A COMPARISON OF THE WORK DONE BY GENERALIZED SEQUENTIAL MACHINES AND TURING MACHINES

BY

SEYMOUR GINSBURG AND GENE F. ROSE

Introduction

For a long time it has been accepted that Turing machines are more powerful than generalized sequential machines. Since a generalized sequential machine is also a Turing machine, the class of Turing machines is at least as powerful as the class of generalized sequential machines. The purpose of this paper is to examine the adverb "more" from the aspect of work accomplished. Three manifestations of work performed by machines are considered, namely (i) numerical functions, (ii) output tapes as a function of input tapes, and (iii) the sets of accepted tapes. For each of these expressions of work there is studied the question of the existence of codes so that the resulting set (i.e., the set of numerical functions, the set of input-to-output tape functions, and sets of accepted tapes respectively) for the class of Turing machines is transformed into a subset of the resulting set for the class of generalized sequential machines. In other words, the problem under investigation is to see if the behavior of Turing machines can be reinterpreted as behavior of generalized sequential machines.

The main results achieved are as follows.

(1) There exist effective mappings f and g, f transforming a subset of the natural numbers into sequential input tapes and g transforming a set of sequential output tapes into the natural numbers, with the following property. To each partial recursive function h there corresponds a generalized sequential machine S so that for each natural number x, $g(S(f(x)))=h(x)$, where $S(f(x))$ is the output from machine S upon application of the input tape $f(x)$. Furthermore, all these machines S have a finite joint input alphabet and a finite joint output alphabet (Theorem 3).

(2) Given any set of Turing machines in a finite joint alphabet there is a function f such that (a) f maps all Turing tapes one to one into sequential input tapes, and (b) for each Turing machine Z there is a generalized sequential machine S such that $S(f(X))=Z(X)$, X being any Turing tape, $S(f(X))$ being as above, and $Z(X)$ being the analogous function to Z that $S(f(X))$ is to S (Theorem 5).

(3) Given any set of Turing machines there is a function f such that (a) f maps all Turing tapes one to one into sequential tapes; and (b) for each Turing machine Z there is a sequential machine S such that the set of tapes

Received by the editors April 25, 1961.

accepted by Z maps, under f, onto the set of tapes accepted by S (Theorem 6).

1. **Numerical functions.** One well-known measure of work performed by a Turing machine is its associated singulary (i.e., one variable) partially computable function. For comparative purposes it is therefore natural to associate with each sequential machine a singulary function defined in an analogous manner. This we shall do. First though let us recall some preliminary concepts.

It is assumed that the reader is familiar with the basic concepts of Turing machines. The particular formulation and related theory used here is that of [1] with the following additions. A (Turing) tape is either (i) a single symbol of the alphabet (which may be B, the blank symbol) or (ii) a finite, nonempty sequence of symbols from the alphabet that neither begins nor ends with B. For each Turing machine Z (always assumed to have a start state, say q_1) let $Z(X)$ be the following partially defined function from tapes to tapes. For any tape X, $Z(X)$ is defined if and only if the resultant of q_1X exists, in which case $Z(X)$ is the largest tape contained as a subsequence in the resultant.

A generalized sequential machine S (with a start state) is a 6-tuple $(K_S, \Sigma_S, \Delta_S, q_1, \delta_S, \lambda_S)$, where

(i) K_S is a finite, nonempty set (of "states");

(ii) Σ_S is a finite, nonempty alphabet (of "inputs")(1);

(iii) Δ_S is a finite, nonempty alphabet (of "outputs")(1);

(iv) q_1 is an element of K_S (the "start" state);

(v) δ_S (the "state" function) is a mapping of a subset of $K_S \times \Sigma_S$ into K_S; and

(vi) λ_S (the "output" function) is a mapping of a subset of $K_S \times \Sigma_S$ into the free semi-group (without identity) generated by Δ_S. That is, for certain pairs (q, I) of states q and inputs I, $\lambda_S(q, I)$ is a sequence of elements of Δ_S.

If, for each state q and each input I, $\lambda_S(q, I)$ is either undefined or is an element of Δ_S, then S is called a sequential machine.

An input (output) tape is a sequence of elements of $\Sigma_S(\Delta_S)$.

Let $I_1 \cdots I_k$ be an input tape to the generalized sequential machine S. $\lambda_S(p_1, I_1 \cdots I_k)$ is said to exist if $p_{i+1} = \delta_S(p_i, I_i)$ exists for each $i<k$; and $\lambda_S(p_1, I_1 \cdots I_k)$, when it exists is defined to be the output tape

$$(*) \qquad \lambda_S(p_1, I_1)\lambda_S(p_2, I_2) \cdots \lambda_S(p_k, I_k),$$

it being understood that each $\lambda_S(p_i, I_i)$ which does not exist does not occur in the sequence (*)(2).

A singulary numerical function is associated with a Turing machine as

(1) Because of the special nature of the symbol B, except for Theorem 5, it will always be assumed that B is not an element of $\Sigma_S \cup \Delta_S$.

(2) In case S is a sequential machine this definition of $\lambda_S(p_1, I_1 \cdots I_k)$ differs from that given in [2].

follows. Let N denote the non-negative integers. N is coded into the Turing tapes by a function f, and the set of Turing tapes is coded into N by a function g. For each Turing machine Z there is then determined a function $g(Z(f(x)))$ which maps a subset of N into N. The particular functions chosen for f and g in [1] and used here are f^* and g^*, where f^* maps each non-negative integer x into the tape 1^{x+1}, and g^* maps each tape Y into the number of 1's in Y.

Clearly there is an analogous procedure for associating singulary numerical functions with generalized sequential machines given a pair (f, g) of functions such that f maps a subset of N into a set of input tapes and g maps a set of output tapes into N. Any such pair (f, g) of functions will be called an *interpretation*. For any interpretation (f, g) and any generalized sequential machine S, the singulary function $g(S(f(x)))$ (as a function of x) will be called the (*singulary*) numerical function associated with S under (f, g)(3).

With respect to associated numerical functions, Turing machines and generalized sequential machines exhibit similarities as well as differences. Under the particular interpretation (f^*, g^*), the function associated with any generalized sequential machine is a monotonically (where defined) non-decreasing, effectively calculable function. Hence the numerical functions associated with any class of generalized sequential functions, under (f^*, g^*), form a proper subclass of the partially computable functions. On the other hand, in view of Theorem 1 below, there is an interpretation under which the numerical functions associated with the complete sequential machines(4), the numerical functions associated with all generalized sequential machines, and the partially computable functions are the same. Moreover, this interpretation (f, g) is effective, i.e., for both f and g, there are algorithms that give, for any argument, their values (if they exist). All partially computable functions are computable by Turing machines in a finite joint alphabet(5). In contrast, by Theorem 2 below, for no set of complete sequential machines in a finite joint output alphabet do the associated numerical functions even include the computable functions. By Theorem 3 below, however, there is a set of generalized sequential machines with finite joint output alphabet such that the associated numerical functions are exactly the partially computable functions.

THEOREM 1. *There is an effective interpretation under which the numerical functions associated with the complete sequential machines, as well as the func-*

(3) For any generalized sequential machine S, $S(X)$ is the following partially defined function with input tapes as arguments and output tapes as values. For any input tape X, $S(X) = \lambda_S(q_1, X)$ where q_1 is the start state of S.

(4) A complete sequential machine S is a sequential machine in which $\lambda_S(q, I)$ and $\delta_S(q, I)$ are defined for every state q and every input I.

(5) For a set of Turing machines, the joint alphabet is the union of the alphabets of the various machines in the set. For a set of generalized sequential machines, the joint input alphabet and the joint output alphabet are similarly defined.

tions associated with all generalized sequential machines, are exactly the partially computable functions.

Proof. Let $f(x) = I^{x+1}$, $g(X) = U(\min_y T_1(z, x, y))$ if $X = E_z^{x+1}$ and if $U(\min_y T_1(z, x, y))$ is defined, and let g be otherwise undefined. Here I is a particular input and $E_0, E_1, \cdots$ are particular distinct outputs; U is the particular computable singulary function and T_1 the particular computable ternary predicate defined in [1]. Clearly (f, g) is an effective interpretation. Consequently the function $g(S(f(x)))$ associated with any generalized sequential machine S is partially computable. Thus the set of functions associated with the complete sequential machines is a subset of the set of functions associated with all generalized sequential machines; which, in turn, is a subset of the set of all partially computable functions. The proof is now completed by describing, for each partially computable function c, a complete sequential machine with which c is associated under the given interpretation.

To this end, given c, let z be a number such that $c(x) = U(\min_y T_1(z, x, y))$ (cf. [1]); let S be the complete sequential machine with $K_S = \{q\}$ (where q is a particular state), $\Sigma_S = \{I\}$, $\Delta_S = \{E_z\}$, $\delta_S(q, I) = q$, and $\lambda_S(q, I) = E_z$. Then $g(S(f(x))) = g(\lambda_S(q, f(x))) = g(\lambda_S(q, I^{x+1})) = g(E_z^{x+1}) = U(\min_y T_1(z, x, y)) = c(x)$. Therefore c is associated with S.

Remark. The proof of Theorem 1 is easily altered to yield the following result. For each interpretation (f, g) (with respect to Turing tapes) there exists an interpretation (f_1, g_1) (with respect to input tapes and output tapes) having this property. To each function h associated with a Turing machine under (f, g) there is a complete sequential machine S so that $h = g_1(S(f_1(x)))$.

Theorem 2. *Under any interpretation, the set of functions associated with any set of complete sequential machines in a finite joint output alphabet*(5) *does not include all computable functions (and consequently does not include all partially computable functions).*

Proof. Assume that the theorem is false, i.e., that there is an interpretation (f, g) and a set M of complete sequential machines such that all computable functions are associated with M. Let $c_0, c_1, \cdots$ be the computable constant functions with respective values $0, 1, \cdots$. Then each function c_i is associated with a machine S_i in M. Now $g(S_0(f(0))) = 0$, so that $f(0)$ is defined. Let d be the length of the input tape $f(0)$. Then each $S_i(f(0))$ is one of the finitely many output tapes of length d in the finite joint output alphabet. It follows that $\{g(S_0(f(0))), g(S_1(f(0))), \cdots\}$ is finite, contrary to the condition (imposed by the associated functions) that this set is $\{0, 1, \cdots\}$. Hence the result.

Remark. It is easily seen that Theorem 2 may be extended by removing the word "complete."

Theorem 3. *There is an effective interpretation and a set of generalized sequential machines in a finite joint alphabet such that the associated functions are exactly the partially computable functions.*

Proof. Let $f(x)=I^{x+1}$, $g(X)=U(\min_y T_1(z, x, y))$ if $X=(E_1^{z+1}E_2)^{x+1}$ and $U(\min_y T_1(z, x, y))$ is defined, and let g be otherwise undefined. Here I is a particular input and E_1 and E_2 are particular distinct outputs. Clearly (f, g) is an effective interpretation. Consequently any function associated with a generalized sequential machine is partially computable. The proof is now completed by describing, for each partially computable function c, a generalized sequential machine S with which c is associated under the interpretation. To this end, given c, let z be a number such that $c(x)=U(\min_y T_1(z, x, y))$. Let S be the generalized sequential machine with $K_S=\{q\}$, $\Sigma_S=\{I\}$, $\Delta_S=\{E_1, E_2\}$, $\delta_S(q, I)=q$, and $\lambda_S(q, I)=E_1^{z+1}E_2$. Then $g(S(f(x)))=g(\lambda_S(q, f(x)))=g(\lambda_S(q, I^{x+1}))=g((E_1^{z+1}E_2)^{x+1})=U(\min_y T_1(z, x, y))=c(x)$. Therefore c is associated with S.

2. **Input tapes versus output tapes.** The work done by a given Turing or generalized sequential machine will now be represented by the corresponding function $Z(X)$ or $S(X)$. A Turing tape will be regarded as an "input tape" or an "output tape" according to its role as argument or value for $Z(X)$.

It is readily shown that for any generalized sequential machine S there is a Turing machine Z such that $S(X)=Z(X)$ for any input tape X. On the other hand, if S is a generalized sequential machine then the length of $S(X)$ never exceeds the length of $S(X')$ if X' is a continuation of X, whereas $Z(X)$ is not thus restricted. In a sense, then, Turing machines as a class "do more work" than generalized sequential machines. It is not precluded, however, that by suitable codings of Turing tapes into sequential tapes, all Turing functions $Z(X)$ might be reinterpreted as generalized sequential functions $S(X)$. More precisely, for all Turing machines in a given joint alphabet, it is proposed to map all Turing input tapes one to one into sequential input tapes, and all Turing output tapes one to one into sequential output tapes. Can such mappings f and g be found so that for each Turing machine Z there is a (generalized) sequential machine S so that $S(f(X))=g(Z(X))$? Theorems 4 and 5 below show that no such mappings can be found with both f and g effectively calculable, but that there are such mappings for the generalized sequential machine case with a nonconstructive f. For the sequential machine case there are no such mappings f and g as the following analysis shows.

Assume that there is a joint Turing alphabet for which such f and g exist. Let Δ be any finite subset of the given joint alphabet. Then there is a Turing machine Z such that $Z(X)=X$ for all tapes X in Δ. For the sequential machine S associated with Z, $S(f(X))=g(Z(X))=g(X)$ for all X in Δ. Hence g maps all Turing output tapes in Δ into tapes in the finite output alphabet Δ_0 of S. Now consider a fixed Turing input tape X_0 in Δ. For each output tape Y_i in Δ there is a Turing machine Z_i such that $Z_i(X_0)=Y_i$. For the associated sequential machine S_i, $S_i(f(X_0))=g(Z_i(X_0))=g(Y_i)$. Now $S_i(f(X_0))$ is one of the finitely many output tapes in Δ_0 with length no greater than that of $f(X_0)$. Consequently the set $\{g(Y_i)\mid i\geqq 1\}$ is finite. Since $\{Y_i\mid i\geqq 1\}$ is infinite and g is one to one, $\{g(Y_i)\mid i\geqq 1\}$ is infinite, a contradiction

THEOREM 4. *For any finite Turing alphabet there is no pair* (f, g) *of effectively calculable one to one mappings with the following properties*: (1) f *maps all Turing input tapes into sequential input tapes*; (2) g *maps all Turing output tapes into sequential output tapes*; *and* (3) *for each Turing machine* Z, *there is a generalized sequential machine* S *such that* $S(f(X))=g(Z(X))$.

Proof. Assume the contrary, that is, that there is a finite Turing alphabet Δ for which effectively calculable f and g exist. Let $X_1, X_2, \cdots$ be an effective enumeration, without repetition, of all Turing tapes, and let $S_1, S_2, \cdots$ be an effective enumeration of all generalized sequential machines in an arbitrary joint alphabet. Define a function h as follows. For each Turing tape X_i, $h(X_i)$ is the first X_j (which exists since g is one to one) such that $g(X_j)\neq S_i(f(X_i))$ if $S_i(f(X_i))$ is defined, and $h(X_i)=X_i$ otherwise. In view of the solvability of the halting problem for generalized sequential machines, h is effectively calculable. Therefore there exists a Turing machine Z^* such that $Z^*(X)=h(X)$. By construction, for each generalized sequential machine S_i, $S_i(f(X_i))\neq g(Z^*(X_i))$, contradicting condition (3). Hence the result.

Note that Theorem 4 holds even if f is allowed to be many to one, and that the Turing machine Z^* providing the contradiction for condition (3) computes a value for every input tape in its alphabet.

THEOREM 5. *Given any set of Turing machines in a finite joint alphabet there is a function* f *such that* (1) f *maps all Turing tapes one to one into sequential input tapes, and* (2) *for each Turing machine* Z *there is a generalized sequential machine* S *such that* $S(f(X))=Z(X)$.

Proof. Let $Z_1, Z_2, \cdots$ be the given Turing machines, and $X_1, X_2, \cdots$ the tapes in their joint alphabet Δ. Let f be the following one to one mapping of Turing tapes in Δ into sequential tapes in the alphabet $\Delta\cup\{A, C, D\}$, where A, C, and D are distinct symbols not in Δ. For all $n\geqq 1$ let $f(X_n)=A^{n-1}DCZ_1(X_n)C\cdots CZ_n(X_n)$(6). For each $n\geqq 1$, let S_n be the generalized sequential machine with input alphabet $\Delta\cup\{A, C, D\}$; output alphabet Δ; states $p_1, \cdots, p_n, q_1, \cdots, q_n, q_0$; start state p_1; and state function δ_n and output function λ_n next defined. Let δ_n and λ_n be the functions so that, for each state p and each input I, $\delta_n(p, I)=p$ and $\lambda_n(p, I)$ is undefined except in these cases: $\delta_n(p_i, A)=p_{i+1}$ if $1\leqq i<n$, $\delta_n(p_n, C)=q_1$, $\delta_n(q_i, C)=q_{i+1}$ if $1\leqq i<n$, $\delta_n(q_n, C)=q_0$, $\lambda_n(p_i, D)=Z_n(X_i)$ if $1\leqq i<n$ and $Z_n(X_i)$ is defined, $\lambda_n(q_n, I)=I$ if I is in Δ.

The proof is now completed by showing that $S_n(f(X_i))=Z_n(X_i)$ for all Turing input tapes X_i. Write $f(X_i)$ in the abbreviated form $A^{i-1}DY$, where Y is a tape in $\Delta\cup\{C\}$. Two cases exist.

(1) $i<n$. Then $S_n(f(X_i))=\lambda_n(p_1A^{i-1}DY)=\lambda_n(p_i, DY)=Z_n(X_i)\lambda_n(p_i, Y)=Z_n(X_i)$.

(6) If $Z_m(X_n)$ is undefined (as distinguished from the case that $Z_m(X_n)$ is defined and is B), it is to be regarded as causing no separation between C's on either side (i.e., $PZ_m(X_n)Q$ is to be regarded as denoting PQ).

(2) $i \geq n$. Then $S_n(f(X_i)) = \lambda_n(p_1, A^{i-1}DY) = \lambda_n(p_n, DY) = \lambda_n(p_n, Y) = \lambda_n(q_1, Z_1(X_i)C \cdots CZ_i(X_i)) = \lambda_n(q_n, Z_n(X_i)C \cdots CZ_i(X_i)) = Z_n(X_i)$ (if $i = n$) or $Z_n(X_i)\lambda_n(q_0, Z_{n+1}(X_i)C \cdots CZ_i(X_i)) = Z_n(X_i)$ (if $i > n$).

REMARK. Theorem 5 can be strengthened to apply to Turing machines in an infinite joint alphabet Δ. In this case a function g would be introduced for mapping Turing output tapes. The present proof could then be modified by letting f and g map Δ into a suitable binary alphabet.

3. **Sets of accepted tapes.** One well-known manifestation of work performed by generalized sequential machines is that of accepted input tapes. For a generalized sequential machine S (in this section that also includes a set F of designated (final) states (which may be empty)), an input tape X is said to be accepted by S if $\delta_S(q_1, X)$(7) is in F [3]. The set of *all* tapes accepted by such an S is said to be a set of accepted tapes. Because the notion of accepted tapes is formulated without reference to output it is unnecessary here to distinguish between generalized sequential machine and sequential machine. It is also known [3, Theorem 11] that, as far as sets of accepted tapes are concerned, it is unnecessary to distinguish between sequential and complete sequential machines.

An analogous formulation of the phrase "set of tapes accepted by a Turing machine T" occurs. It is readily seen that the sets of tapes accepted by Turing machines are precisely all recursively enumerable sets and the empty set. On the other hand, the sets of tapes accepted by sequential machines form a subclass of the general recursive sets. From the strict point of view, then, Turing machines do more work than sequential machines.

Consistent with the view of the preceding sections the following question arises. Is it possible to translate Turing tapes into sequential tapes so that all Turing accepted sets are reinterpreted as sequential accepted sets? More precisely, let f map all Turing tapes in the Turing joint alphabet one to one into sequential tapes. Can f be chosen so that for any Turing machine Z there is a sequential machine S such that the set of tapes accepted by Z maps onto the set accepted by S? Theorem 6 answers this question affirmatively. It is immediately apparent, however, that neither the function f nor the procedure for finding the sequential machine S for an arbitrary Turing machine Z can be algorithmic. For if f were effectively calculable, then every set of Turing accepted tapes, and consequently every recursively enumerable set, being recursive in a set of accepted sequential tapes, would be general recursive, contrary to fact. Suppose, on the other hand, that there were an algorithm giving, for every Turing machine Z, a sequential machine S such that the set of tapes accepted by Z is in a one to one correspondence with the set accepted by S. Then, inasmuch as the decision problem for *finiteness* of sequential accepted sets is solvable, the corresponding decision problem for Turing accepted sets, and consequently for all recursively enumerable sets,

(7) $\delta_S(q_1, I_1 \cdots I_k)$ is said to exist if each state $q_{i+1} = \delta_S(q_i, I_i)$ for $i \leq k$ exists; and $\delta_S(q_1, I_1 \cdots I_k)$, when it exists, is defined to be q_{k+1}.

would be solvable, contrary to fact.

THEOREM 6. *Given any set of Turing machines, there is a function f such that* (1) *f maps all Turing tapes in the given joint alphabet one to one into sequential tapes*; *and* (2) *for each Turing machine Z there is a sequential machine S such that the set of tapes accepted by Z maps, under f, onto the set of tapes accepted by S.*

Proof. The only property required of the set of tapes accepted by Z is that it is enumerable (no appeal being made to the recursive enumerability). The sets of tapes accepted by sequential machines in a given joint alphabet will be called the ***regular*** sets. The only properties required of the class of regular sets are that it includes all finite sets, all Boolean combinations of regular sets, and (for any tape Y and natural numbers a and b) all sets of the form $\{Y^{a+bn} \mid n \geqq 0\}$ [3].

Of the given Turing machines, consider only those for which the sets of accepted tapes and their complements are infinite(8). Let $E_1, E_2, \cdots$ be the corresponding sets of accepted Turing tapes. A function f will be defined that maps the Turing tapes one to one onto the set $\{Y^n \mid n \geqq 3\}$, where Y is a particular sequential tape. It suffices to show that the images of the sets E_i are regular, for the image of any other set, being either finite or finitely different(9) from the set $\{Y^n \mid n \geqq 3\}$, is regular.

For $i=1, 2, \cdots; j=1, \cdots, 2^i$, define the sets $E(i, j)$ of Turing tapes thus: $E(1, 1)=E_1$ and $E(1, 2)=\overline{E}_1$. For all $i>1$ and j from 1 to 2^{i-1}, $E(i, j)$ is to be the set $E(i-1, j)\cap E_i$ if the latter is infinite and $E(i-1, j)$ otherwise. $E(i, j+2^{i-1})$ is to be the set $E(i-1, j)\cap \overline{E}_i$ if the latter is infinite and $E(i-1, j)$ otherwise.

The sets $E(i, j)$ have the following properties.

(A) $E(i, j)$ is infinite.

(B) For each i, the $E(i, j)$ are exhaustive.

(C) $E(i, j)\cap E(i, k)$ is infinite if and only if $E(i, j) \doteq E(i, k)$.

(D) For each i, $E_i \doteq$ a union of the $E(i, j)$.

(E) If $i' \geqq i$ and $j' \equiv j \bmod 2^i$, then $E(i', j') \subseteq E(i, j)$.

(A) and (B) are easily established by induction on i.

(C) is proved by induction on i, with an obvious basis. For the induction step, note that, for $j>1$, any $E(i, j) \doteq$ either $E(i-1, u)\cap E_i$ or $E(i-1, u)\cap \overline{E}_i$ (where u is either j or $j-2^{i-1}$). Hence, if $E(i, j)\cap E(i, k)$ is infinite; then either $E(i, j) \doteq E(i-1, u)\cap \overline{E}_i$ and $E(i, k) \doteq E(i-1, v)\cap \overline{E}_i$, or $E(i, j) \doteq E(i-1, u)\cap \overline{E}_i$ and $E(i, k) \doteq E(i-1, v)\cap \overline{E}_i$. In either case, $E(i-1, u) \cap E(i-1, v)$ is infinite, so that by the induction hypothesis $E(i-1, u) \doteq E(i-1, v)$. Consequently $E(i, j) \doteq E(i, k)$.

To prove (D) express E_i as $(E_i\cap E(i, 1))\cup \cdots \cup(E_i\cap E(i, 2^i))$. By con-

(8) Complements are with respect to the set of all tapes in the given joint alphabet. The complement of A is denoted by $\overline{A}$.

(9) Sets A and B are "finitely different" if $A\cap \overline{B}$ and $\overline{A}\cap B$ are both finite. This relation will be denoted by "$A \doteq B$."

struction, $E_i \cap E(i, j)$ is either $\overset{\circ}{=} E(i, j)$ or finite. Therefore $E_i \overset{\circ}{=}$ the union of those $E(i, j)$ for which $E_i \cap E(i, j)$ is infinite.

(E) is proven by induction of i'. The case $i' \not> i$ is obvious. Hence assume $i' > i$ and $j' \equiv j \bmod 2^i$. By construction, $E(i', j')$ is a subset of $E(i'-1, j')$ if $j' \leqq 2^{i'-1}$ and is a subset of $E(i'-1, j'-2^{i'-1})$ if $2^{i'-1} < j' \leqq 2^{i'}$. But j' and $j'-2^{i'-1}$ are congruent mod 2^i to j' and hence to j. Therefore, by the induction hypotheses, $E(i'-1, j')$ and $E(i'-1, j'-2^{i'-1})$, and consequently $E(i', j')$, are subsets of $E(i, j)$.

Let the sets $E(i, j)$ be listed so that $E(i', j')$ occurs later than $E(i, j)$ if $i' > i$ or if $i' = i$ and $j' > j$. Run through the list, choosing from each $E(i, j)$ the first Turing tape not already chosen. By (A), each $E(i, j)$ will contribute a distinct tape $X(i, j)$, and by (B), the $X(i, j)$ will include all Turing tapes.

For each (i, j) $(i = 1, 2, \cdots; j = 1, \cdots, 2^i)$, let $E^*(i, j)$ be the set $\{X(i', j') \mid i' \geqq i, 1 \leqq j' \leqq 2^{i'}$ and $j' \equiv j \bmod 2^i\}$. The sets $E^*(i, j)$ have the following properties.

(F) $E^*(i, j) \subseteq E(i, j)$.

(G) For each i, j, $E(i, j) \overset{\circ}{=}$ a union of the $E^*(i, u)$.

(F) follows from (E) and the fact that $X(i', j')$ is in $E(i', j')$.

To prove (G), note that, for each i, $E^*(i, 1) \cup \cdots \cup E^*(i, 2^i)$ includes all but the finitely many $X(i', j')$ contributed by the $E(i', j')$ preceding $E(i, j)$. Hence

$$E(i, j) \overset{\circ}{=} (E(i, j) \cap E^*(i, 1)) \cup \cdots \cup (E(i, j) \cap E^*(i, 2^i)).$$

If $E(i, j) \cap E^*(i, u)$ is infinite, then by (F), so is $E(i, j) \cap E(i, u)$. Then by (C), $E(i, j) \overset{\circ}{=} E(i, u)$, so that $E(i, j) \cap E^*(i, u) \overset{\circ}{=} E(i, u) \cap E^*(i, u) = E^*(i, u)$. Therefore $E(i, j) \overset{\circ}{=}$ the union of those $E^*(i, u)$ for which $E(i, j) \cap E^*(i, u)$ is infinite.

Now let Y be any sequential tape, and define $f(X(i, j)) = Y^{2^i+j}$. Thus f maps all Turing tapes one-one onto $\{Y^n \mid n \geqq 3\}$. The proof is completed by showing that the map of each $E^*(i, j)$ is regular. For by (D) and (G), the map of E_i is then regular. For any element $X(i', j')$ of $E^*(i, j)$ there is an integer $n \geqq 0$ such that $2^{i'} + j' = 2^i + j + 2^i n$. Conversely, for every $n \geqq 0$ there exist i' and j'; where $i' \geqq i$, $1 \leqq j' \leqq 2^{i'}$, and $j' \equiv j \bmod 2^i$; such that $2^{i'} + j' = 2^i + j + 2^i n$. Therefore $E^*(i, j)$ maps onto the regular set $\{Y^{2^i+j+2^i n} \mid n \geqq 0\}$. Q.E.D.

Bibliography

1. M. Davis, *Computability and unsolvability*, McGraw-Hill, New York, 1958.

2. S. Ginsburg, *On the reduction of superfluous states in a sequential machine*, J. Assoc. Comput. Mach. **6** (1959), 259–282.

3. M. Rabin and D. Scott, *Finite automata and their decision problem*, IBM J. Research Develop. **3** (1959), 114–126.

System Development Corporation,
Santa Monica, California