On Formalisms for Turing Machines

PATRICK C. FISCHER

Harvard University, Cambridge, Massachusetts

Abstract. Turing's original quintuple formalism for an abstract computing machine is compared with the quadruple approach of Post and with some new alternatives. In each case the possibility or nonpossibility of two-symbol or two-state universal machines is demonstrated.

1. *Introduction*

The term "Turing machine" has been applied to several different characterizations of an abstract computing machine. Since each of the formalisms has been adequate for a development of recursive function theory, no serious trouble has arisen from the multiple use of the term. In this paper various formal definitions for the notion of a general-purpose abstract computer are compared, and some new alternative definitions are introduced. Particular attention is paid to one of Turing's original formalisms and to one by Post; the latter has been used extensively by Davis in [2].

Most of the theorems below assert that a certain kind of machine simulates another kind of machine. However, the concept of simulation of one machine by another is extremely difficult to define precisely. Too stringent a definition excludes cases in which one intuitively feels a bona fide simulation is being performed. Too liberal a definition allows the use of encodings of input and output in which the real computational work is done by the encoding and decoding algorithms and not by the machine which is supposedly performing the simulation.

The notion of simulation of one machine by another used here requires that intermediate results of the computations by the two machines be closely related as well as the outputs of the computations; i.e., the simulation is "step by step." An attempt at a precise definition is given in the Appendix, and it is hoped that the notion of simulation is correctly captured by the definition. However, the theorems of this paper clearly satisfy any reasonable definition of simulation, and the author invites suggestions for improving the definition.

Theorem 2 is due jointly to S. Aanderaa and the author [1], and Theorem 8 is due to P. K. Hooper [4]. The author is also indebted to the referee for his comments and for his suggestion of a way to strengthen the originally submitted version of Theorem 3.

2. *Turing Machines*

A Turing machine is usually regarded as a small computer with a finite number of states and a (potentially) infinite tape marked off into discrete squares. Upon each square of the tape is written one symbol selected from a finite alphabet; all but a finite number of the squares contain the same symbol, B (blank).

In Turing's original formulation in [8], the operation of a Turing machine was

An earlier version of this work was presented under the same title to the Fifth Annual Symposium on Switching Circuit Theory and Logical Design, Princeton, November, 1964.

This work was partially supported by Bell Telephone Laboratories and by a grant from the National Science Foundation.

the following. At a given time, the machine would be in some state and would be scanning a square on the tape. For each state-symbol pair there would be either a well defined operation or a command to halt. A nonhalting operation (step) would consist of three suboperations:

(1) A new symbol (possibly the same as the previous one) would be written on the square being scanned.

(2) The scanning head of the machine would move to the left or to the right one square on the tape. (Turing also allowed the machine to stay on the same square.)

(3) The machine would enter a new state (possibly the same as the previous one).

The machine would then be scanning a new symbol, and to the new state-symbol pair there would correspond another operation, etc.

A slightly more formal approach to the above version of a Turing machine is to define a machine as a set of quintuples $\langle q_i , S_j , S_k , D, q_l \rangle$ with the five components of the quintuple representing present state, present symbol being scanned, new symbol written, direction moved on tape, and next state. (By convention, the machine halts if it reaches a state-symbol configuration for which there is no quintuple.) In order for the machine to be well defined, q_i and q_l must be selected from a finite set of states $\{q_1 , q_2 , \cdots , q_n\}$, S_j and S_k must be selected from a finite set of symbols $\{S_0 (= B), S_1 , S_2 , \cdots , S_{m-1}\}$, and D must be either L, R or N (left, right or no tape motion). Furthermore, no two distinct quintuples of the machine may have as their leftmost two components the same state-symbol pair.

Often the possibility $D = N$ is excluded from the characterization of Turing machines. It is well known that this entails no loss of generality. For the sake of completeness of the discussion, a proof is given below.

Definition. A Turing machine has a *blocking-loop* if for some k there exist integers $i_1 , i_2 , \cdots , i_k$ and $j_1 , j_2 , \cdots , j_k$ such that the machine includes the k quintuples:

$$\langle q_{i_1} , S_{j_1} , S_{j_2} , N, q_{i_2} \rangle, \langle q_{i_2} , S_{j_2} , S_{j_3} , N, q_{i_3} \rangle, \cdots, \langle q_{i_k} , S_{j_k} , S_{j_1} , N, q_{i_1} \rangle$$

Clearly, one can tell effectively whether or not a Turing machine has a blocking loop, although one cannot in general tell whether or not the loop will ever be entered. A machine in a blocking loop is not halted, but except for the square being scanned at that time, the tape will undergo no further changes. Therefore, one can dispense with the ability of a machine to enter a blocking loop without weakening its computing ability.

LEMMA. *For each Turing machine without any blocking loops, there exists a Turing machine with the same sets of states and symbols, which simulates the first machine and which has no quintuples with* $D = N$.

PROOF. Whenever there exist two quintuples of the machine of the form $\langle q_i , S_j , S_k , N, q_l \rangle$ and $\langle q_l , S_k , S_r , D, q_t \rangle$ where $D \neq N$, let the first of these be replaced by $\langle q_i , S_j , S_r , D, q_t \rangle$. This results in the elimination of a quintuple with $D = N$. The absence of blocking loops guarantees that every quintuple with $D = N$ will eventually be eliminated. It is clear that the new machine simulates the original machine.

Henceforth, the term "Turing machine" will denote a quintuple machine in which there are no quintuples with $D = N$.

Shannon has shown in [7] that, given enough symbols, one can construct a uni-

versal Turing machine with only two states. Furthermore, given enough states, one can construct a universal Turing machine with only two symbols (by using the procedure of encoding 2^k symbols as binary sequences of length k). Thus, there is a certain symmetry between states and symbols, or in the parlance of computing, between instructions and data.

3. *Post Machines*

A popular variant on Turing's formalism is due to Post [6] and is the model used by Davis in [2]. In this formalism a machine is represented by a set of quadruples of the form $\langle q_i , S_j , X, q_l\rangle$ where q_i, S_j and q_l have the same interpretations as in the quintuple model and X is either L, R or some symbol S_k. Other criteria for "well definedness" are as before. If X is L or R the machine leaves the symbol under scan alone and moves left or right on the tape. If X is a symbol S_k then S_k is written on the square being scanned but the machine does not move. Thus in a single operation a Post machine can only perform two of the three possible suboperations of a Turing machine and it can never perform both suboperations (1) and (2) on the same step.

Since many of the operations in a "typical" Turing machine program tend to be searches along the tape for a specific symbol or combination of symbols, one seldom takes advantage of the quintuple machine's ability to write and to move on the same step. Thus the number of quadruples in a Post machine would often be of the same order of magnitude as the number of quintuples in an equivalent Turing machine, and the description of the Post machine would take fewer characters than the description of the Turing machine. For this reason the Post formulation has been used relatively often in the literature of computability theory.

Let us now seek some more precise statements about the relationship between Turing machines and Post machines. The first theorem is elementary.

THEOREM 1. *For every m-symbol n-state Turing machine there exists a Post machine with m symbols and at most $3n$ states which simulates the Turing machine.*

PROOF. Let $S_0 , S_1 , \cdots , S_{m-1}$ be the symbols of the Turing machine and $q_1 , q_2 , \cdots , q_n$ be its states. Then the Post machine will have the same symbols as the Turing machine and will have as states $q_1 , q_2 , \cdots , q_n$; $q_{1,L} , q_{2,L} , \cdots , q_{n,L}$; $q_{1,R} , q_{2,R} \cdots , q_{n,R}$. For each quintuple $\langle q_i , S_j , S_k , D, q_l\rangle$ in the Turing machine, the Post machine will have a quadruple $\langle q_i , S_j , S_k , q_{l,D}\rangle$. In addition, the Post machine will have mn quadruples of the form $\langle q_{l,L} , S_k , L, q_l\rangle$ and mn quadruples of the form $\langle q_{l,R} , S_k , R, q_l\rangle$ $(l = 1, 2, \cdots , n; k = 0, 1, \cdots , m-1)$. The verification that the Post machine is well defined and exhibits the correct computational behavior (when given the same initial instantaneous description as the Turing machine) is trivial.

As an immediate consequence of the above we have the well known

Remark. There exists a 2-symbol universal Post machine.

Although one can map Turing machines into equivalent Post machines without increasing the number of symbols, it is not possible in general to find equivalent Post machines for Turing machines without increasing the number of states. In particular, the 2-state universal Turing machine of Shannon must map into a Post machine of more than two states. The main result in this area is Theorem 2 below.

THEOREM 2 (with S. Aanderaa). *The halting problem for the class of 2-state Post machines is recursively solvable.*

PROOF. A proof of Theorem 2 is planned for publication in a separate paper [1].

COROLLARY. *There is no 2-state universal Post machine.*

PROOF. A universal machine must have an unsolvable halting problem (cf. [3]).

The construction of Theorem 1 shows that there is a 6-state universal Post machine. However, one can improve on this number as a consequence of the following theorem.

THEOREM 3. *For every m-symbol n-state Turing machine there exists a Post machine with $n+1$ states and at most $m(n+1)$ symbols which simulates the Turing machine.*

PROOF. Let the symbols and states of the Turing machine be as in Theorem 1. The Post machine will have the same states plus an additional state, q_{n+1}. Its symbols will be $S_0, S_1, \cdots, S_{m-1}$ plus mn additional symbols of the form $S_{i,j}$ $(i = 1, 2, \cdots, n;\ j = 0, 1, \cdots, m-1)$. For each quintuple $\langle q_i, S_j, S_k, D, q_l\rangle$ of the Turing machine, the Post machine will have the quadruple $\langle q_i, S_j, S_{i,j}, q_{n+1}\rangle$, n quadruples of the form $\langle q_p, S_{i,j}, S_{p,k}, q_{n+1}\rangle$ $(p = 1, 2, \cdots, n)$, and the quadruple $\langle q_{n+1}, S_{i,j}, D, q_l\rangle$. The initial instantaneous description of the Post machine will be the same as that of the Turing machine.

During the operation of the Post machine, a symbol $S_{i,j}$ has two functions. When the Post machine is in state q_{n+1}, $S_{i,j}$ is used to indicate the proper direction and state change given by the quintuple $\langle q_i, S_j, S_k, D, q_l\rangle$. When the Post machine is not in q_{n+1}, $S_{i,j}$ plays the role of the unique S_k determined by the Turing machine quintuple. The mapping h: $S_{i,j} \to S_k$; $S_k \to S_k$; $q_l \to q_l$ yields the instantaneous descriptions of the Turing machine whenever the Post machine is not in state q_{n+1}.

COROLLARY. *There exists a 3-state universal Post machine.*

PROOF. Since the proof is obvious, it is omitted in this discussion.

COROLLARY. *There is no 1-state universal Turing machine.*

PROOF. If there were a 1-state universal Turing machine, there would be a 2-state universal Post machine,—a contradiction.

The second corollary is a stronger version of a result of Shannon in [7]. (Direct proofs not using Theorem 3 have been given by R. Abbott and R. Boyd.) Shannon showed that no 1-state machine could simulate the behavior of a machine which performed an infinite computation yielding convergence to the binary expansion of $1/\sqrt{2}$. His approach employs a definition of universality stronger than that in [3]; consequently, it is somewhat easier to show that a machine is not universal. The fact that a machine cannot give the entire binary expansion of $1/\sqrt{2}$ does not necessarily imply that it could not produce as output the first x bits of $1/\sqrt{2}$, given x as input. In other words, if one considered the infinite class of all possible finite computations, such a machine might still be able to do arbitrarily complex things although it could not handle the infinite computations properly.

4. *D-Machines*

There are clearly other quadruple formalisms for abstract computing machines, since one might invoke other restrictions than those of Post on the possible suboperations performed on a given step. These cases are considered below.

Definition. A *D-machine* is a set of quadruples of the form $\langle q_i, S_j, X, D\rangle$ where D is either L, R or N, and X is either a symbol S_k or a state q_l. The restrictions on the quadruples of a well defined D-machine are analogous to those for well defined Turing machines.

The state-symbol pair $\langle q_i , S_j \rangle$ has the usual interpretation. If X is a symbol S_k then S_k is written on the scanned square and the machine moves in direction D but remains in the same state. If X is a state q_l the machine does not write a new symbol, but moves in direction D and enters state q_l. Thus a D-machine can never perform suboperations (1) and (3) on the same step.

THEOREM 4. *For every m-symbol n-state Turing machine there exists a D-machine with m symbols and at most $(m+1)n$ states which simulates the Turing machine.*

PROOF. Let the symbols and states of the Turing machine be as in Theorem 1. The D-machine will have the same symbols. Its states will be $q_1 , q_2 , \cdots , q_n$ plus mn additional states of the form $q_{i,j}$ $(i = 1, 2, \cdots , n; j = 0, 1, \cdots , m-1)$. For each quintuple $\langle q_i , S_j , S_k , D, q_l \rangle$ of the Turing machine, if $j \neq k$ the D-machine will have the three quadruples $\langle q_i , S_j , q_{i,j} , N \rangle$, $\langle q_{i,j} , S_j , S_k , N \rangle$, $\langle q_{i,j} , S_k , q_l , D \rangle$. If $j = k$ the D-machine will simply have the quadruple $\langle q_i , S_j , q_l , D \rangle$, and state $q_{i,j}$ will not be used. The D-machine will have the same initial instantaneous description as the Turing machine.

In operation, the D-machine first changes to a state which represents uniquely the three suboperations that must be done to simulate the effect of the given Turing machine quintuple. Then the D-machine writes the proper symbol. Finally, it changes to the correct state and makes the correct move. Whenever the D-machine is in one of the states $q_1 , q_2 , \cdots , q_n$, its instantaneous description is the same as the appropriate one of the Turing machine.

COROLLARY. *There exists a 2-symbol universal D-machine.*

Theorem 4 made use of the ability of a D-machine to have $D = N$. If a D-machine is restricted so that all quadruples have $D \neq N$, then the above method of proof will not go through. In particular, in the case of a 2-symbol restricted D-machine, the number of alternations of blanks and 1's on the tape cannot be increased over the number on the tape at the start of the computation. However, a 2-symbol restricted D-machine can be shown universal via the simulation of a 2-counter machine, which Minsky has shown can simulate a universal Turing machine.

Definition. A *2-counter machine* is a set of triples of the form $\langle q_i , X, q_j \rangle$ where X is either I_1 , I_2, of the form $D_1 q_k$, or of the form $D_2 q_k$. No two triples begin with the same q_i, and q_i , q_j and q_k are all members of the same finite set Q. The machine has two counters; the value of each is a non-negative integer. When the machine is in state q_i, its behavior is determined by the triple beginning with q_i. If X is I_1 the value of the first counter is increased by 1 and the machine enters state q_j. If X is $D_1 q_k$ and the value of the first counter is 0 the machine enters state q_k; otherwise, the value of the first counter is decreased by 1 and the machine enters state q_j. Operations I_2 and $D_2 q_k$ similarly affect the second counter.

THEOREM 5. *For any 2-counter machine there exists a 2-symbol D machine which simulates the 2-counter machine and which has no quadruples with $D = N$.*

PROOF. Denote S_0 by B and S_1 by 1. The tape of the D-machine will always be of the form $...BBB1B^m1B^n1BBB...$, where B^m means m consecutive B's. The values of the first and second counters will be represented by $m-1$ and $n-1$, respectively. Constructions will be given to handle operations I_2 and $D_2 q_k$; I_1 and $D_1 q_k$ are handled by a symmetrical construction.

At the start of a cycle, the D-machine is either scanning the square to the left or the square to the right of the middle 1, depending on whether the operation to be simulated affects the first or second counter, respectively. States of the form $q_{i,p}$

are used for simulating the action of the triple beginning with q_i. The quadruples used for the simulation of $\langle q_i, I_2, q_j\rangle$ are the following:

$$\begin{array}{lll}
\langle q_{i,1}, B, B, R\rangle & \langle q_{i,2}, 1, q_{i,3}, R\rangle & \langle q_{i,4}, B, q_{i,5}, L\rangle \\
\langle q_{i,1}, 1, q_{i,2}, R\rangle & \langle q_{i,3}, 1, q_{i,4}, L\rangle & \langle q_{i,5}, B, B, L\rangle \\
\langle q_{i,2}, B, 1, L\rangle & \langle q_{i,4}, 1, B, L\rangle & \langle q_{i,5}, 1, q_{j,1}, D\rangle
\end{array}$$

where $D = L$ if the triple beginning with q_j affects the first counter and $D = R$ if it affects the second counter.

The quadruples used for the simulation of $\langle q_i, D_2 q_k, q_j\rangle$ are the following:

$$\begin{array}{llll}
\langle q_{i,1}, B, q_{i,2}, R\rangle & \langle q_{i,2}, B, q_{i,4}, R\rangle & \langle q_{i,5}, 1, q_{i,6}, L\rangle & \langle q_{i,8}, B, B, L\rangle \\
\langle q_{i,2}, 1, q_{i,3}, L\rangle & \langle q_{i,4}, B, B, R\rangle & \langle q_{i,6}, 1, q_{i,7}, R\rangle & \langle q_{i,8}, 1, q_{i,9}, L\rangle \\
\langle q_{i,3}, B, B, L\rangle & \langle q_{i,4}, 1, q_{i,5}, L\rangle & \langle q_{i,7}, 1, B, R\rangle & \langle q_{i,9}, B, B, L\rangle \\
\langle q_{i,3}, 1, q_{k,1}, D\rangle & \langle q_{i,5}, B, 1, R\rangle & \langle q_{i,7}, B, q_{i,8}, L\rangle & \langle q_{i,9}, 1, q_{j,1}, D'\rangle
\end{array}$$

where D and D' depend on the counters affected by the triples beginning with q_k and q_j respectively.

COROLLARY. *There exists a 2-symbol universal D machine in which all quadruples have $D \neq N$.*

PROOF. The proof follows from Theorem 5 and from Minsky's Theorem that there exists a universal 2-counter machine (see [5]).

Turning now to the question of the existence of a 2-state universal D-machine, one can give a positive answer via a more tightly coded version of Shannon's ingenious proof in [7]. The proof does not need quadruples with $D = N$.

THEOREM 6. *For every m-symbol n-state Turing machine, there exists a D-machine with only 2 states and at most $m(4n + 7)$ symbols which simulates the Turing machine.*

PROOF. Let the states and symbols of the Turing machine be as in Theorem 1. The D-machine will have states q_1 and q_2. Its symbols will be $S_0, S_1, \cdots, S_{m-1}$, plus $4m(n + 1)$ symbols of the form $S_{i,j,*,D}$ $(i = 0, 1, \cdots, m-1;\ j = 0, 1, 2, \cdots, n;\ * = +, -;\ D = L, R)$, and $2m$ symbols of the form $S_{i,n+1,+,D}$ $(i = 0, 1, \cdots, m-1;\ D = L, R)$. For each Turing machine quintuple of the form $\langle q_j, S_i, S_k, L, q_l\rangle$ the D-machine will have the two quadruples

$$\langle q_1, S_{i,j,-,R}, S_{k,l+1,+,L}, L\rangle \quad \text{and} \quad \langle q_2, S_{i,j,-,L}, S_{k,l,+,L}, R\rangle.$$

For each quintuple of the form $\langle q_j, S_i, S_k, R, q_l\rangle$ the D-machine will have the two quadruples

$$\langle q_1, S_{i,j,-,R}, S_{k,l,+,R}, L\rangle \quad \text{and} \quad \langle q_2, S_{i,j,-,L}, S_{k,l+1,+,R}, R\rangle.$$

In addition, the machine will have the following sets of quadruples for each i $(i = 0, 1, \cdots, m-1)$:

Quadruples Used when the Turing Machine is Moving to the Right

$$\begin{array}{lll}
\langle q_1, S_{i,j,+,R}, q_2, R\rangle & & (1 \leqq j \leqq n) \\
\langle q_1, S_{i,0,+,R}, S_i, R\rangle & & \\
\langle q_2, S_i, S_{i,0,-,R}, L\rangle & & \\
\langle q_2, S_{i,j,+,R}, S_{i,j-1,+,R}, R\rangle & & (1 \leqq j \leqq n+1) \\
\langle q_2, S_{i,0,+,R}, q_1, R\rangle & & \\
\langle q_2, S_{i,j,-,R}, S_{i,j+1,-,R}, L\rangle & & (0 \leqq j \leqq n-1)
\end{array}$$

Quadruples Used when the Turing Machine is Moving to the Left

$$\begin{array}{lll}
\langle q_2, S_{i,j,+,L}, q_1, L\rangle & & (1 \leqq j \leqq n) \\
\langle q_2, S_{i,0,+,L}, S_i, L\rangle & & \\
\langle q_1, S_i, S_{i,0,-,L}, R\rangle & & \\
\langle q_1, S_{i,j,+,L}, S_{i,j-1,+,L}, L\rangle & & (1 \leqq j \leqq n+1) \\
\langle q_1, S_{i,0,+,L}, q_2, L\rangle & & \\
\langle q_1, S_{i,j,-,L}, S_{i,j+1,-,L}, R\rangle & & (0 \leqq j \leqq n-1)
\end{array}$$

If the initial instantaneous description of the Turing machine is $S_{i_{-p}} \cdots S_{i_{-2}} S_{i_{-1}} q_j S_{i_0} S_{i_1} \cdots S_{i_r}$, then the initial instantaneous description of the D machine will be $S_{i_{-p}} \cdots S_{i_{-2}} S_{i_{-1},0,+,R} q_1 S_{i_0,j,-,R} S_{i_1} \cdots S_{i_r}$.

At each step in the computation of the D-machine only two tape squares contain symbols other than S_0, S_1, $\cdots$, S_{m-1}. The subscript of each of these special symbols has four components, the first of which contains the index of the ordinary symbol which is currently on the associated square of the Turing machine tape. One of the special symbols has a "+" in the third component of its subscript and transmits information about the state of the Turing machine to the other square, which contains a symbol with a "−" in the third component. The fourth components of the subscripts of both symbols are the same as the direction of the Turing machine move being simulated. If the fourth components are L then the receiving symbol is to the left of the transmitting symbol; otherwise the reverse holds.

The second components of the subscripts of the two symbols contain information about the state of the Turing machine. The D-machine oscillates back and forth between the two squares decreasing the second component of the subscript of the transmitting symbol and increasing that of the receiving symbol until the former quantity is zero. The D-machine then enters the receiving square in a different state, thus indicating the end of one simulation cycle. At this point, the value of the second component of the subscript of the receiving symbol is the index of the current Turing machine state. (One more visit is made to the transmitting square to change the symbol there to an ordinary tape symbol.)

Whenever the D-machine is in state q_1 scanning a symbol of the form $S_{i,j,-,R}$ or in state q_2 scanning a symbol of the form $S_{i,j,-,L}$, the instantaneous description of the Turing machine can be recovered from the tape configuration of the D-machine via the mapping h: $S_i \rightarrow S_i$; $S_{i,0,+,D} \rightarrow S_i$; $S_{i,j,-,D} \rightarrow q_j s_i$.

The notation used above is similar to that used by Shannon in [7], and the reader wishing to gain a more thorough understanding of this construction may find the discussion there a helpful supplement. An example of a computation by the 2-state universal D-machine is given in the Appendix.

5. *S-Machines*

For completeness, we consider the third of the restricted quadruple formalisms.

Definition. An *S-machine* is a set of quadruples of the form $\langle q_i, S_j, S_k, X \rangle$ where X is either L, R or a state q_l. The restrictions on the quadruples of a well defined S-machine are analogous to those for well defined Turing machines.

The entries q_i, S_j, S_k have the same interpretation as in a Turing machine quintuple. If X is L or R the S-machine moves after writing S_k but does not change state. If X is a state q_l then the machine enters state q_l but does not move on that step. Thus an S-machine can never perform suboperations (2) and (3) on the same step.

THEOREM 7. *For every m-symbol n-state Turing machine there exists an S-machine with n states and at most 3m symbols which simulates the Turing machine.*

PROOF. Let the symbols and states of the Turing machine be as in Theorem 1. The S-machine will have the same states as the Turing machine and will have as symbols S_0, S_1, $\cdots$, S_{m-1} ; $S_{0,L}$, $S_{1,L}$, $\cdots$, $S_{m-1,L}$; $S_{0,R}$, $S_{1,R}$, $\cdots$, $S_{m-1,R}$. For

each quintuple $\langle q_i, S_j, S_k, D, q_l\rangle$ of the Turing machine the S-machine will have a quadruple $\langle q_i, S_j, S_{k,D}, q_l\rangle$. In addition, the S-machine will have mn quadruples of the form $\langle q_l, S_{k,L}, S_k, L\rangle$ and mn quadruples of the form $\langle q_l, S_{k,R}, S_k, R\rangle$ ($l = 1, 2, \cdots, n; k = 0, 1, \cdots, m-1$). The initial instantaneous description of the S-machine will be the same as that of the Turing machine.

COROLLARY. *There exists a 2-state universal S-machine.*

THEOREM 8 (P. K. Hooper). *The halting problem for the class of 2-symbol S-machines is recursively solvable.*

PROOF. Given a 2-symbol S-machine, add a new state q_I and add two quadruples which begin with q_I and are otherwise the same as the two quadruples associated with the initial state of the machine. Redesignate q_I as the initial state of the machine. Now let the machine be further modified as follows: Whenever a quadruple of the form $\langle q_i, S_j, S_k, q_l\rangle$ is found where there is no quadruple beginning with $\langle q_l, S_k\rangle$, delete the quadruple; whenever two quadruples of the form $\langle q_i, S_j, S_k, q_l\rangle$ and $\langle q_l, S_k, S_u, q_v\rangle$ are found, replace the first quadruple by $\langle q_i, S_j, S_u, q_v\rangle$. Whenever a noninitial state becomes inaccessible, eliminate it and its quadruples. Continue until no further transformations can be made.

The process will eventually terminate, and the modified S-machine will halt if and only if the original machine halts. Since state changes can be caused only by quadruples of the form $\langle q_i, S_j, S_k, q_l\rangle$ it follows that with each q_i $(i \neq I)$ there will be associated at most one such quadruple, else q_i would have been made inaccessible by the above procedure. Thus, each noninitial state in the modified machine can have at most one successor state. Consequently, after leaving q_I, the machine will begin to follow a path of potential state changes involving a finite number of states. If the path cannot lead to a halting situation (because every state in the path has a successor state), then the halting question can be answered in the negative. If the path eventually leads to a state with no successor, one can still solve the halting problem, for it is clear that, having reached a given state, one can effectively determine whether or not the successor of that state will ever be reached. The machine halts only if the end of the path is attained.

THEOREM 9. *For every m-symbol n-state Turing machine there exists an S machine with $m+1$ symbols and at most $(m+1)n$ states which simulates the Turing machine.*

PROOF. Let the symbols and states of the Turing machine be as in Theorem 1. The S-machine will have the same symbols plus an additional symbol, S_m. The states will be $q_1, q_2, \cdots, q_n$ plus mn additional states of the form $q_{i,j}$ $(i = 1, 2, \cdots, n; j = 0, 1, \cdots, m-1)$. For each quintuple $\langle q_i, S_j, S_k, D, q_l\rangle$ of the Turing machine the S-machine will have the quadruple $\langle q_i, S_j, S_m, q_{i,j}\rangle$, m quadruples of the form $\langle q_{i,j}, S_p, S_m, q_{l,p}\rangle$ $(p = 0, 1, \cdots, m-1)$, and the quadruple $\langle q_{i,j}, S_m, S_k, D\rangle$. The initial instantaneous description of the S-machine will be the same as that of the Turing machine.

A state $q_{i,j}$ in conjunction with S_m causes the symbol and move specified by the quintuple $\langle q_i, S_j, S_k, D, q_l\rangle$. When the S-machine is in state $q_{i,j}$ and not scanning an S_m, it behaves as though it were in the state q_l determined by the Turing machine quintuple. The mapping h: $q_l \to q_l$; $q_{i,j} \to q_l$; $S_k \to S_k$ yields the instantaneous description of the Turing machine whenever the S-machine is not scanning an S_m.

COROLLARY. *There exists a 3-symbol universal S machine.*

Theorems 7, 8 and 9 are essentially the duals of Theorems 1, 2 and 3.

6. *U-machines*

Having considered the various quadruple formalisms, one is led naturally to consider machines which can perform only one of the three suboperations of a Turing machine on a given step. Such a machine can be described as a set of triples, rather than a set of quadruples. It is mildly surprising that the class of such machines is still as powerful as the class of Turing machines if one ignores considerations of computational speed or machine size.

Definition. A *U-machine* is a set of triples $\langle q_i , S_j , X\rangle$ where X is either L, R, a symbol S_k or a state q_l. The restrictions on the triples of a well defined U-machine are analogous to those for well defined Turing machines.

The state-symbol pair $\langle q_i , S_j\rangle$ has the usual interpretation. If X is L or R, the machine moves in the appropriate direction leaving S_j alone and remaining in state q_i. If X is a symbol S_k then S_k is written on the scanned square and the machine remains in state q_i scanning the same square. If X is a state q_l the machine does not move or change the symbol S_j, but enters state q_l.

THEOREM 10. *For every m-symbol n-state Turing machine there exists a U-machine with at most* $3m$ *symbols and* $2(m + 2)n$ *states which simulates the Turing machine.*

PROOF. Let the symbols and states of the Turing machine be as in Theorem 1. The symbols of the U-machine will be S_0, S_1, $\cdots$, S_{m-1}; S_0', S_1', $\cdots$, S_{m-1}'; S_0'', S_1'', $\cdots$, S_{m-1}''. The U-machine will have $2n$ states of the form $q_{i,D}$, $2n$ states of the form $q_{i,D}'$ and $2mn$ states of the form $q_{i,D,j}$ $(i = 1, 2, \cdots, n;\ D = L, R;\ j = 0, 1, \cdots, m-1)$. For each quintuple $\langle q_i , S_j , S_k , D, q_l\rangle$ of the Turing machine, the U-machine will have the two triples $\langle q_{i,L} , S_j' , q_{l,D,k}\rangle$ and $\langle q_{i,R} , S_j' , q_{l,D,k}\rangle$. In addition, the U machine will have the following sets of triples for each l and D $(l = 1, 2, \cdots, n;\ D = L, R)$.

$\langle q_{l,D,k} , S_j', S_k''\rangle$	$(0 \leqq j \leqq m-1;\ \ 0 \leqq k \leqq m-1)$
$\langle q_{l,D,k} , S_k'' , D\rangle$	$(0 \leqq k \leqq m-1)$
$\langle q_{l,D,k} , S_p , q_{l,D}'\rangle$	$(0 \leqq k \leqq m-1;\ \ 0 \leqq p \leqq m-1)$
$\langle q_{l,D} , S_p , S_p'\rangle$	$(0 \leqq p \leqq m-1)$
$\langle q_{l,D} , S_p', \bar{D}\rangle$	$(D \in \{L, R\}$ and $\bar{D} \neq D;\ \ 0 \leqq p \leqq m-1)$
$\langle q_{l,D} , S_k'', q_{l,D}\rangle$	$(0 \leqq k \leqq m-1)$
$\langle q_{l,D} , S_k'', S_k\rangle$	$(0 \leqq k \leqq m-1)$
$\langle q_{l,D} , S_k , D\rangle$	$(0 \leqq k \leqq m-1)$

The initial instantaneous description of the U-machine will be the same as that of the Turing machine except that the symbol on the square being scanned will be primed and the initial state will be $q_{i,L}$ instead of q_i. Whenever the U-machine is in an unprimed state scanning a primed symbol, the instantaneous description of the Turing machine can be recovered via the mapping h: $q_{i,D} \to q_i$; $S_j' \to S_j$; $S_j \to S_j$.

To understand the operation of the U-machine, consider the following example. Let the Turing machine have instantaneous description $\cdots q_i S_j S_p \cdots$ and let the appropriate quintuple be $\langle q_i , S_j , S_k , R, q_l\rangle$. Following the machine specifications given above in order, regarding the variables i, j, k, l and p as free rather than bound and setting $D = R$. The successive instantaneous descriptions of the U-machine will be as follows, where D is the direction of the previous move of the Turing

machine:

$$\begin{array}{ll}
\cdots q_{i,D}S_j{}'S_p\cdots & \cdots S_k''q'_{l,R}S_p{}'\cdots \\
\cdots q_{l,R,k}S_j{}'S_p\cdots & \cdots q'_{l,R}S_k''S_p{}'\cdots \\
\cdots q_{l,R,k}S_k''S_p\cdots & \cdots q_{l,R}S_k''S_p{}'\cdots \\
\cdots S_k''q_{l,R,k}S_p\cdots & \cdots q_{l,R}S_kS_p{}'\cdots \\
\cdots S_k''q'_{l,R}S_p\cdots & \cdots S_kq_{l,R}S_p{}'\cdots
\end{array}$$

THEOREM 11. *There is neither a 2-state nor a 2-symbol universal U-machine.*

PROOF. A U-machine can be viewed either as a restricted Post machine or as a restricted S-machine. The result then follows immediately from Theorems 2 and 8.

REFERENCES

1. AANDERAA, S., AND FISCHER, P. C. The solvability of the halting problem for 2-state Post machines. In preparation.
2. DAVIS, M. *Computability and Unsolvability*. McGraw-Hill, New York, 1958.
3. ——. A note on universal Turing machines. *Automata Studies*, Princeton U. Press, Princeton, N. J., 1956.
4. HOOPER, P. K. A note on Turing machines. *Amer. Math Soc. Notices* (Jan. 1965).
5. MINSKY, M. L. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. *Annal. Math. 74* (1961), 437–455.
6. POST, E. L. Recursive unsolvability of a problem of Thue. *J. Symbolic Logic. 12* (1947), 1–11.
7. SHANNON, C. E. A universal Turing machine with two internal states. *Automata Studies*, Princeton U. Press, Princeton, N. J., 1956.
8. TURING, A. M. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc. 42-2* (1936–37), 230–265; Correction, *ibid., 43* (1937), 544–546.

APPENDIX

1. *Simulation of One Machine by Another*

The terminology below is that in [2]. Let M_1 and M_2 be two machines. Let p be a characteristic function defined over the set C_2 of all possible instantaneous descriptions of M_2. Let h be a mapping from C_2 onto C_1, the set of all possible instantaneous descriptions of M_1, and f be a 1–1 function from C_1 into C_2. Let $C_1(I, t)$ be the instantaneous configuration of M_1 at time t if M_1 was started in instantaneous configuration I at time 0 (thus $C_1(I, 0) = I$). $C_2(I, t)$ is defined in a similar fashion.

Definition. For every initial instantaneous configuration I for M_1 let a function g be defined on the non-negative integers by the following recursion equations:

$$g(0) = \mu z[p(C_2(f(I), z)) = 1]$$

$$g(t+1) = \mu z[z > g(t) \quad \text{and} \quad p(C_2(f(I), z)) = 1]$$

Then M_2 *simulates* M_1 with respect to a class of functions $\mathfrak{C}$ if and only if for every initial instantaneous description I of M_1 and every time $t \geqq 0$: (1) $C_1(I, t) = h(C_2(f(I), g(t)))$; (2) $C_1(I, t)$ is final if and only if $C_2(f(I), g(t))$ is final; and (3) f, p and $h \subset \mathfrak{C}$. The class $\mathfrak{C}$ should consist of functions which are in some

sense "easy to compute." A reasonable candidate for $\mathfrak{C}$ might be the class of elementary functions of Kalmár.

2. *Operation of the 2-state Universal D-Machine*

Suppose the Turing machine to be simulated has instantaneous description $S_1q_3S_2S_3$ and quintuples

$$\langle q_3 , S_2 , S_4 , R, q_2\rangle, \qquad \langle q_2 , S_3 , S_5 , L, q_1\rangle, \qquad \langle q_1 , S_4 , S_6 , L, q_1\rangle.$$

The successive instantaneous descriptions of the Turing machine will be:

(1)		S_1	q_3	S_2		S_3	
(2)		S_1		S_4	q_2	S_3	
(3)		S_1	q_1	S_4		S_5	
(4)	q_1	S_1		S_6		S_5	(final)

The successive instantaneous descriptions of the 2-state D-machine will be the following:

(1)		$S_{1,0,+,R}$	q_1	$S_{2,3,-,R}$		S_3	
	q_1	$S_{1,0,+,R}$		$S_{4,2,+,R}$		S_3	
		S_1	q_1	$S_{4,2,+,R}$		S_3	
		S_1		$S_{4,2,+,R}$	q_2	S_3	
		S_1	q_2	$S_{4,2,+,R}$		$S_{3,0,-,R}$	
		S_1		$S_{4,1,+,R}$	q_2	$S_{3,0,-,R}$	
		S_1	q_2	$S_{4,1,+,R}$		$S_{3,1,-,R}$	
		S_1		$S_{4,0,+,R}$	q_2	$S_{3,1,-,R}$	
		S_1	q_2	$S_{4,0,+,R}$		$S_{3,2,-,R}$	
(2)		S_1		$S_{4,0,+,R}$	q_1	$S_{3,2,-,R}$	
		S_1	q_1	$S_{4,0,+,R}$		$S_{5,2,+,L}$	
		S_1		S_4	q_1	$S_{5,2,+,L}$	
		S_1	q_1	S_4		$S_{5,1,+,L}$	
		S_1		$S_{4,0,-,L}$	q_1	$S_{5,1,+,L}$	
		S_1	q_1	$S_{4,0,-,L}$		$S_{5,0,+,L}$	
		S_1		$S_{4,1,-,L}$	q_1	$S_{5,0,+,L}$	
(3)		S_1	q_2	$S_{4,1,-,L}$		$S_{5,0,+,L}$	
		S_1		$S_{6,1,+,L}$	q_2	$S_{5,0,+,L}$	
		S_1	q_2	$S_{6,1,+,L}$		S_5	
	q_1	S_1	q_1	$S_{6,1,+,L}$		S_5	
		$S_{1,0,-,L}$		$S_{6,1,+,L}$		S_5	
	q_1	$S_{1,0,-,L}$		$S_{6,0,+,L}$		S_5	
		$S_{1,1,-,L}$	q_1	$S_{6,0,+,L}$		S_5	
(4)	q_2	$S_{1,1,-,L}$		$S_{6,0,+,L}$		S_5	(final)

RECEIVED FEBRUARY, 1965