Real-Time Definable Languages

ARNOLD L. ROSENBERG

IBM Watson Research Center, Yorktown Heights, New York

ABSTRACT. The closure properties of the class of languages defined by real-time, online, multitape Turing machines are proved. The results obtained are, for the most part, negative and, as one would expect, asymmetric. It is shown that the results remain valid for a broad class of real-time devices. Finally, the position of the class of real-time definable languages in the "classical" linguistic hierarchy is established.

1. *Introduction*

A major problem in the theory of artificial languages is that of finding efficient algorithms for syntactic analysis. If an automaton processes a new input symbol at each time unit, and decides membership in the language immediately after scanning the last input symbol, then, in some sense, the automaton recognizes the language in minimal time. Automata which operate in this manner are said to operate in *real time*, and the class of languages defined by such automata are called real-time definable languages. In this paper we study the properties of a class of automata which operate in real time and of the class of languages defined by such automata.

We begin in Section 2 with a definition and description of the model under investigation.

In Section 3 the positive closure properties of the class **R** of real-time definable languages are given. In particular it is shown that **R** is a Boolean algebra, and that **R** is closed under minimization, under the inverse of a real-time transducer mapping and under suffixing with a regular set. Using the closure of **R** under union, we exhibit an inherently ambiguous real-time definable context-free language.

In Section 4 the negative results about **R** are given. It is proved that **R** is not closed under concatenation, even with regular sets, nor under the Kleene closure operator[1] or reversal. Several less important operations are also considered, namely, mapping by sequential machine, the operations of taking derivatives and quotients, and the operation of maximization. As a corollary of these results, we exhibit a language which is definable in "twice real time" but which is not in **R**. Finally we show that there is a deterministic context-free language (Ginsburg and Greibach [2]) which is not in **R**, establishing the position of **R** in the linguistic hierarchy. The proofs of our negative results depend only on the amount of information our automata can access as a function of time. We exploit this fact to show that our results are valid for a rather broad class of machines which includes most of the machines which have been used in the literature to study real-time computation. This observation settles some open questions concerning these other classes of automata.

[1] If A is a language, then A^k denotes the k-fold concatenation of A with itself, A^0 being the language consisting of the null word. The *closure* of A is then $A^* = \bigcup_{k \geq 0} A^k$.

In Section 5 we mention a number of problems related to real-time definability which are recursively unsolvable. In particular, we indicate that the problem of deciding whether or not a language is real-time definable is unsolvable.

2. *The Model Studied*

2.1 We assume familiarity with the concepts of alphabet, tape, and set of tapes, as well as with the various operations on tapes and sets of tapes.

We denote by $L(t)$ the length of tape t. For finite sets A, we denote by $\#(A)$ the cardinality of A. Letting $A \times B$ denote the cross-product of sets A and B, $[A]^k$ is defined recursively by:

$$[A]^1 = A$$
$$[A]^{i+1} = A \times [A]^i.$$

2.2 The model we study here is the online multitape Turing machine of Hartmanis and Stearns [5] and Rabin [8].

An n-tape online Turing machine (n-TM) comprises a finite-state control and n two-way infinite tapes on each of which is positioned a read-write scanning device.

At the outset of a computation, the n-TM is in a designated initial state, and its read-write heads are positioned on arbitrary squares of the initially blank scratch tapes (we denote the blank symbol by B). The state set of the n-TM is partitioned into "polling" states and "autonomous" states.

If the n-TM is in a polling state, then it receives a single input symbol. On the basis of the current state, the input symbol, and the symbols scanned on the scratch tapes, the n-TM changes state, and each read-write head writes at most one symbol from the working alphabet of the n-TM and moves independently at most one square to the left or right on its scratch tape. The initial state must be a polling state.

If the n-TM is in an autonomous state, then it does not receive any input symbol; but it goes through one primitive action as described above solely on the basis of the current state and the symbols being scanned on the scratch tapes.

The polling states are partitioned into accepting and rejecting states. An input sequence is accepted (resp. rejected) by an n-TM if, when started in its initial state with blank scratch tapes, and with the input sequence available at its input terminal, the n-TM goes through a sequence of the primitive actions described above and ends up in an accepting state (resp. does not end up in an accepting state).

An n-TM operates in real time if all states of the n-TM are polling states. It is well-known that this "real time" restriction severely restricts the computing power of an n-TM.

More formally,

Definition 1. An n-tape ($n \geq 0$) online Turing machine, n-TM, is an 8-tuple $(K_p, K_a, \Sigma, W, M, S, s_0, F)$ where

(1) K_p and K_a are disjoint finite sets (the *polling* and *autonomous* states, respectively);

(2) Σ is an alphabet (of input symbols), and W is an alphabet (of working symbols, containing B);

(3) M is the state-transition function which maps $(K_p \times \Sigma \times [W]^n) \cup (K_a \times [W]^n)$ into $K_p \cup K_a$;
(4) S is the action function which maps $(K_p \times \Sigma \times [W]^n) \cup (K_a \times [W]^n)$ into $\{R, L, N\}]^n \times [W]^n$;
(5) s_0 (the initial state) is a member of K_p;
(6) F (the accepting states) is a subset of K_p.

When Σ and W are clear from context, we often denote an n-TM as (K_p, K_a, M, S, s_0, F).

A *configuration* of an n-TM $(K_p, K_a, \Sigma, W, M, S, s_0, F)$ is an $(n+2)$-tuple $C = (q, t, x_1my_1, \cdots, x_nmy_n)$ where q is a state, t is in Σ^*, the x_i and y_i are nonnull members of W^*, and m is a special marker (not in W) indicating the positions of the read-write heads of the n-TM. If q is a member of K_p, and q'' a member of K_a, then

$$(q, at, x_1u_1mv_1y_1, \cdots, x_nu_nmv_ny_n) \rightarrow (q', t, x_1'my_1', \cdots, x_n'my_n')$$

or

$$(q'', t, x_1u_1mv_1y_1, \cdots, x_nu_nmv_ny_n) \rightarrow (q', t, x_1'my_1', \cdots, x_n'my_n')$$

for a in Σ, t in Σ^*, the u_i and v_i in W, the x_i, y_i in W^*, and the x_i' and y_i' nonnul[1] members of W^*, under the following conditions:

(1) $q' = M(q, a, v_1, \cdots, v_n)$ (resp., $q' = M(q'', v_1, \cdots, v_n)$);
(2) if $S(q, a, v_1, \cdots, v_n)$ (resp., $S(q'', v_1, \cdots, v_n)$) is equal to $(D_1, \cdots, D_n, w_1, \cdots, w_n)$, each D_i in $\{R, L, N\}$, each w_i in W, then for $i = 1, \cdots, n$, $x_i'my_i'$ is equal to $x_iu_imw_iy_i$ or $x_imu_iw_iy_i$ or $x_iu_iw_imy_i$ according as $D_i = N$ or L or R.[2]

We write $C = (q, t, x_1my_1, \cdots, x_nmy_n) \Rightarrow C' = (q', t', x_1'my_1', \cdots, x_n'my_n')$ if either $C = C'$ or if there is a sequence $C_1, \cdots, C_k$ of configurations with $C_1 = C$, $C_k = C'$, and $C_i \rightarrow C_{i+1}$ for each i.

The *computation* of a tape t in Σ^* by an n-TM is a sequence of configurations, $(s_0, t, BmB, \cdots, BmB) = C_0, C_1, \cdots, C_j, \cdots$ where $C_i \rightarrow C_{i+1}$ for all i. The computation is *proper* if, for some k, $C_k = (q, \lambda, x_1my_1, \cdots, x_nmy_n)$ (λ being the null tape) and q is in K_p. (One readily verifies that a computation is proper iff the n-TM halts, and that the attained state q is unique if it exists.) The n-TM *accepts* the tape t if there is a proper computation of t ending in an accepting state; i.e., if $(s_0, t, BmB, \cdots, BmB) \Rightarrow (q, \lambda, x_1my_1, \cdots, x_nmy_n)$ for q in F. The n-TM *rejects* t if it does not accept t.

Definition 2. Let $\mathfrak{z}$ be a computable monotone increasing function from the natural numbers into the natural numbers. An n-TM *operates in time* $\mathfrak{z}(p)$ if, for every tape t of length p, there is a proper computation of t by the n-TM of length not exceeding $\mathfrak{z}(p) + 1$.

Our main interest is in a subclass of the class of n-TM's, namely, those which operate in time $\mathfrak{z}(p) = p$.

Definition 3. A *real-time* n-tape ($n \geq 0$) online Turing machine, n-RTTM, is an n-TM $(K_p, \phi, M, S, s_0, F)$. For brevity, we often denote an n-RTTM by (K, M, S, s_0, F).

Our notion of acceptance and rejection by an n-RTTM can be strengthened:

[2] If either x_i or y_i is null, we replace it by B to conform to our definition of configuration.

A tape $t = a_1 \cdots a_m$ ($m \geq 0$) in Σ^* is accepted (resp., rejected) by an n-RTTM $\mathbf{T}$ if the computation of t by $\mathbf{T}$ is a sequence $C_0, \cdots, C_m$, and the state of $\mathbf{T}$ in configuration C_m is a member of F (resp., $K - F$).

We denote by $A(\mathbf{T})$ the set of tapes (the language) defined (alternatively, accepted) by the n-TM $\mathbf{T}$. The class of languages accepted by multitape online RTTM's is denoted $\mathbf{R}$, and it is called the class of *real-time definable languages* (RTDL's).

Remark. The number of squares of work tape that an n-RTTM can scan while computing input sequence t is clearly no greater than $nL(t)$. It follows that any language in $\mathbf{R}$ is definable by a deterministic linear bounded automaton (Kuroda [6]), whence $\mathbf{R}$ is a subclass of the context-sensitive languages.

In Section 4.8, we show that $\mathbf{R}$ is a proper subclass.

3. *Positive Closure Properties*

In this section we demonstrate the closure of $\mathbf{R}$ under a number of operations.

3.1 Boolean Operations

THEOREM 1. *The class of real-time definable languages is closed under complementation, intersection, and union. It is thus a Boolean algebra.*

PROOF. Let X and Y be languages defined by the n-RTTM $\mathbf{T}_X = (K_X, M_X, S_X, s_0, F_X)$ and the m-RTTM $\mathbf{T}_Y = (K_Y, M_Y, S_Y, t_0, F_Y)$, respectively.

(a) $\bar{X}$, the complement of X, is defined by the n-RTTM $\overline{\mathbf{T}}_X = (K_X, M_X, S_X, s_0, K_X - F_X)$.

In effect, $\overline{\mathbf{T}}_X$ simulates the computation by $\mathbf{T}_X$, merely "switching answers" at each step.

Clearly $A(\overline{\mathbf{T}}_X) = \overline{A(\mathbf{T}_X)} = \bar{X}$.

(b) Assuming that $\mathbf{T}_X$ and $\mathbf{T}_Y$ are defined over the same input alphabet, the intersection of X and Y is defined by the $(m + n)$-RTTM $\mathbf{T}_V = (K_V, M_V, S_V, v_0, F_V)$ where

(1) $K_V = K_X \times K_Y$.

(2) For $\langle s, t\rangle$ in $K_X \times K_Y$, $w_1, \cdots, w_n$ in W_X, and $w_1', \cdots, w_m'$ in W_Y, $M_V(\langle s, t\rangle, \sigma, w_1, \cdots, w_n, w_1', \cdots, w_m') = \{M_X(s, \sigma, w_1, \cdots, w_n)\} \times \{M_Y(t, \sigma, w_1', \cdots, w_m')\}$.

(3) If
$S_X(s, \sigma, w_1, \cdots, w_n) = \langle D_1, \cdots, D_n, \beta_1, \cdots, \beta_n\rangle$
and
$S_Y(t, \sigma, w_1', \cdots w_m') = \langle D_1', \cdots, D_m', \delta_1, \cdots, \delta_m\rangle$
then
$S_V(\langle s, t\rangle, \sigma, w_1, \cdots, w_m, w_1', \cdots w_m') = \langle D_1, \cdots, D_n, D_1', \cdots, D_m', \beta_1, \cdots, \beta_n, \delta_1, \cdots, \delta_m\rangle$.

(4) $v_0 = \langle s_0, t_0\rangle$.

(5) $F_V = F_X \times F_Y$.

$\mathbf{T}_V$ simulates $\mathbf{T}_X$ on tapes $1, \cdots, n$; and it simulates $\mathbf{T}_Y$ on tapes $(n + 1), \cdots, (n + m)$. $\mathbf{T}_V$ accepts an input tape when, and only when, both $\mathbf{T}_X$ and $\mathbf{T}_Y$ accept it; therefore, $A(\mathbf{T}_V) = A(\mathbf{T}_X) \cap A(\mathbf{T}_Y) = X \cap Y$.

(c) The closure of $\mathbf{R}$ under union follows from parts (a) and (b) and De Morgan's law. Q.E.D.

It is obvious that the sets $A = \{a^i b^i c^j : i, j \geq 1\}$ and $B = \{a^i b^j c^j : i, j \geq 1\}$ are both real-time definable. Since $A \cap B$ is not context-free, and since $A \cup B$ is inherently ambiguous (see Ginsburg and Ullian [4]), Theorem 1 yields the following two corollaries.

COROLLARY 1.1. *There are real-time definable languages which are not context-free.*

COROLLARY 1.2. *There are inherently ambiguous real-time definable context-free languages.*

Rabin [8] has proved that, if $\mathbf{T}_1$ is a 1-RTTM and $\mathbf{T}_2$ is a 1-RTTM, then it may be the case that $A(\mathbf{T}_1) \cup A(\mathbf{T}_2)$ (and, hence, $A(\mathbf{T}_1) \cap A(\mathbf{T}_2)$) is definable by a 2-RTTM, but by no 1-RTTM. Thus, the construction of Theorem 1(b) may be the best possible with respect to the number of scratch tapes.

3.2 SUFFIXING WITH A REGULAR SET. In this section we prove that, whenever a set A is real-time definable, then for every regular set B, AB is real-time definable. In Section 4.1, we shall show that BA need not be real-time definable, thus obtaining the first asymmetric result about $\mathbf{R}$.

Definition 4. A finite automaton (FA) with input alphabet Σ is a 4-tuple $\mathbf{T} = (K, M, s_0, F)$ where

(1) K is a finite set of states;
(2) M is the state-transition function which maps $K \times \Sigma$ into K;
(3) $s_0 \in K$, the initial state;
(4) $F \subseteq K$, the accepting states.

An FA is, thus, a 0-RTTM.

An FA-definable language is called a regular set. Thus, every regular set is real-time definable.

THEOREM 2. *If A is a real-time definable language, and if R is a regular set, then the language AR is real-time definable.*

PROOF. Let A be defined by the n-RTTM,

$$\mathbf{T}_A = (K_A, M_A, S_A, s_0, F_A),$$

and let R be defined by the FA,

$$\mathbf{T}_R = (K_R, M_R, t_0, F_R).$$

The n-RTTM,

$$\mathbf{T}_{AR} = (K_{AR}, M_{AR}, S_{AR}, u_0, F_{AR}),$$

defined as follows, recognizes the set AR.

(1) $K_{AR} = K_A \times 2^{K_R}$ where 2^K is the set of all subsets of the set K.
Let $u = \langle s, X \rangle$ where $s \in K_A$ and $X \subseteq K_R$. For σ in Σ and $w_1, \cdots, w_n$ in W,

(2) $M_{AR}(u, \sigma, w_1, \cdots, w_n) =$
(i) $\{M_A(s, \sigma, w_1, \cdots, w_n)\} \times (\{t_0\} \cup \bigcup_{t \in X} \{M_R(t, \sigma)\})$ if
$M_A(s, \sigma, w_1, \cdots, w_n)$ is in F_A;
(ii) $\{M_A(s, \sigma, w_1, \cdots, w_n)\} \times (\bigcup_{t \in X} \{M_R(t, \sigma)\})$ otherwise.

(3) $S_{AR}(u, \sigma, w_1, \cdots, w_n) = S_A(s, \sigma, w_1, \cdots, w_n)$.

(4) $u_0 = \begin{cases} \langle s_0, \{t_0\}\rangle & \text{if } s_0 \text{ is in } F_A \\ \langle s_0, \phi\rangle & \text{otherwise.} \end{cases}$

(5) $F_{AR} = \{u \in K_{AR} : u = \langle s, X\rangle \text{ and } X \cap F_R \neq \phi\}$.

The proof that $A(\mathbf{T}_{AR}) = A(\mathbf{T}_A)A(\mathbf{T}_R) = AR$ is straightforward and is omitted. Q.E.D.

3.3 Inverse of a Real-Time Transducer Mapping. In this section we show that the inverse image of a RTDL under mapping by a real-time transducer is again real-time definable. In Section 4 we shall show that the direct image of a set in $\mathbf{R}$ under sequential machine mapping (a degenerate case of real-time transducer mapping) need not be real-time definable.

An n-tape real-time transducer, n-RTT, is essentially an n-RTTM with outputs. More precisely, an n-RTT is an 8-tuple $\mathbf{U} = (Q, \Sigma, V, \Delta, N, S, P, q_0)$ where

(1) Q is the set of states of $\mathbf{U}$;

(2) Σ is the *input* alphabet of $\mathbf{U}$;
V is the *working* alphabet of $\mathbf{U}$;
Δ is the *output* alphabet of $\mathbf{U}$;

(3) N is the state-transition function which maps $Q \times \Sigma \times [V]^n$ into Q;

(4) S is the action function which maps $Q \times \Sigma \times [V]^n$ into $[\{R, L, N\}]^n \times [V]^n$;

(5) P is the output function which maps $Q \times \Sigma \times [V]^n$ into Δ^*;

(6) $q_0 \in Q$ is the initial state.

An n-RTT operates much as does an n-RTTM with two exceptions. First, an n-RTT has no distinguished "accepting" states. Second, and more basic, at every primitive action, the n-RTT emits a (possibly null) output word over its output alphabet Δ.

Let $\mathbf{U} = (Q, \Sigma, V, \Delta, N, S, P, q_0)$ be an n-RTT. The P-function is extended to tapes in the obvious way; if q is in Q; σ in Σ; $w_1, \cdots, w_m$ in V, then for any tape σx over Σ $P(q, \sigma x, w_1, \cdots, w_m) = P(q, \sigma, w_1, \cdots, w_n)P(t, x, u_1, \cdots, u_n)$ where $t = N(q, \sigma, w_1, \cdots, w_n)$ and $u_1, \cdots, u_n$ are the symbols under scan on the work tapes of $\mathbf{U}$ after the primitive action caused by state q, input symbol σ, and work symbols $w_1, \cdots, w_n$.

For any tape x over Σ we denote by $\mathbf{U}(x)$ the tape $P(q_0, x, B, \cdots, B)$ over Δ. We further denote by $L_{\mathbf{U}}$ the integer

$$L_{\mathbf{U}} = \max_{\substack{q \in Q \\ \sigma \in \Sigma \\ w_i \in V}} \{L(P(q, \sigma, w_1, \cdots, w_n))\}.$$

For any language A and n-RTT $\mathbf{U}$, the inverse image of A under $\mathbf{U}$ is the set

$$\mathbf{U}^{-1}(A) = \{x : \mathbf{U}(x) \text{ is in } A\}.$$

Our proof of the closure of $\mathbf{R}$ under inverse n-RTT mapping depends on the fact that when an n-RTTM is presented with an input sequence of length k, it can alter at most k squares on any scratch tape.

Assuming familiarity with the so-called timing functions of Hartmanis and Stearns [5], we proceed to our theorem by a series of intermediate results. (Cf. Definition 2.)

Theorem 3 (Hartmanis and Stearns [5]). *If the language A is definable in time $L(t) + E(L(t))$, $E(L(T)) \geq 0$, then, for any $k > 0$, A is definable in time $L(t) + [kE(L(t))]$.*[3]

[3] If x is a real number, then $[x]$ denotes the least integer k such that $k \geq x$.

In particular,

COROLLARY 3.1. *If A is definable in time $cL(t)$ for some $c \geq 1$, then A is definable in time $(1 + [(c - 1)/k])L(t)$ for any $k > 0$.*

The import of Theorem 3 is that, given an n-TM, for any integer r, one can effectively find an n-TM which does in one scratch tape operation what the original did in r scratch tape operations between polling input symbols.

Unfortunately, an n-TM which recognizes a set A in time $2L(t)$ cannot, in general, be replaced by any m-RTTM. In Section 4.3 we present a set A which is $2L(t)$-recognizable, but which is not in **R**. However, if the compression of r operations into one suffices to speed up an n-TM to operate in real time, then the algorithm of Hartmanis and Stearns [5, Theorem 2, p. 290] affords us a method for effecting this speed-up. In particular, if an n-TM reads its inputs at a constant rate, the algorithm will produce an equivalent n-RTTM. We now show that the n-TM which defines $\mathbf{U}^{-1}(A)$ for A in **R** is of this special variety.

THEOREM 4. *If A is a RTDL and if $\mathbf{U}$ is a real-time transducer, then $\mathbf{U}^{-1}(A)$ is a RTDL.*

PROOF. Let the RTDL A be defined by the n-RTTM $\mathbf{T} = (K, \Delta, W, M, S, s_0, F)$, and let $\mathbf{U} = (Q, \Sigma, V, \Delta, N, R, P, q_0)$ be an m-RTT. Note that Δ is the input alphabet of $\mathbf{T}$ and the output alphabet of $\mathbf{U}$.

The $(m + n)$-TM $\mathbf{T}' = (L_p, L_a, \Sigma, W \cup V, M', S', s_0', F')$ defines $\mathbf{U}^{-1}(A)$ in time $(c + 1)L(t)$ where $c = L_{\mathbf{U}}$.

(1) $L_p = K \times Q \times \{\lambda\}$ where λ is the null word;

(2) L_a is empty (i.e., $\mathbf{T}'$ is an $(m + n)$-RTTM) if $c = 0$; otherwise, $L_a = K \times Q \times \bigcup_{i \leq c} \Delta^i$ where Δ^i is the set of all words over Δ of length i.

Let a be in Σ, $w_1, \cdots, w_n$ in W, and $v_1, \cdots, v_m$ in V.

(3) For $\langle s, q, \lambda \rangle$ in L_p,

$$M'(\langle s, q, \lambda \rangle, a, w_1, \cdots, w_n, v_1, \cdots, v_m) = \langle s, N(q, a, v_1, \cdots, v_m), P(q, a, v_1, \cdots, v_m) \rangle,$$

and

$$S'(\langle s, q, \lambda \rangle, a, w_1, \cdots, w_n, v_1, \cdots, v_m) = \langle \underbrace{N, \cdots, N}_{n}, D_1, \cdots, D_m, w_1, \cdots, w_n, u_1, \cdots, u_m \rangle$$

where $R(q, a, v_1, \cdots, v_m) = \langle D_1, \cdots, D_m, u_1, \cdots, u_m \rangle$.

(4) For $\langle s, q, b_1 \cdots b_r \rangle$ in L_a ($r \geq 1$ and b_i in Δ),

$$M'(\langle s, q, b_1 \cdots b_r \rangle, w_1, \cdots, w_n, v_1, \cdots, v_m) = \langle M(s, b_1, w_1, \cdots, w_n), q, x \rangle$$

where

$$x = b_2 \cdots b_r \quad \text{if} \quad r > 1, \quad \text{and} \quad x = \lambda \quad \text{if} \quad r = 1,$$

and

$$S'(\langle s, q, b_1 \cdots b_r \rangle, w_1, \cdots, w_n, v_1, \cdots, v_m) = \langle D_1, \cdots, D_n, \underbrace{N, \cdots, N}_{m}, u_1, \cdots, u_n, v_1, \cdots, v_m \rangle$$

where

$$S(s, b_1, w_1, \cdots, w_n) = \langle D_1, \cdots, D_n, u_1, \cdots, u_n \rangle.$$

(5) $s_0' = \langle s_0, q_0, \lambda \rangle$.

(6) $F' = F \times Q \times \{\lambda\}$.

Thus $\mathbf{T}'$ alternately simulates $\mathbf{T}$ on tapes $1, \cdots, n$ and $\mathbf{U}$ on tapes $n+1, \cdots, n+m$. It is clear from the description of $\mathbf{T}'$ that $A(\mathbf{T}') = \mathbf{U}^{-1}(A(\mathbf{T})) = \mathbf{U}^{-1}(A)$. Now, $\mathbf{T}'$ scans a new input symbol after no more than c scratch tape operations. Obviously, by adding "dummy" symbols, we could alter $\mathbf{T}'$ so that it read at the constant rate of c operations per input symbol. Thus, if we could compress c scratch tape operations of $\mathbf{T}'$ into one, we could dispose of the autonomous states L_a, and, thereby speed $\mathbf{T}'$ up from $(c+1)L(t)$ to real time. However, the algorithm of Hartmanis and Stearns does just this. Thus, using the above construction and their algorithm, we can effectively define an $(m+n)$-RTTM $\mathbf{T}''$ such that $A(\mathbf{T}'') = A(\mathbf{T}') = \mathbf{U}^{-1}(A)$. Q.E.D.

A generalized sequential machine (gsm) is a 0-RTT. Theorem 4 thus yields

COROLLARY 4.1. *If A is a RTDL, and if $\mathbf{G}$ is a gsm, then $\mathbf{G}^{-1}(A)$ is a RTDL.*

COROLLARY 4.2. *If some homomorphic[4] image of a language A is real-time definable, then A is real-time definable.*

In Section 4.4 we shall prove that the inverse image of a RTDL under *nondeterministic* sequential machine mapping need not be in $\mathbf{R}$.

3.4 MINIMIZATION. In this section we demonstrate the closure of $\mathbf{R}$ under the operation of minimization.

We say a tape x is a proper prefix of a tape y, denoted $x < y$, if there is a nonnull tape z such that $y = xz$. Let A be a language. We define $\min A = \{y : y \in A$, and if $x < y$, $x \notin A\}$.

THEOREM 5. *The class of real-time definable languages is closed under minimization.*

PROOF. Let the RTDL A be defined by the n-RTTM $\mathbf{T} = (K, M, S, s_0, F)$. Then $\min A$ is defined by the n-RTTM $\mathbf{T}_{\min} = (K', M', S', s_0, F)$ where

(1) $K' = K \cup \{d\}$ where d is a symbol not in K;

For σ in Σ, and $w_1, \cdots, w_n$ in W,

(2) $M'(s, \sigma, w_1, \cdots, w_n) = \begin{cases} M(s, \sigma, w_1, \cdots, w_n) & \text{if } s \text{ is in } K - F \\ d & \text{if } s \text{ is in F,} \end{cases}$

$M'(d, \sigma, w_1, \cdots, w_n) = d$;

(3) $S'(s, \sigma, w_1, \cdots, w_n) = S(s, \sigma, w_1, \cdots, w_n)$ for s in K,

$S'(d, \sigma, w_1, \cdots, w_n) = [\{N\}]^n \times \{\langle w_1, \cdots, w_n \rangle\}$.

$\mathbf{T}_{\min}$ simulates the action of $\mathbf{T}$ until $\mathbf{T}$ would accept some prefix of the input tape. $\mathbf{T}_{\min}$ then enters a "dead" state d. Clearly, $A(\mathbf{T}_{\min}) = \min A(\mathbf{T})$. Q.E.D.

One can define a max operator symmetrically to the definition of min. In Section 4, we shall prove that $\mathbf{R}$ is not closed under max, thus obtaining yet another asymmetric result.

3.5 TAPE CONSERVATION. Thus far we have avoided explicit mention of the minimum number of tapes needed to recognize a particular RTDL. In general, the

[4] A *homomorphism* of Σ^* into Δ^* is a mapping h such that $h(\lambda) = \lambda$, $h(a)$ is in Δ^* for each a in Σ, and $h(a_1 \ldots a_k) = h(a_1) \ldots h(a_k)$ for each sequence $a_1 \ldots a_k$ of elements of Σ. For $A \subseteq \Sigma^*$, the image of A under the homomorphism h is the set $h(A) = \{h(x) : x$ is in $A\}$.

problem of whether or not an $(n + 1)$-RTTM is strictly more powerful than an n-RTTM is still open. Rabin [8] has showed that a 2-RTTM is strictly more powerful than a 1-RTTM. We can, however, assert the following.

Remark. Let $\mathbf{R}(n)$ denote the class of languages defined by n-RTTM's. Then, for all n,

(a) $\mathbf{R}(n)$ is closed under complementation;
(b) $\mathbf{R}(n)$ is closed under union and intersection with regular sets;
(c) $\mathbf{R}(n)$ is closed under suffixing with a regular set;
(d) $\mathbf{R}(n)$ is closed under inverse gsm mapping;
(e) $\mathbf{R}(n)$ is closed under minimization.

Remark. $\mathbf{R}(1)$ is not closed under union or intersection, nor under inverse real-time transducer mapping.

A more detailed study of $\mathbf{R}(n)$ will be the subject of further research.

4. *Negative Closure Properties*

We now exhibit several operations which do not preserve real-time definability.

4.1 Concatenation. In this section we prove that $\mathbf{R}$ is not closed under concatenation, even with regular sets. The proof depends on a series of definitions and lemmata.

Definition 5. Let A be a set of tapes. The relation $E_k(\mathrm{mod}\, A)$ is defined as follows for tapes x and y: $xE_ky(\mathrm{mod}\, A)$ if for all tapes z of length less than or equal to k, xz is in A when, and only when, yz is in A.

Obviously, $E_k(\mathrm{mod}\, A)$ is an equivalence relation.

Lemma 1 (Hartmanis and Stearns [5]). *If* $\mathbf{T}$ *is an n-RTTM with d states and w working symbols (including B), then the number of equivalence classes of* $E_k(mod\ A(\mathbf{T}))$ *cannot exceed* $d \cdot w^{(2k+1)n}$.

Let us now restrict our attention to alphabets Σ containing at least two elements. We may assume that $\{0, 1\} \subseteq \Sigma$.

We define the language

$$P = \{11t1100x\beta t^T : x \in \Sigma^*,\ t \in \Sigma^* - \Sigma^*11\Sigma^*,\ \beta \notin \Sigma\}.$$

Lemma 2. *The language P is real-time definable.*

In fact, P is definable by a real-time pushdown automaton (Ginsburg and Greibach [2]).

Our goal is to show that Σ^*P is not real-time definable. We accomplish this by examining the rate of growth of the index of $\mathrm{E}_k(\mathrm{mod}\, \Sigma^*P)$.

Definition 6. For all integers m, let X_m be the set of all tapes of length m over $\{0, 1\}$ such that no tape contains two consecutive 1's.

The Fibonacci numbers are defined thus: (1) $a_0 = a_1 = 1$; (2) for $n > 1$, $a_n = a_{n-1} + a_{n-2}$. It is well-known from the theory of finite difference equations that, for sufficiently large n, $a_n > (1.6)^n$.

Lemma 3. *For all* $m > 0$, *the number of tapes in* X_m, *denoted* $\aleph(m)$, *is precisely the* $(m + 1)$*-th Fibonacci number.*

Proof. The proof is by induction on m.

Base: $\aleph(1) = 2 = a_2$, and $\aleph(2) = 3 = a_3$, since $X_1 = \{0, 1\}$ and $X_2 = \{00, 01, 10\}$.

Hypothesis: Assume that, for $j \leq r$, $\aleph(j) = a_{j+1}$.

$r + 1$: The members of X_{r+1} are obtained from those of X_r as follows:

(1) If t is in X_r, then $0t$ is in X_{r+1}. This contributes $\aleph(r)$ tapes to X_{r+1}.

(2) If t is in X_r, then $1t$ is in X_{r+1} if, and only if, the first symbol of t is a 0. By step (1), this contributes precisely $\aleph(r-1)$ tapes to X_{r+1}.

Obviously, X_{r+1} is exhausted by steps (1) and (2), whence

$$\aleph(r+1) = \aleph(r) + \aleph(r-1) = a_{r+1} + a_r = a_{r+2}. \qquad \text{Q.E.D.}$$

Now, there are precisely $2^{\aleph(m)}$ subsets of X_m. We now show that each nonempty subset determines a distinct equivalence class of $E_{m+1}(\text{mod } \Sigma^*P)$.

Definition 7. For each tape t_i in X_m, we form the tape $\alpha_i = 11t_i1100$ of length $m + 6$.

For each nonempty subset $Y_i = \{t_1, \cdots, t_r\}$ of X_m $(1 \leq i \leq 2^{\aleph(m)} - 1)$ we form the tape $\delta_i = \alpha_1\alpha_2 \cdots \alpha_r$ of length $(m+6)r$.

LEMMA 4. *If the tape t is in $Y_i \subseteq X_m$, then $\delta_i\beta t^T$ is in Σ^*P.*

PROOF. Obvious from construction of δ_i.

Letting $k = m + 1$, we obtain

LEMMA 5. *Letting δ_i and Y_i be as in Definition 7, $\delta_i E_k \delta_j (mod\ \Sigma^*P)$ only if $Y_i = Y_j$.*

PROOF. Assume that $Y_i \neq Y_j$. Then there is a tape t in either $Y_i - Y_j$ or $Y_j - Y_i$. We may assume without loss of generality that there is a tape t in $Y_i - Y_j$.

By Lemma 4, $\delta_i\beta t^T$ is in Σ^*P. We claim that $\delta_j\beta t^T$ is not in Σ^*P.

Assume that $\delta_j\beta t^T$ were in Σ^*P. This would, by definition of P, imply that δ_j has a subtape of the form $11x1100$ such that x contains no two consecutive 1's and $x = t$. However, by construction, $\delta_j = 11t_11100 \cdots 11t_r1100$ where $Y_j = \{t_1, \cdots, t_r\}$. Thus the only subtapes of δ_j of the form $11x1100$ where $L(x) = m$ and x contains no two consecutive 1's are precisely those of the form $11t_a1100$ where t_a is in Y_j. Thus, if $\delta_j\beta t^T$ is in Σ^*P, then t is in Y_j, contrary to assumption.

We conclude that $\delta_j\beta t^T$ is not in Σ^*P if t is not in Y_j. Since $L(t) = m$, we have shown that $\delta_i E_k \delta_j (\text{mod } \Sigma^*P)$ only if $Y_i = Y_j$. Q.E.D.

Since each $\delta_i (1 \leq i \leq 2^{\aleph(m)} - 1)$ is thus in a different equivalence class of $E_k(\text{mod } \Sigma^*P)$, we have proved:

LEMMA 6. *The number of equivalence classes of $E_{k+1}(mod\ \Sigma^*P)$ is no less than $2^{\aleph(k)} - 1$.*

We are now in a position to show that Σ^*P is not in **R**.

THEOREM 6. *The class of real-time definable languages is not closed under concatenation.*

PROOF. Assume, for contradiction, that the n-RTTM **T** defines Σ^*P. By Lemma 1, the number of equivalence classes of $E_k(\text{mod } A(\mathbf{T}))$ cannot exceed $e_k = d \cdot w^{(2k+1)n}$. Now, for large k, $\aleph(k) = a_{k+1} > (1.6)^{k+1}$. Therefore, the number of equivalence classes of $E_k(\text{mod } \Sigma^*P)$ satisfies

$$2^{(1.5)^k} \leq 2^{\aleph(k-1)} - 1 \leq \text{index of } E_k(\text{mod } \Sigma^*P)$$

for large k. Since $2^{(1.5)^k} > e_k$, for large k, **T** cannot define Σ^*P contrary to assumption. Thus Σ^*P is not in **R**.

We have thus found two RTDL's, Σ^* and P, whose concatenation is not a RTDL. Q.E.D.

Since Σ^* is a regular set, we have, in fact, proved

Proposition 1. The class of real-time definable languages is not closed under concatenation with regular sets.

4.2 KLEENE CLOSURE OPERATOR. Using the results of Section 4.1, we show that **R** is not closed under the operation of closure. The proof will follow from the following general result.

Let $\mathcal{C}$ be a class of languages. We say that $\mathcal{C}$ is *admissible* if

(a) $\mathcal{C}$ is closed under union and intersection with regular sets;

(b) there is a language L, over an alphabet Σ, in $\mathcal{C}$ such that, if δ is a symbol not in Σ, then $L\{\delta\}$ is in $\mathcal{C}$ while $\Sigma^*L\{\delta\}$ is not.

THEOREM 7. *If $\mathcal{C}$ is an admissible class of languages, then $\mathcal{C}$ is not closed under the operation of closure.*

PROOF. Let L be a member of $\mathcal{C}$ which satisfies part (b) of the definition of admissible; and let δ be a symbol not in the alphabet Σ over which L is encoded.

Since Σ^* is a regular set, it follows that the language $H = L\{\delta\} \cup \Sigma^*$ is in $\mathcal{C}$. We claim that H^* is not in $\mathcal{C}$.

Assume, for contradiction, that H^* were in $\mathcal{C}$. Since $\Sigma^*\{\delta\}$ is a regular set, it would follow that $J = H^* \cap \Sigma^*\{\delta\}$ were in $\mathcal{C}$. However, one can easily see that $J = \Sigma^*L\{\delta\}$ which is known not to be in $\mathcal{C}$. We conclude that $\mathcal{C}$ is not closed under closure. Q.E.D.

LEMMA 7. **R** *is an admissible class of languages.*

PROOF. That **R** is closed under union and intersection with regular sets is a corollary of Theorem 1.

To verify part (b) of the definition of admissible, consider the language P of Section 4.1. Since **R** is closed under suffixing with regular sets, it follows that $P\{\delta\}$ is in **R** for any symbol δ. Moreover, the proof of Theorem 6 shows that $\Sigma^*P\{\delta\}$ is not in **R**. (The terminal δ does not affect the proof.) Q.E.D.

Since **R** is admissible, we obtain

THEOREM 8. *The class of real-time definable languages is not closed under the operation of closure.*

Remark. The proofs of Theorems 6 and 8 serve to demonstrate that the class of languages defined by real-time pushdown automata is not closed under concatenation or closure.

4.3 REVERSAL. In this section we prove that **R** is not closed under reversal. As a corollary to this proof, we show that there are languages which are definable in time $2L(t)$ (and hence, by Corollary 3.1, in time $(1 + \epsilon)L(t)$), but not in real time.

The language Q defined as follows is a RTDL, while its reversal is not.

Q is the set of sequences of the form

$$t\beta 0011t_1110011t_211 \cdots 0011t_k11$$

where (1) $t, t_1, \cdots, t_k$ are in X_m for some m, and $\beta \in \{0, 1\}$, and (2) for at least one $i \in \{1, \cdots, k\}$, $t_i = t^T$, the reversal of the initial segment t.

LEMMA 8. *The language Q is a RTDL.*

PROOF. We sketch the operation of a 2-RTTM **T** which defines Q. The interested reader can easily fill in the details.

The operation of **T** proceeds as follows:

(1) While reading the initial segment t of the input, **T** copies t on each of its work tapes, verifying that t contains no two consecutive 1's.

(2) As t_1 comes in, **T** checks, on tape 1, that $t_1 = t^T$. If $L(t_1) \neq L(t)$, **T** enters a dead state; if $t_1 = t^T$, **T** proceeds to step (4); if $t_1 \neq t^T$, **T** proceeds to step (3).

(3) As t_{2i} (resp., t_{2i+1}) comes in, **T** checks, on tape 2 (resp. tape 1), that t_{2i} (resp. t_{2i+1}) $= t^T$, simultaneously restoring tape 1 (resp., tape 2) so that the read-write head is on the rightmost square. If $L(t_{2i})$ (resp., $L(t_{2i+1})$) $\neq L(t)$, **T** enters a dead state; if t_{2i} (resp., t_{2i+1}) $= t^T$, **T** proceeds to step (4); otherwise, **T** repeats step (3).

(4) Having found a $t_i = t^T$, **T** checks, on tape 1 that all subsequent t_j are of length $L(t)$. **T** enters an accepting state whenever it finishes scanning a segment t_i11 where $L(t_i) = L(t)$.

The tape situations of **T** are illustrated in Figure 1. It is clear that **T** defines Q in real time. Q.E.D.

THEOREM 9. *The class of real-time definable languages is not closed under reversal.*

PROOF. Note that Q^T is a subset of $\Sigma^* P$ from Section 4.1. The proof of Theorem 6 actually shows that Q^T is not a RTDL. The nonclosure of **R** under reversal thus follows from Lemma 8 and the proof of Theorem 6. Q.E.D.

LEMMA 9. *The language Q^T is definable in time $2L(t)$.*

PROOF. A 3-TM defining Q^T merely copies the input on one of its tapes, and then proceeds to simulate the 2-RTTM **T** of Lemma 8. The 3-TM rejects any tape

$$11t_1110011t_211\cdots 11t_k1100\beta t$$

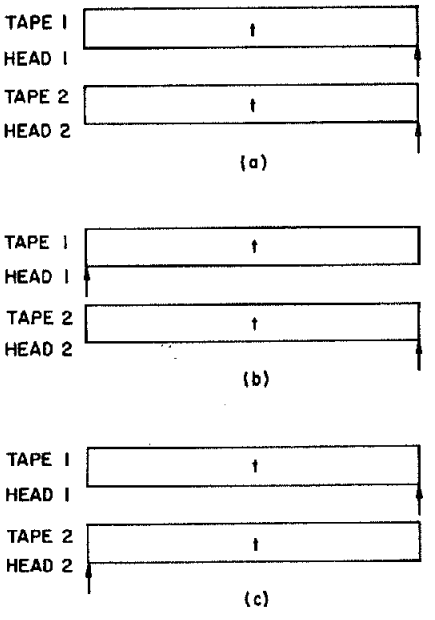

FIG. 1. (a) The tapes of **T** after reading t, (b) the tapes of **T** after reading t_{2i-1}, (c) the tapes of **T** after reading t_{2i}.

such that $L(t) \neq L(t_1)$—this is how it determines the end of the input tape. Once more, the details are left to the reader. Q.E.D.

We thus obtain:

Proposition 2. There is a language which is definable in time $2L(t)$, and hence, in time $(1 + \epsilon)L(t)$ for any $\epsilon > 0$, which is not real-time definable.

4.4 Mapping Devices. The results of Section 4.1 afford us an easy proof of the nonclosure of **R** under mapping by various devices.

A sequential machine is a 0-RTT $(Q, \Sigma, \Delta, M, R, q_0)$ such that R (the output function) maps $Q \times \Sigma \to \Delta$.

Let us recall the language P of Section 4.1. P is a language over the alphabet $\Sigma = \{0, 1, \beta\}$. Therefore, if we let $\Delta = \{a, b\}$ where a and b are not in Σ, it is obvious that Δ^*P is a RTDL.

Consider now the sequential machine $\mathbf{U} = (\{q_0\}, \Sigma \cup \Delta, \Sigma, M, R, q_0)$ where (1) for σ in $\Sigma \cup \Delta$, $M(q_0, \sigma) = q_0$, (2) $R(q_0, a) = R(q_0, 0) = 0$, $R(q_0, b) = R(q_0, 1) = 1$, $R(q_0, \beta) = \beta$.

Clearly $\mathbf{U}(\Delta^*P) = \{0, 1\}^*P$ which we know not to be in **R**.

We thus have shown,

Theorem 10. *The image of a real-time definable language under sequential machine mapping need not be real-time definable.*

Corollary 10.1. *The image of a RTDL under generalized sequential machine mapping need not be a RTDL.*

Corollary 10.2. *The homomorphic image of a RTDL need not be a RTDL.*

Proof. A homomorphism can be effected by a one-state gsm.

Corollary 10.3. *The image of a RTDL under real-time transduction need not be a RTDL.*

Finally, we prove that the inverse image of a RTDL under *nondeterministic* sequential machine mapping need not be real-time definable.

A nondeterministic sequential machine is a 6-tuple $\mathbf{U} = (Q, \Sigma, \Delta, M, R, Q_0)$ where Q is a finite set of states, Σ is the input and Δ the output alphabet, M is a map from $Q \times \Sigma \to 2^Q$, R is a function from $Q \times \Sigma \to \Delta^*$, and $Q_0 \subseteq Q$ is the set of initial states. **U** operates in the obvious way. The image, with R extended to $Q \times \Sigma^*$, $\mathbf{U}(x) = \bigcup_{q \in Q_0} R(q, x)$ of a tape x under **U** is now a set of tapes: precisely that set obtained by following all paths **U** describes (via the M-mapping) under input x.

Consider now the nondeterministic sequential machine **U** defined as follows: $\Sigma = \{0, 1, \beta\}$, $\Delta = \{a, 0, 1, \beta\}$, $Q_0 = \{q_0, q_1\}$.

M	0	1	β
q_0	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_3\}$
q_1	$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_3\}$
q_2	$\{q_2\}$	$\{q_2\}$	$\{q_3\}$
q_3	$\{q_3\}$	$\{q_3\}$	$\{q_3\}$

R	0	1	β
q_0	a	a	β
q_1	0	1	β
q_2	a	a	β
q_3	0	1	β

The language $L = \{utv\beta t^r : t \in \{0, 1\}^*; u, v \in \{a\}^*; \beta \notin \{0, 1, a\}\}$ is a RTDL. The inverse image of L under **U** is $\mathbf{U}^{-1}(L) = \{x : \mathbf{U}(x) \cap L \neq \phi\} = \{xty\beta t^r : x, t, y \in \{0, 1\}^*, \beta \notin \{0, 1\}\}$. The proof of Theorem 6 in Section 4.1 suffices to show that $\mathbf{U}^{-1}(L)$ is not in **R**.

Proposition 3. The inverse image of a RTDL under nondeterministic sequential machine mapping need not be real-time definable.

4.5 Derivatives and Quotients. In this section we prove that neither the derivative[5] nor the quotient of a RTDL by a regular set need be real-time definable.

Definition 8. Let A and B be languages. The *derivative* of A w.r.t. B, denoted $D_B(A)$, is the set

$$D_B(A) = \{y : xy \text{ is in } A \text{ for some } x \text{ in } B\}.$$

In the terminology of Ginsburg and Spanier [3], $D_B(A)$ is the *left quotient* of A by B and is denoted $B\backslash A$.

Recalling the language Q of Section 4.3, we consider the set V of sequences of the form

$$t\beta 11t_1 1100 11t_2 1100 \cdots 11t_k 1100\beta t$$

where (1) for some m, t, t_1, $\cdots$, t_k are in X_m, and β is not in $\{0, 1\}$, and (2) for at least one $i \in \{1, \cdots, k\}$, $t_i = t^T$.

Lemma 10. *The language V is a RTDL.*

Proof. A 2-RTTM which defines V operates almost identically to the 2-RTTM which defines Q (Lemma 8) except that upon encountering the second β it checks that the terminal string is identical to the initial string. A glance at Figure 1(b) and (c) indicates that this is always possible. Q.E.D.

Theorem 11. *The derivative of a RTDL with respect to a regular set need not be real-time definable.*

Proof. By Lemma 10, V is a RTDL; the set $A = \{0, 1\}^*\{\beta\}$ is regular. However $D_A(V) = Q^T$ which we showed not to be real-time definable (Theorem 9). Q.E.D.

Now, drawing on the results of Section 4.1, we show that the quotient of a set in **R** by a regular set need not be real-time definable.

Definition 9. Let A and B be sets of tapes. The *quotient of A by B*, denoted A/B, is the set $A/B = \{x : xy \text{ is in } A \text{ for some } y \text{ in } B\}$.

Ginsburg and Spanier [3] refer to A/B as the *right quotient* of A by B.

Recalling the sets P and Σ^*P of Section 4.1, we define R to be the set of all tapes of the form

$$x11t1100y\beta t^T \delta a^{L(y)+L(t)+6} \delta a^{L(t)+6}$$

where x, y are in Σ^*, t is in $\Sigma^* - \Sigma^*11\Sigma^*$, a is a member of Σ, a^k denotes a string of k a's, and β, δ are not in Σ.

Lemma 11. *The language R is definable by a 2-RTTM.*

Proof. As in the previous lemma, we informally describe the operation of the 2-RTTM **T** which defines R.

(1) **T** copies the input symbols onto tape 1 until it encounters the symbol β. It then stores the next input symbols on tape 2 until it encounters the first δ. At this point, the read-write head of tape 1 (resp. 2) is on the rightmost symbol of a copy of $x11t1100y$ (resp. t^T).

(2) While receiving the string $a^{L(t)+L(y)+6}$, **T** moves tapes 1 and 2 so that, by the time the second δ is encountered, the read-write head of tape 2 is on the leftmost nonblank symbol of tape 2, and the read-write head of tape 1 is on some symbol of

[5] The term "derivative" was, to the author's knowledge, introduced by Brzozowski, Derivatives of regular expressions, *J. ACM 11* (1964), 481–494.

the segment $x11t1100y$. Note, if the input tape is, indeed, in R, the read-write head of tape 1 will be on the leftmost symbol of the segment $11t1100y$.

(3) While receiving the terminal sequence $a^{L(t)+6}$, **T** checks that tape 2 contains t^T, where the initial segment of length $L(t) + 6$ to the right of the head on tape 1 is $11t11C0$. **T** enters an accepting state if this last condition is fulfilled.

Clearly, $A(\mathbf{T}) = R$. Q.E.D.

As the reader can readily guess, we now consider the regular set $A = \{\delta\}\{a\}^*\{\delta\}\{a\}^*$. It is clear that $R/A = \Sigma^*P$ which we showed (Theorem 6) not to be real-time definable.

Since R is in **R**, and A is a regular set, while R/A is not in **R**, we have proved

THEOREM 12. *The quotient of a real-time definable language by a regular set need not be real-time definable.*

4.6 MAXIMIZATION. The final operation we shall consider is maximization. Let L be a language; then,

$$\max L = \{x : x \in L, \text{ and if } x < y, \text{ then } y \notin L\}.$$

THEOREM 13 (A. R. Meyer[6]). *There is a RTDL L such that max L is not real-time definable.*

PROOF. We assume familiarity with the concepts of universal Turing machine and the arithmetization of Turing machines.

Let us consider an arithmetization of Turing machines under which every natural number x corresponds to a Turing machine. We assume the integers are encoded in binary.

We are given a universal TM **T**, and we define the language H as follows. H is the set of all sequences of the form $x\beta z$ where

(1) x is in $\{1\}\{0, 1\}^*$, β is not in $\{0, 1\}$, z is in $\{0\}^*$;

(2) the universal TM **T** discovers in at most $L(z) - L(x)$ steps that the TM corresponding to x considered as a binary integer, halts when applied to input x. Symbolically, **T** verifies in no more than $L(z) - L(x)$ steps the predicate $(\exists n)T(x, x, n)$.[7]

The language L is now defined as

$$L = \{1\}\{0, 1\}^*\{\beta\} \cup H.$$

Now H is, by definition, definable by a 1-RTTM.[8] A 1-RTTM copies the input string on its work tape until it encounters the symbol β. It then backspaces on its work tape, one square for each incoming 0 until it reaches the leftmost end of x. From this point on, it simulates one step of the universal TM **T** for each incoming 0. The 1-RTTM enters an accepting state when it finds that **T** would halt with a YES answer on input x, remaining in this accepting state as long as it receives 0 inputs.

Since H is a RTDL, it follows from Theorem 1 that L is a RTDL. However, one readily verifies that

$$\max L = \{x\beta : x \in \{1\}\{0, 1\}^*, \beta \notin \{0, 1\} \quad \text{and} \quad (\forall n) \backsim T(x, x, n)\},$$

[6] Private communication

[7] The predicate $T(x, y, n)$ means that TM x, applied to input y, halts in n steps.

[8] By considering the language $H/\{\beta\}\{0\}^*$, one obtains an alternative proof of Theorem 12. In fact, one obtains the strengthened result that the quotient of a set in **R**(1) by a regular set need not be in R.

which is obviously not a recursive set. Since every RTDL is context-sensitive and, therefore, recursive, we conclude that $\max L$ is not real-time definable. Q.E.D.

4.7 INVARIANCE OF THE RESULTS. In this section we note that the negative results obtained in the previous sections are valid for a large class of machines, including almost all the models which have been used to study real-time computation. As a corollary of this observation, we obtain a proof that, for all n, the class of languages defined by n-dimensional iterative arrays of finite automata (Cole [1]) is not closed under the Kleene closure operator.

We recall the definition of the relation $E_k(\text{mod } A)$ (Definition 5).

Definition 10. A class $\mathfrak{M}$ of machines is of *polynomial-limited accessibility* if, for any machine $\mathbf{M}$ in $\mathfrak{M}$, there exist constants c and n such that, for $k \geq 1$, the index of $E_k(\text{mod } A(\mathbf{M}))$ does not exceed c^{k^n}.

We mention several examples of polynomial-limited access classes of machines:

(1) the class of real-time multitape Turing machines;

(2) for any n, the class of real-time multihead TM's with n-dimensional tapes.

(3) for any n, the class of real-time n-dimensional iterative arrays of finite automata.

A perusal of the proofs in Section 4 suffices to show:

THEOREM 14. *Let $\mathfrak{M}$ be any class of polynomial-limited accessibility machines.*

(a) *If the class of languages, $L(\mathfrak{M})$, defined by $\mathfrak{M}$-machines includes the class of languages defined by real-time PDA's, then $L(\mathfrak{M})$ is not closed under concatenation, even with regular sets, under closure, not under sequential machine mapping.*

(b) *If the class of languages defined by $\mathfrak{M}$-machines includes the class defined by 2-RTTM's (which, perforce, subsumes case* (a)*), then $L(\mathfrak{M})$ is not closed under reversal, nor under the operations of taking derivatives and quotients.*

One may note that the languages defined by any of the classes of machines mentioned above are all recursive. They are, thus, not closed under maximization, by Section 4.6.

4.8 POSITION OF **R** IN THE LINGUISTIC HIERARCHY. In Section 2 we noted that **R** is a subclass of the context-sensitive languages. The results of Sections 3 and 4 indicate that **R** is incomparable to the CFL's; that is, there are CFL's (such as Σ^*P of Section 4.1) which are not real-time definable, and there are RTDL's (such as $\{a^nb^nc^n : n \geq 1\}$) which are not context-free. In this section we exhibit a *deterministic* CFL which is not in **R**. This will verify Figure 2 which indicates the position of **R** in the classical linguistic hierarchy.

Let L be the set of all sequences

$$0^{k_1}10^{k_2}1 \cdots 0^{k_{r-1}}10^{k_r} @^s 0^{k_{r-s+1}}$$

where (1) $@$ is not 0 or 1;
(2) $r \geq 1$, and $k_i \geq 1$ $(1 \leq i \leq r)$;
(3) $1 \leq s \leq r$.

One can easily verify that L is a deterministic CFL. We show L not to be in **R** by considering the index of $E_p(\text{mod } L)$ as p grows.

NOTE. Cole [1] and the author [10] have independently found another example of a deterministic CFL which is not in **R**. We consider L because of its simplicity and intrinsic interest.

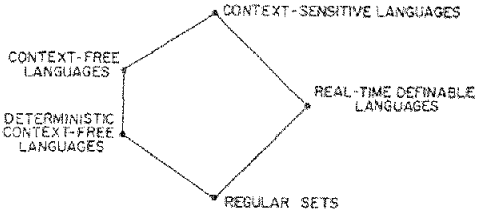

FIG. 2. The position of **R** in the linguistic hierarchy

THEOREM 15. *There is a deterministic CFL which is not real-time definable.*

PROOF. Assume that the language L were defined by an n-RTTM **T** with d internal states and w working symbols.

Consider those subsets $L_1, L_2, \cdots$ of L such that, for each integer m, L_m is the set of all sequences

$$0^{k_1}10^{k_2}1 \cdots 0^{k_{r-1}}10^{k_r} @^s 0^{k_{r-s+1}}$$

in L with $k_i \leq m$ $(1 \leq i \leq r)$.

For given r, one readily verifies that the number of equivalence classes of $E_{m+r}(\text{mod } L)$ is no less than m^r. For, consider distinct initial sequences

$$t_1 = 0^{h_1}10^{h_2}1 \cdots 0^{h_r}, \qquad 1 \leq h_1, \cdots, h_r \leq m,$$

and

$$t_2 = 0^{k_1}10^{k_2}1 \cdots 0^{k_r}, \qquad 1 \leq k_1, \cdots, k_r \leq m.$$

There clearly is a sequence $u = @^a 0^b$, with $a + b \leq m + r$, such that t_1u is in L_m, and hence in L, while t_2u is not in L_m, and hence not in L by definition of L_m.

Now, by Lemma 1, we know that the number of equivalence classes of $E_{m+r}(\text{mod } A(\mathbf{T}))$ cannot exceed $d \cdot w^n \cdot w^{2n(m+r)}$. However, it is easy to verify that, for sufficiently large r and m, $m^r > c^{m+r}$ for any constant c.

It follows that **T** cannot define L, contradicting our assumption. Q.E.D.

5. *Decision Problems*

We briefly mention several problems concerning real-time definability which are recursively unsolvable. The proofs follow readily from the unsolvability of Post's Correspondence Problem [7].

THEOREM 16. (a) *The problem of deciding whether or not an arbitrary language is a RTDL is recursively unsolvable.*

(b) *The following problems are recursively unsolvable for RTDL's X and Y:*

(i) *Is X equal to ϕ?*

(ii) *Is X equal to Σ^*?*

(iii) *Is X context-free?*

(iv) *Is X regular?*

(v) *Are X and Y disjoint?*

(vi) *Is $X \subseteq Y$?*

(vii) *Is X equal to Y?*

ACKNOWLEDGMENT. The author gratefully acknowledges the many helpful suggestions and criticisms of Mr. A. R. Meyer.

REFERENCES

1. COLE, S. N. Real-time computation by iterative arrays of finite-state machines. Doctoral thesis, Rep. BL-36 by Staff of Harvard Computation Lab. to Bell Telephone Labs., 1964.
2. GINSBURG, S., AND GREIBACH, S. A. Deterministic context free languages. IEEE Conference Record on Switching Circuit Theory and Logical Design, IEEE Pub. 16C13, 1965, pp. 203–220.
3. ——, AND SPANIER, E. H. Quotients of context free languages. *J. ACM 10* (1963), 487–492.
4. ——, AND ULLIAN, J. Ambiguity in context free languages. *J. ACM 13* (1966), 62–89.
5. HARTMANIS, J., AND STEARNS, R. E. On the computational complexity of algorithms. *Trans. Amer. Math. Soc. 117*, 5 (May 1965), 285–306.
6. KURODA, S. Classes of languages and linear-bounded automata. *Inform. and Contr. 7* (1964), 207–223.
7. POST, E. A variant of a recursively unsolvable problem. *Bull. Amer. Math. Soc. 52* (1946), 264–268.
8. RABIN, M. O. Real time computation. *Israel J. Math. 1* (1963), 203–211.
9. ROSENBERG, A. L. On the nonclosure of the real-time definable sets under concatenation, closure, and reversal. IBM Res. Rep. RC-1572, 1966.
10. ——. A remark on real-time definable languages. IBM Res. Note NC-616, 1966.

RECEIVED NOVEMBER, 1966; REVISED MARCH, 1967