'s tape), and A (an auxiliary counter). In addition, the finite memory of $\mathbf{N}$ will have a bounded register, capable of holding any nonnegative integer less than w.

(b) *Correspondence Between Configurations*

Let $\mathbf{M}$ have working alphabet $\{S_0, \ldots, S_{w-1}\}$, S_0 being the blank symbol. Associate with each symbol S_i $(0 \leqslant i < w)$ the integer label $| S_i | = i$.

Assume that, at some state in its computation, the nonblank portion of $\mathbf{M}$'s tape is of the form

$$a_p a_{p-1} \cdots a_0 c b_0 b_1 \cdots b_q$$

with the read-write head residing on c. At the corresponding point in the simulation, counter R_d $(0 \leqslant d < r)$ of **M** will contain the integer

$$x_d = \sum_{i=0}^{\lceil q/r \rceil} | b_{ir+d} | w^i,$$

counter L_e $(0 \leqslant e < r)$ of **M** will contain the integer

$$y_e = \sum_{i=0}^{\lceil p/r \rceil} | a_{ir+e} | w^i,$$

counter A will contain 0, and the bounded register will contain $| c |$.

(c) *Simulation of* **M** *by* **N**

Assume now that **M** overwrites symbol c by c' and shifts its head to the left to arrive at the configuration

$$a_p a_{p-1} \cdots a_0 c' b_0 b_1 \cdots b_q$$

with the read-write head residing on a_0. **N** responds with the following actions:

(i) Using finite state control, **N** renames each counter R_d as R_{d+1} $(0 \leqslant d < r - 1)$. It then renames the original counter R_{r-1} as R_0 and places the integer

$$x = x_{r-1} \cdot w + c'$$

into the new R_0 (using A as an auxiliary counter).

(ii) Similarly, **N** renames counter L_e as L_{e-1} $(1 \leqslant e \leqslant r - 1)$. It then renames the original counter L_0 as L_{r-1} and places the integer

$$y = [y_{r-1}/w]$$

into the new L_{r-1} (again using A).

(iii) Finally **N** replaces the contents of the bounded register by

$$y_{r-1} - w[y_{r-1}/w] = | a_0 |.$$

(d) *Timing Estimates*

One easily verifies that **N**'s simulation of a step of **M** above requires at most $2w(w^{\lceil (p+1)/r \rceil} + w^{\lceil (q+1)/r \rceil})$ steps.

Now, if **M** operates in space $S(n)$, then, for the $T(i) - T(i-1)$ steps between the $(i-1)$-th and i-th outputs, we must have $p + q + 3 \leqslant S(i)$. The simulation by **N** of a step by **M** in this time interval requires, therefore, at most $4w^{\lceil S(i)/r+2 \rceil}$ steps. The result now follows immediately.

A commonly occurring special case yields an interesting corollary.

COROLLARY. *Let* **M** *be a* k-TM *with* w *working symbols which operates in time* $T(n)$ *and space* $S(n) = \log T(n)$. *One can find, for any integer* $q > 0$, *a* $(2q + 1)k$-CM *equivalent to* **M** *which operates within time* $T'(n) = v \cdot \lceil T(n)^{1+(\log w)/q} \rceil$ *for some* $v > 0$.

This corollary and the sequel lend import to the following:

Open Problem. Given a TM which operates in time $T(n)$, does there exist a time-equivalent TM which operates in space $S(n) = \log T(n)$?

We now establish a lemma which yields a lower bound on the time required by a CM to simulate a TM. If the above open problem is settled in the affirmative, then the lower bound obtained is not bad; otherwise, it is quite weak.

LEMMA 4.1. *Let* α *be an infinite binary sequence, and let* f *and* g *be functions with the following property: For infinitely many* $n > 0$, *the initial* $f(n)$ *slice of* α *contains at least* $g(n)$ *distinct* n *slices. If* α *is generable within time* T *by an* m-CM, *then*

$$\sup_{n\to\infty} \frac{nT(f(n))}{g(n)^{1+1/m}} > 0.$$

Proof. Let **M** be an m-CM with q states which generates α within time T. Let $w_i\,[i = 1,..., g(n)]$ be distinct n slices in the initial $f(n)$ slice of α, and let $t_i\,[i = 1,..., g(n)]$ be the time at which the first digit of w_i appears in **M**'s generation of α. Since the w_i are all distinct, the configurations of **M** at times $t_1, ..., t_{g(n)}$ must also be distinct. There must, therefore, be a *least* integer $d > 0$ such that more than half of these $g(n)$ distinct configurations are mutually d inequivalent.

Since the number of d inequivalent configurations of **M** cannot exceed $q \cdot d^m$, we must have

$$q \cdot d^m > g(n)/2$$

whence

$$d > k \cdot g(n)^{1/m},$$

where $k = (2q)^{-1/m}$.

For $i = 1,..., g(n)$, let u_i be the time at which **M** emits the last digit of w_i, and refer to $u_i - t_i$ as the *time required for* w_i. Note that, since the w_i are all distinct, if **M** is in e-equivalent configurations (for some $e > 0$) at times t_i and $t_j \neq t_i$, then the time required for each of w_i and w_j must exceed e.

Now, by definition of d, at least half of the configurations of **M** at times $t_1, ..., t_{g(n)}$ are $(d - 1)$-equivalent to some other of these configurations. Thus, no fewer than half of the n slices $w_1, ..., w_{g(n)}$ each requires at least time d. Of these $g(n)/2$ slices requiring no less than time d, at least $g(n)/2n$ do not overlap in α. (w_i and w_j do not overlap if either $u_i < t_j$ or $u_j < t_i$.) These nonoverlapping slices, therefore, account

for at least $d \cdot g(n)/2n$ steps during **M**'s emitting the initial $f(n)$ slice of α. We must then have

$$T(f(n)) \geqslant d \cdot g(n)/2n > (k/n) \cdot g(n)^{1+1/m}$$

where $k = (2^{m+1}q)^{-1/m}$, and the lemma follows. Q.E.D.

THEOREM 4.4. *The sequence of binary representations of integers is real-time generable by a* 1-TM *but requires time T on any m-*CM *where*

$$\sup_{n\to\infty} \frac{\log nT(n)}{n^{1+1/m}} > 0.$$

The sequence referred to obviously satisfies the lemma with $f(n) = 2^n$ and $g(n) = 2^n/3$.

As a corollary to Theorem 4.4, we obtain the following result which was already obvious from the tools of Section 1.3.

COROLLARY 4.4. $\mathbf{T}_n(1) - \mathbf{C}_n \neq \varnothing$.

4.3. $(m + 1)$-CM's *and* m-CM's

In this final section we investigate the time required for an m-CM to simulate a CM with more counters. Corresponding problems for TM's are discussed by Hartmanis and Stearns [5] and by Hennie and Stearns [6].

By our previous remark that 0-CM's and 1-CM's are both equivalent to finite state machines (as sequence generators), we need consider only machines with $m \geqslant 2$.

The following theorem gives a coarse upper bound, which regrettably does not depend upon the number of counters of the machine **M** being simulated.

THEOREM 4.5. *Given a* CM **M** *which operates within time $T(n)$, one can effectively find an r-*CM *($r \geqslant 2$) which operates within time $T'(n) \leqslant cT(n)^{1+d/(r-1)}$ for some positive constants c and d.*

Proof. (a) Since **M** operates within time $T(n)$, it also operates within space $T(n)$.

(b) Using Theorem 4.1, we can find a 1-TM **N**, equivalent to **M**, which operates within time $T(n)$ and space $\log T(n)$.

(c) Using the corollary to Theorem 4.3, we can find, for any $q > 0$, a $(2q + 1)$-CM, equivalent to **N**, which operates within time $c[T(n)]^{1+(\log w)/q}$ where w is the number of symbols in the alphabet of **N** and c is a constant. Choosing $r = 2q + 1$, and $d = 2 \cdot \log w$ we obtain the stated result.

Other simulation results were reported in [4].

Using Lemma 4.1, we can find a lower bound on the time required by an r-CM to simulate an m-CM.

THEOREM 4.6. *The sequence* α_n *which was shown to be in* $\mathbf{C}_n(m+1) - \mathbf{C}_n(m)$ *requires time* T *on any* r-CM ($r \leqslant m$), *where*

$$\sup_{n\to\infty} \{T(n)/n^{1+[m-r+1/(m+1)r]}\} > 0.$$

Proof. Note that the sequence in question has $f(n) = n^{m+1}$ and $g(n) = a \cdot f(n)$ for some $a > 0$. Thus

$$T(f(n)) > k \cdot [f(n)]^{-1/(m+1)} \cdot [af(n)]^{1+1/r}.$$

The result now follows immediately.

REFERENCES

1. A. COBHAM, Functional equations for register machines. *Proc. Hawaii International Conference on System Sciences* (1968), 10–13.
2. P. C. FISCHER, Turing machines with a schedule to keep, *Information and Control* 11 (1967), 138–146.
3. P. C. FISCHER, A. R. MEYER, AND A. L. ROSENBERG, Real-time counter machines, *Proc. Eighth Ann. Symp. on Switching and Automata Theory* (1967), 148–154.
4. P. C. FISCHER, A. R. MEYER, AND A. L. ROSENBERG, Counter machines and counter languages. *Math. Systems Theory* 2 (1968), 265–283.
5. J. HARTMANIS AND R. E. STEARNS, On the computational complexity of algorithms, *Trans. Amer. Math. Soc.* 117 (1965), 285–306.
6. F. HENNIE AND R. E. STEARNS, Two-tape simulation of multitape turing machines, *J. Assoc. Comput. Mach.* 13 (1966), 533–546.
7. A. R. MEYER, A. L. ROSENBERG, AND P. C. FISCHER, Turing machines with several read-write heads, *Proc. Eighth Ann. Symp. on Switching and Automata Theory* (1967), 117–127.
8. M. L. MINSKY, Recursive unsolvability of Post's problem of tag and other topics in the theory of turing machines, *Ann. of Math.* 74 (1961), 437–455.
9. M. O. RABIN, Real-time computation, *Israel J. Math.* 1 (1963), 203–211.
10. A. L. ROSENBERG, Real-time definable languages, *J. Assoc. Comput. Mach.* (1967), 645–662.
11. A. THUE, Über die Gegenseitige Lage Gleicher Teile Gewisser Zeichenreihen, *Mat. Nat. Klasse* 1, Kristiana (1913).
12. H. YAMADA, Real-time computation and recursive functions not real-time computable. *IRE Trans. E. C.* 11 (1962), 753–760.