REAL TIME COUNTER MACHINES
Preliminary Version

Patrick C. Fischer
Cornell University
Ithaca, N. Y.

Albert R. Meyer
Arnold L. Rosenberg
IBM Watson Research Center
Yorktown Heights, N. Y.

INTRODUCTION

Parallelling recent work on time and space-restricted Turing machine computations, we consider in this paper similar restrictions on counter machines (CM's). Although Turing machines provide a useful model of a computer for the theory of computability, many details in their definition seem unnatural when one considers restrictions on time or space. CM's are as powerful as Turing machines and can be defined formally using little more mathematics than vector addition. For this reason CM's lead to notions of computation time and space which are at least as natural and somewhat more tractable than the corresponding notions for Turing machines. As an example of the tractability of CM's, we obtain a straight forward relation (Theorem B1) between time and space requirements for CM's. We also prove (Theorem D2) that m+1 counters can generate sequences faster than m counters. The corresponding questions for multitape Turing machines have not yet been settled.

CM's have appeared frequently in the study of computability. Minsky (1961) first showed that a machine with two counters (a 2-CM) could simulate a Turing machine. The unlimited register machines of Shepherdson and Sturgis (1963), and the Loop programs of Meyer and Ritchie (1967) are essentially programmed CM's. Kreider and Ritchie's (1966) marking automata are easily shown to be equivalent to linear space bounded counter machines. These latter are also equivalent to the multi-tape finite automata of Kobayashi and Sekiguchi[1] (1966). Recently, Rabin has used a variation of the CM to streamline several results on word problems in semigroups.

Section A contains a description of CM's and conventions for input and output. For convenience we have considered only problems of language recognition and sequence generation. In Section B the hierarchies arising from space bounds on Turing machines and CM's are shown to be isomorphic. Sections C and D contain a comparison of sequences which are real-time (r-t) generable by CM's and by Turing machines. All the sequences which Yamada (1962) proved to be r-t generable by multi-tape Turing machines turn out to be r-t generable by CM's (Corollary C2). There are, however, sequences which are r-t generable by a one tape Turing machine but not by CM's (Theorem D1). Although m+1 counters can generate sequences faster than m counters, sequences which are r-t generable using any number of counters can always be r-t generated by multitape Turing machines with at most four tapes (Theorem D3). The behavior of r-t generable sequences under coordinatewise Boolean operations is also considered. Section E enumerates some of the properties of r-t recognizable languages. Section F contains some remarks on the precise time required for one kind of machine to simulate another.

The study of r-t recognizable languages may be applicable to the design of programming languages which compile rapidly. The work of Hartmanis and Stearns (1965) and Fischer (1966) indicates that r-t generable sequences play a fundamental role in the timing of more general computations. A surprising application of r-t generable sequences has been in proofs by Cobham (1967) of some special cases of the following:

Conjecture. Let a be an irrational number, $0<a<1$. If the infinite binary expansion of a is r-t generable by a CM, then a is a transcendental number.

For these reasons, and as a simplifying assumption in several proofs, the r-t restriction has been emphasized. Most of our results readily extend to more general time restrictions. These extensions follow from the fact that a sequence generable within any computable time restriction is by definition (cf. Section A) a homomorphic image of a r-t generable sequence.

This paper is presented as a preliminary report. Some proofs have been omitted and many are merely sketched.

A. Preliminaries. We consider CM's in two roles: as language recognizers and sequence generators. Recognizers have a read-only input tape with one scanning head, while generators run autonomously. Both variations of CM's consist of finite state units controlling a fixed number of counters each of which may contain any integer[2]. The states of the control unit are partitioned into three classes: accepting, rejecting and intermediate.

At the start of any computation, the control unit is assumed to be in some designated initial state and all counters are set to zero. A step in a CM computation is uniquely determined by the state of the control unit, the particular subset of counters which contain zero, and, in the case of recognizers, the symbol being scanned on the input tape. The action in one step may consist of a change of state in the control unit, a change of absolute value at most one in the contents of each counter, and a shift of at most one square by the reading head of a recognizer. (No writing may take place on the input tape of a recognizer.)

A CM without an input tape is a sequence generator providing that, once started, its computation is infinite. Let $\vec{s} = s_0, s_1, s_2, \ldots$ be the sequence of states of the control unit of a CM at each step of the computation. The (binary) sequence $\vec{\alpha} = \alpha_0, \alpha_1, \alpha_2, \ldots$ generated by the CM is obtained by replacing each accepting state in $\vec{s}$ by one, each rejecting state by zero, and deleting all intermediate states. The time required to generate $\vec{\alpha}$ is defined as a function, T, from the nonnegative integers to the nonnegative integers, where T(n) = m if and only if α_n is the result of replacing s_m by zero or one[3]. The space required to generate $\vec{\alpha}$ is a function, S, where S(n) is the sum of the largest absolute values of the contents of each counter up to the T(n)th step in the computation. A sequence is r-t generable if the time required to generate it is the identity function. (This is equivalent to the assertion that there is a generator for the sequence which has no intermediate states.)

A vocabulary Σ is a finite set of elements called letters. A word over Σ is any finite (possibly empty) sequence of letters of Σ. We let Σ^* denote the set of all words over Σ (including the null word); and we call any subset of Σ^* a language (over Σ). We refer the reader to Rabin and Scott (1959) for the definition of the various operations on words and languages.

Let w be a word composed from some finite vocabulary Σ, and let "\$" be a symbol not in Σ. A CM with an input tape is given input w under the convention that "\$w\$" is placed on the tape, and the CM is started with the head scanning the symbol immediately to the right of the leftmost occurrence of "\$". A CM is a recognizer (with input vocabulary Σ) if, during the computation of the CM with any input w, the reading head never leaves the region of the input tape containing "\$w\$", and the control unit eventually enters a non-intermediate state. The CM accepts (rejects) w if the first non-intermediate state entered by the control unit is an accepting (rejecting) state. The time required to recognize w is the number of steps taken by the CM with input w until a non-intermediate state is entered, and the space required is the sum of the largest absolute values of the contents of each counter up to the time w is processed.

The language recognized by a CM recognizer is the set of words accepted by the CM. The time required to recognize a language is a function T, where T(n) is the largest time required by the CM to recognize any of the words of length n over the input vocabulary. The space required to recognize a language is defined similarly. A language is r-t recognizable if the time required to recognize it is the function T(n)=n+1. (This is equivalent to the assertion that there is a recognizer for the language whose reading head shifts right at every step and which enters a non-intermediate state as soon as it reaches the end of any input word.)

Multitape Turing machines which are generators and recognizers can be defined in the same way that CM generators and recognizers are defined above - the only difference being that Turing machine tapes appear instead of counters, and space is measured by the number of tape squares scanned by the heads on the work tapes. The reader is referred to our other paper in this volume for a more detailed description of multitape Turing machines.

CM's can alternatively be described as a special case of multitape Turing machines whose work tapes remain blank throughout every computation except for a single "marked" square on each tape. The work tapes become the counters - the contents of a counter being defined as the number of tape squares from the marked square to the reading head on the tape. The time and space required by such Turing machines is the same as our definition of the time and space required by CM's.

B. Time and Space Relations. We can find a strong relation between the time and space requirements of CM's.

Theorem B1. Let L be a CM-recognizable language and S be a function with $S(n) \geq n$. L is recognizable in space bounded by S if, and only if, L is recognizable in time bounded by a polynomial in S.

Proof. (1) If L is recognized by a CM with d states, and m counters in space bound S, then in processing inputs of length n, the CM can assume at most $d(n+2)(S(n)+1)^m$ distinct configurations. (A configuration is determined by the state of the control unit, the position of the input head, and the contents of the counters.) Clearly, if a configuration is repeated in the course of a computation, the CM will never halt. The CM must, therefore, operate in time bound $T(n) = d(n+2) \cdot (S(n)+1)^m \leq kS(n)^{m+1}$ for some constant k independent of n.

(2) If time is bounded by a polynomial in S, then so certainly is space, since a CM can increment its counters by at most one at every step. We will describe how any given counter in a CM may be replaced by three counters whose contents remain bounded by a linear function of the square root of the contents of the given counter. By applying this construction several times, any CM with time (and hence, space) bounded by a polynomial in S

can be transformed into a CM which recognizes the same language in space bounded by S. This will prove the theorem.

Suppose that counter A contains a number $m \geq 0$, and let $n = [\sqrt{m}]$ where $[x]$ is the integer part of the real number x. Counters B, C, and D, containing n, $m-n^2$, and zero respectively, will appear in place of A. To simulate a step in which the contents of A are incremented by one, counters B, C, and D operate as follows:

Counter C is incremented by one. It is then compared to twice the contents of B, using D as an auxiliary. If C exceeds twice B, then B is incremented by one, and C is decremented until it contains zero. If C does not exceed twice B, no further operation is necessary.

After this simulated addition, B, C, and D contain $[\sqrt{m+1}]$, $(m+1)-[\sqrt{m+1}]^2$, and zero respectively. Simulated subtraction works by reversing the process of simulated addition. Thus, whenever A contains any $m \geq 0$, B, C and D will contain n, $m-n^2$, and zero. A CM with B, C and D can detect when A contains zero by detecting when B and C both contain zero at the end of a simulated step. Since $m-n^2 \leq 2n$, the contents of B, C and D never exceed one plus twice the square root of the contents of A. The case in which the contents of A become negative can be treated similarly.

Stearns, Hartmanis, and Lewis (1966) have defined, for a function S, the tape complexity class C_S to be the set of languages recognizable by Turing machines with space bound S. Tape complexity classes partially ordered by set inclusion have a rich structure which includes infinitely many incomparable infinite chains. If one similarly defines C_S' to be the languages recognizable within space S by CM's, then one obtains the same structure. The following theorem implies that $C_S' = C_{\log(S)}$ for any function S.[4]

Theorem B2. A language is recognizable by a CM with space bound S if, and only if, it is recognizable by a Turing machine with space bound log (S).

Proof. By encoding the contents of the counters of a CM in (possibly compressed) binary notation, a Turing machine can clearly simulate a CM, with space bound S, using space bounded by log(S).

For the converse, the reader is referred to an algorithm reported in Fischer (1966) for simulating a Turing machine tape with three counters. A careful analysis of that algorithm shows that the numbers in the three counters never exceed $m^{\log(S)+1} \leq mS^{\log m}$ when simulating an m symbol Turing machine which operates in space S.

We close this section with the following open problem:

Open Problem: Is there an analogue to Theorem B1 which applies to Turing machines.

C. Real-time Countable Functions. The notion of a r-t countable function was introduced by Yamada (1962) as a convenient way to describe sequences r-t generable by Turing machines.

The definition carries over immediately to CM's. A strictly increasing function f can be associated with any sequence $\vec{\alpha} = \alpha_0, \alpha_1, \alpha_2, \ldots$ containing infinitely many ones by letting $f(n) = m$ if, and only if, α_m is the nth occurrence of a one in $\vec{\alpha}$. If $\vec{\alpha}$ is r-t generable, the associated function f is called r-t countable, and the r-t generator for $\vec{\alpha}$ is referred to as a counter for f.

A counter for f does not compute f in the natural sense. A CM with at least n counters is said to compute a function g of n variables, if, when the CM is started with integers $x_1, \ldots, x_n$ in its first n counters (and all other counters containing zero), then it leaves $g(x_1, \ldots, x_n)$ in its last counter when its control unit first enters an accepting state. The time required to compute g is the function $T(x_1, \ldots, x_n)$ equal to the number of steps before $g(x_1, \ldots x_n)$ appears and an accepting state is entered.

Thus, the r-t countable functions should not be confused with functions which can be computed within a slow growing time function: the time required to compute f(x) obviously must be at least as large as f(x) - since f(x) steps are necessary simply to leave the value in a counter. Rather, r-t countability is related to the property that a function can be computed in time proportional to its own values.

Definition C1. A function f of n variables is self-computable if, and only if, there is an integer $K > 0$ such that

(1) f can be computed in time bounded by $K \cdot f$, and

(2) $K \cdot f(x_1, \ldots, x_n) \geq \max \{x_1, \ldots, x_n\}$ for all sufficiently large integers $x_1, \ldots, x_n$.[5]

By modifying a counter for a r-t countable function one can show that any r-t countable function is self-computable. Conversely, under quite general circumstances, strictly increasing self-computable functions of one variable are r-t countable. This provides a powerful criterion for showing that functions are r-t countable.

Lemma C1. Let F be a strictly increasing function of one variable, and let $\Delta F(x) = F(x+1)-F(x)$. If ΔF is self-computable, then there is an integer $k' > 0$ such that $k' \cdot F$ is r-t countable.

A CM which computes ΔF in time $\leq k \cdot \Delta F$ can be transformed to a counter for $6k \cdot F$ by adding four counters to it. Roughly speaking, the new CM computes $\Delta F(x)$ from time $6k \cdot F(x)$ to time $6k \cdot F(x) + 6k \cdot \Delta F(x) = 6k \cdot F(x+1)$ at which time it emits a one. The details are omitted.

Lemma C2. If f is a function of one variable and $K \cdot f$ is r-t countable for some integer $K>0$, then f is r-t countable.

Proof. A counter for $K \cdot f$ can be accelerated to become a counter for f along lines similar to Hartmanis and Stearns' (1965) speed-up theorem for multitape Turing machines. When the counter for $K \cdot f$ has counters containing integers $x_1, \ldots, x_n$, the accelerated machine has counters containing $[x_1/K], \ldots, [x_n/K]$, retaining $x_1 - K[x_1/K], \ldots, x_n - K[x_n/K]$ in its finite memory. One step of the accelerated machine can be made to correspond to K steps of the counter for $K \cdot f$. The details are left to the reader.

Theorem C1. If f is a strictly increasing function of one variable and Δf is self-computable, then f is r-t countable.

The proof is immediate.

Corollary C1. If f is a strictly increasing self-computable function of one variable, and there is a real number $a > 1$ such that $f(x+1) \geq a \cdot f(x)$ for all x, then f is r-t countable.

Under the hypotheses of the corollary, one easily shows that Δf is self-computable. The result then follows from Theorem C1.

Many familiar arithmetic functions -- e.g., $x+y$, $x \cdot y$, x^y, $x!$ -- can easily be shown to be self-computable. Similarly straightforward arguments show (1) the self-computable functions are closed under composition and explicit transformation (substituting constants, permuting and identifying variables); and (2) if $F(y, x_1, \ldots, x_n)$ is self-computable, then so are $\Sigma_{i=0}^{y} F(i, x_1, \ldots, x_n)$ and $\Pi_{i=1}^{y} F(i, x_1, \ldots, x_n)$.

Corollary C2. If F and g are positive r-t countable functions, then so are:

(1) $F(g(x))$

(2) $F(x)+g(x)$

(3) $F(x) \cdot g(x)$

(4) $F(x)^{g(x)}$

(5) $\left.F(x)^{F(x)^{\cdot^{\cdot^{F(x)}}}}\right\}$ height $g(x)$

(6) $F(x)!$

(7) $\Pi_{i=1}^{x} F(i)$

(8) $\Sigma_{i=0}^{x} F(i)$

Parts (1), (2), (8) are straightforward. Parts (4)-(7) follow from the preceding remarks and Corollary C1. Part (3) follows from Corollary C1 along with the observations that (i) the function $h(x_1, \ldots, x_6) = x_1x_2 + x_3x_4 + x_5x_6$ is self-computable, and (ii) $\Delta(Fg) = h(F, \Delta F, g, \Delta g, \Delta F, \Delta g)$.

Yamada (1962) proved all of Corollary C2, save part (5) which he conjectured in a weaker form, for functions r-t countable by multitape Turing machines. In fact, all of the examples and properties established by Yamada for Turing machines actually are true for CM's.

D. Counter machines and Turing machines. Let G_m be the set of sequences r-t generable by CM's with at most m counters, and let $G = \bigcup_m G_m$. In this section we derive a number of properties of the classes G_m. Our main tool is stated in Lemma D1.

A length n segment of an infinite binary sequence $\vec{\alpha} = \alpha_0, \alpha_1, \ldots$ is a length n sequence $\alpha_i, \alpha_{i+1}, \ldots, \alpha_{i+n-1}$ of successive elements of $\vec{\alpha}$ $(i \geq 0)$.

Lemma D1. If $\vec{\alpha} \in G_m$, then there is a constant $K>0$ such that for every n the number of distinct length n segments of $\vec{\alpha}$ is bounded by $K \cdot n^m$.

Proof. Suppose $\vec{\alpha}$ is r-t generated by a CM with s states and m counters. A configuration of a generator at any step in its computation is the (m+1)-tuple consisting of the state of its control unit and the contents of its m counters. Two configurations will be called n-equivalent if they differ only in coordinates where both contain integers of absolute value n or larger. If a generator is started in either of two n-equivalent configurations, then its control unit will obviously enter the same sequence of states in the next n steps of either computation. Since $\vec{\alpha}$ is defined as an image of the sequence of states of the control unit, distinct length n segments of $\vec{\alpha}$ can begin appearing only when the generator is in n-inequivalent configurations. By definition there are at most $s \cdot (2n+1)^m$ distinct n-equivalence classes of configurations of the generator for $\vec{\alpha}$. Therefore, the number of distinct length n segments of $\vec{\alpha}$ is also at most $s \cdot (2n+1)^m \leq K \cdot n^m$, for some K independent of n.

Theorem D1. There is a sequence which is r-t generable by a one-tape Turing machine but which is not r-t generable by any CM.

Proof. It is easy to design a one-tape Turing machine which successively prints and scans the positive integers in binary notation. The states of the control unit can be arranged so that accepting or rejecting states are entered after digits zero or one are scanned respectively. The sequence r-t generated by such a machine will include every binary n-tuple as a length n segment. There are 2^n binary n-tuples, so that by Lemma D1 the sequence cannot be in G.

Open Problem: Is there, for any $m \geq 1$, a sequence which is r-t generable by an (m+1)-tape Turing machine but not by any m-tape Turing machine?

In contrast to the above, the corresponding problem for CM's is settled for every m.

Theorem D2. For every $m \geq 1$, G_{m+1} properly includes G_m.

The proof proceeds by constructing an (m+1)-CM which generates a sequence $\vec{\alpha}$ with the following property: For every integer $k \geq 1$, and every binary word w of length k-1 which contains at most m-1 ones, the binary word 0^k1w10^k is a segment of $\vec{\alpha}$. (0^k represents a sequence of k 0's.)

With little difficulty, one now shows that the number of distinct length n segments of $\vec{\alpha}$ exceeds an^{m+1} for some constant $a > 0$. By Theorem D1 the sequence $\vec{\alpha}$, which is in G_{m+1} by construction, is not in G_m, and the proof is complete.

We noted in Section A that an n-CM can be simulated with no time loss by an n-tape Turing machine. We now show that a single tape Turing machine can, with little time loss, simulate any number of counters.

Lemma D2. If a sequence is generable (language is recognizable) by an m-CM in time T(n), then it is generable (recognizable) in time T(n) by an m-CM which alters the contents of at most one counter at each step.

The proof is similar to that of Lemma C2.

Lemma D3. If $\vec{\alpha} \varepsilon G$, then there is a one-tape Turing machine generator for $\vec{\alpha}$ which operates in time $T(n) \leq 6n$.

Proof. We give only a brief sketch of the proof. Say that $\vec{\alpha}$ is r-t generable by an m-CM M which alters at most one counter per step. (CF. Lemma D2.) We construct a one-tape Turing machine T to simulate M as follows:

The tape of T is divided into m "tracks", one for each counter of M, and each track is divided into two channels. A binary representation of a nonnegative integer will appear in each channel of each track, justified so that the low order bits of all integers appear in the same tape square. This encoding is such that if the counters of M contain, at the nth step in the computation, the integers $x_1,\dots, x_m$, then at the nth stage in the simulation, the ith track (i=1,..., m) of T's tape will contain integers y_i and z_i with the properties: (1) $y_i-z_i = x_i$, (2) $y_i+z_i \leq n$, and (3) there is no bit position in which the binary representations of y_i and z_i both contain a one. (Thus $x_i=0$ iff $y_i = z_i = 0$.)

For the case m=3, Figure 1 illustrates a possible tape configuration of T after 15 steps by M. The counters of M contain 7, -5, and 0 respectively.

The process of incrementing (decrementing) counter i consists of incrementing y_i (z_i) in the obvious way and "cleaning up" while returning the head to the low order square so as to maintain condition (3) above.

One easily verifies that, for all n and i, $[n/2^{i+1}]$ steps of M lead to $\leq$i carries and hence require $\leq$ 2i+4 steps of T. Thus n steps of M can be simulated by T(n) steps of T where

$$T(n) \leq \Sigma^{\infty}_{i=0} [n/2^{i+1}](2i+4) = 6n$$

completing the proof.

The following theorem is an immediate consequence of Lemma D3 and the construction given in Fischer (1966) (see also Corollary 3 of Meyer, Rosenberg, and Fischer in this volume).

Theorem D3. Any sequence which is r-t generable by a CM with any number of counters is r-t generable by a multitape Turing machine with only four tapes.

The set G and the set of sequences r-t generable by multitape Turing machines share, in addition to closure under the operations of Section C, closure under the coordinatewise Boolean operations "and", "or" and "complement". For example, if $\vec{\alpha}$ and $\vec{\beta}$ are r-t generable sequences (by either CM's or Turing machines) then so is $\vec{\alpha} \vee \vec{\beta} = \alpha_0 \vee \beta_0, \alpha_1 \vee \beta_1, \dots$ The obvious r-t generator for $\vec{\alpha} \vee \vec{\beta}$ uses the sum of the number of counters (or tapes) required to r-t generate $\vec{\alpha}$ and $\vec{\beta}$. However, for Turing machines the number of tapes required need not grow according to this sum.

The next two theorems, which we state without proof, supply another contrast between CM and Turing machine generators.

Theorem D4. If an infinite binary sequence $\vec{\alpha}$ can be expressed as a Boolean combination of a finite number of sequences each of which is r-t generable by a multitape Turing machine with m tapes, then $\vec{\alpha}$ is r-t generable by a multitape Turing machine with m+3 tapes.

Theorem D5. For any integer $m>0$, there are m sequences $\vec{\alpha}_1, \vec{\alpha}_2, .., \vec{\alpha}_m$ such that each $\vec{\alpha}_i$ is r-t generable by a 5-CM, but $\bigvee_{i=1}^{m} \vec{\alpha}_i$ is not r-t generable by any m-CM.

Theorem D4 implies that four tapes which can interact through a control unit are, for the purposes of time-restricted sequence generation, at least as powerful as any number of tapes operating in parallel. Theorem D5 is based on the fact that a large number of counters, even if they operate in parallel, can r-t generate a sequence with sufficiently many distinct segments that, by Lemma D1, the sequence cannot be generated with a small number of counters. This still leaves unsettled the following:

Open problem. Is there a non-negative integer m such that G is included in the closure of G_m under the coordinatewise Boolean operations?

The necessary condition for a sequence to be r-t generable by an m-CM in Lemma D1 is not a sufficient condition, even for sequences which are r-t generable by Turing machines. This observation is due to A. Cobham, and the following two results are based on his remarks.

Lemma D4. If a sequence $\vec{\alpha}$ is r-t generable by a CM, then there is a word w such that $\vec{\alpha}$ contains arbitrarily long segments of the form ww...w.

Theorem D6. There is a sequence $\vec{\alpha}$ such that (1) $\vec{\alpha}$ is r-t generable by a Turing machine, (2) the number of distinct length n segments in $\vec{\alpha}$ is bounded by a polynomial in n of degree 2, and (3) $\vec{\alpha}$ is not r-t generable by any CM.

One such $\vec{\alpha}$ is the well-known sequence defined by Thue (1913).

E. Real-time Recognizable Languages.

Generation obviously can be regarded as a special case of language recognition. In particular, it is trivially the case that a strictly increasing function, f, is r-t countable if and only if $\{0^n/n \in \text{range}(f)\}$ is a r-t recognizable language. Thus Theorems D1 and D2 imply that a r-t one-tape Turing machine recognizer cannot be simulated by any r-t CM recognizer, and that m+1 counters are better than m counters for r-t recognition.

However, not all of the properties of sequence generators are shared by recognizers in general. For example, a 1-CM is no better than a 0-CM for sequence generation (as the reader can easily verify), but a 1-CM can r-t recognize the language $\{0^n 1^n/n \geq 0\}$. This language is not a regular set and hence not recognizable by a 0-CM, which is equivalent to a finite automaton (Shepherdson, 1959).

Laing (1967), and the authors independently, have found a characterization of a subclass of the r-t recognizable counter languages. Let $\Sigma = \{a_1, \ldots, a_m\}$, and define, for any word w in Σ^*: $\#(w) = \langle (w)_1, \ldots, (w)_m \rangle$ where $(w)_i$ is the number of occurrences of a_i in w.

Theorem. (Laing) Let S be a set of m-tuples of integers, and let $\Sigma = \{a_1, \ldots, a_m\}$. The language L over Σ equal to $\{w: \#(w) \text{ is in } S\}$ is a finite Boolean combination of languages r-t recognizable by 1-CM's if, and only if, S is a semilinear set (Ginsburg and Spanier, 1964).

The proof of Lemma D3 can be applied to languages:

Theorem E1. Any language r-t recognizable by a CM is recognizable in time $T(n) \leq (1+\varepsilon)n$ by a one-tape Turing machine for any $\varepsilon>0$.

The factor 6 of Lemma D3 is reduced to $1+\varepsilon$ by appeal to the "speed-up" theorem of Hartmanis and Stearns (1965). It is still open whether or not ε can be set to 0.

Rosenberg (1967) has exhibited languages which are recognizable in time $(1+\varepsilon)n$ by one-tape Turing machines but which are not r-t recognizable with any number of tapes. The distinction between linear- and real-time recognizability applies to CM's as well as to Turing machines.

Let $L = \{0^n/1^m: n \geq m \geq 1\}$ and let L* represent the Kleene closure of L (CF. Rabin and Scott, 1959).

Theorem E2. The language L* is not r-t recognizable by any CM, but it is recognizable in time $T(n) \leq (1+\varepsilon)n$ by a 1-CM with a one-way read head, for any $\varepsilon > 0$.

It is worth noting that the language L* is r-t recognizable by a 1-CM which can set its counter to zero in one step. Such "store zero" CM's are a natural extension of our model. Except for Theorem E2, all the results in this paper apply equally well to this augmented model.

The closure properties of the class of real-time recognizable counter languages are practically identical to those of the real-time definable languages of Rosenberg (1967). Therefore, we merely state the basic lemma and a number of representative closure properties, referring the reader to Rosenberg's paper for more details.

Lemma E1. Let x, y, z be words and L be a language over some fixed vocabulary. For each integer $n \geq 0$, the equivalence relation "$\overset{n}{\equiv}$(Mod L)" is defined as follows: $x \overset{n}{\equiv} y$ (mod L) if for all words z of length at most n, $xz \in L$ when and only when $yz \in L$. If L is r-t recognizable by an m-CM, then there is a constant $K>0$ such that the number of equivalence classes of words under $\overset{n}{\equiv}$ (mod L) is at most $K \cdot n^m$ for all n.

Theorem E3. The languages r-t recognizable by CM's are closed under union, intersection, and relative complementation, but are not closed under the operations of concatenation with a regular set, reversal, Kleene closure, or length preserving homomorphisms.

Using Lemma E1, one can easily prove

Theorem E4. (a) For all m, n > 0, there exist a language K, r-t recognizable by an m-CM, and a language L, r-t recognizable by an n-CM, such that $K \cup L$ ($K \cap L$) is r-t recognizable by a (m+n)-CM but not by any (m+n-1)-CM.[6] (b) For any m > 0, there are m+1 languages, each r-t recognizable by 1-CM's, whose union is not r-t recognizable by any m-CM.

F. Simulation of one machine by another. The authors have proved a number of results about the time needed to simulate one type of machine on another. Since the results are still incomplete, their reporting is being deferred to the final version of this paper.

FOOTNOTES

1 The equivalence of marking automata and multitape finite automata was pointed out to the authors by Alan Cobham. Cobham independently observed that these machines were equivalent to Turing machines with a logarithmic space bound. This also follows as a corollary of our Theorem B2.

2 Counters appearing elsewhere in the literature have sometimes been defined to contain only nonnegative integers. There is no loss of generality in such a restriction, but some algorithms are easier to describe using both positive and negative integers; hence, the definition above.

3 We shall consider only infinite binary sequences throughout the paper.

4 We use base 2 logarithms throughout the paper.

5 In the arguments below we shall, as a matter of convenience, often tacitly assume that criterion (2) holds for all $x_1, \ldots, x_n$.

6 Lemma E1 was discovered independently by Laing (1967) who used it to prove a special case of Theorem E4(b).

REFERENCES

Cobham,A. Private memorandum (1967).

Fischer,P.C. Turing machines with restricted memory access, Inf Control, 9 (1966), 364-379.

Fischer,P.C. Turing machines with a schedule to keep, to appear Inf Control (1967).

Ginsburg,S. and Spanier,E.H. Bounded ALGOL-Like languages, Trans Amer Math Soc, 113 (1964) 333-368.

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

Kobayashi,K. and Sekiguchi,S. On the class of predicates decidable by two-way multitape finite automata, Jour Assoc Comp Mach, 13 (1966), 236-261.

Kreider,D.L. and Ritchie,R.W. A basis theorem for a class of two-way automata, Zeit Math Logik und Grundlagen, 12 (1966), 243-255.

Laing,R. Realization and complexity of commutative events, Univ of Mich Technical Report 03105-48-T (1967).

Meyer,A.R. and Ritchie,D.M. The complexity of Loop programs, Proc 20th Anniv Conf of Assoc Comp Mach (1967).

Minsky,M. Recursive unsolvability of Post's problem of Tag and other topics in the theory of Turing machines, Ann Math, 74 (1961), 437-455.

Parikh,R.J. On context-free languages, Jour Assoc Comp Mach, 13 (1966), 570-581.

Rabin,M. and Scott,D. Finite automata and their decision problems, IBM Journ Res Dev, 3 (1959) 114-125.

Rosenberg,A.L. Real-time definable languages, to appear Jour Assoc Comp Mach (1967).

Shepherdson,J.C. The reduction of two-way automata to one-way automata, IBM Jour Res and Dev, 3 (1959), 198-200.

Shepherdson,J.C. and Sturgis,H.E. Computability of recursive functions, Jour Assoc Comp Mach, 10 (1963) 217-255.

Thue,A. Ueber die gegenseitige Lage gleicher Teile gewisser Zeichenreihen, Skrifter utgit av Videnskapsselskapet i Kristiania, 1912 Mat Nat Klasse No. 1, Kristiania (1913) 67 pages.

Yamada,H. Real-time computation and recursive functions not real-time computable, IRE Trans Elec Comp, EC-11(1962),753-760.

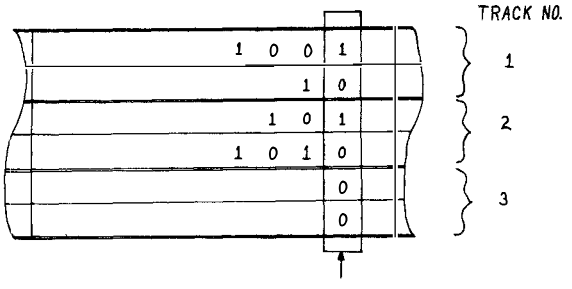

Figure 1:
A tape representing three counters containing 7, -5, and 0, respectively.