# A VARIANT OF TURING MACHINES REQUIRING PRINT INSTRUCTIONS ONLY

J. W. SWANSON

## I

## MACHINE ALGORITHMS

1. *Characteristics of Machine Algorithms.* In the thirty some odd years of its existence there has developed a virtual torrent of algorithmic devices adequate to recursive function theory. Prominent among these has been a certain class of formalisms developed on the basis of the *heuristic fiction* of a machine. The first machine presentation of effective computability was that of Turing [9]. But subsequent to the development of Turing machines there have appeared both simplifications and variations on Turing's original basic notions. Thus, for example, Turing initially framed the instructions for his machines in terms of three kinds of ordered quintuples:

$$q_i\ S_j\ S_k\ L\ q_r,$$
$$q_i\ S_j\ S_k\ R\ q_r,$$
$$q_i\ S_j\ S_k\ N\ q_r,$$

representing the following three machine instructions: "When in internal state $q_i$ scanning symbol $S_j$, replace $S_j$ with $S_k$, shift left (or right, or no shift), and go into internal state $q_r$". It was easily demonstrated subsequently that the third, or "no shift" instruction, was eliminable in favor of the other two. And Emil Post [6] soon showed that ordered quadruples were adequate to prescribe Turing machine computations — ordered quadruples of the forms:

$$q_i\ S_j\ S_k\ q_r,$$
$$q_i\ S_j\ L\ q_r,$$
$$q_i\ S_j\ R\ q_r.$$

Patrick Fisher [3], with his U-machines, further reduced these to

ordered triples of the form $q_i S_j X$, where X is either L, R, a symbol $S_k$ or a state $q_r$.

Now it is an interesting exercise in pure logic to ask the following question: To what extent can machine instructions be simplified for formalisms of this type — i.e. for formalisms based on the fictional notion of a machine? Since the instructions for algorithmic devices of this sort represent the basic operations of fictional machines — scanning a tape, printing or erasing on it, shifting left or right, etc. — it must turn out that the degree of simplicity of the machine instructions will depend very much on the nature of these basic operations. Moreover, results toward simplification of instructions, or of basic operations, or both, should not be altogether devoid of interest, since a reduction of fundamental machinery usually enhances the possibility of getting new unsolvability results. The rule of thumb here is that the less you start with, the easier it is to show that you can*not* do certain things. With this end in mind, we shall describe below a variant of Turing machines in which the basic operations of the machine have been altered in such a fashion that the machine instructions are reduced to certain kinds of scan and print commands. But first let us make a few remarks on the general nature of the components in the fictional machines employed in heuristic descriptions of machine algorithms.

2. *Three parts of machines.* Following a description of Turing machines by Claude Shannon [7], let us conceive the fictional machines we are concerned with as consisting of three parts — a reading-printing head called a "scanner", a control element, and a potentially infinite tape (i.e. finite at any moment of the machine's operation, but indefinitely expandable). In the normal description of a Turing machine, for example, the scanner is conceived of as reading, at any moment of the machine's operation, some symbol $S_j$ (from some fixed alphabet for each machine) in one square of the tape, the machine being at that moment in some particular internal state $q_i$. The control element, which corresponds to the machine instructions (i.e. Turing quintuples), contains commands for certain state-symbol configurations, but not for others. When the scanner reaches a square with symbol $S_j$ while in state $q_i$ for which the control element contains no instruction beginning $q_i S_j \ldots$, then the machine

halts. The abstract formalism for such machines is thus contained entirely in the machine instructions. These make use of symbols '$S_i$', '$S_j$', etc. from the machine, alphabet, change of direction symbols 'R' and 'L', and state symbols '$q_i$', '$q_j$', etc. Although some recent departures from the "classical" form of Turing machines — notably the ingenious B-machines of Hao Wang [10] and the SS machines of Shepherdson and Sturgis [8] — have eliminated the notion of states in favor of conditional transfers, they nevertheless make use in their stead of an *ordered* set of instructions, as opposed to the unordered sets of quintuples in the original Turing version.

What suggests itself at this point is that the simplest sort of machine instructions would be a set of unordered print instructions only, making use of neither shift commands nor of states (internal configurations). In the sequel, we shall show that such a machine can easily be described, using a uniformly moving circular tape ([1]) to eliminate shift instructions, and slightly changing the role of the printing key in the scanner in order to eliminate machine states.

## II

### Universal Machines with Print Instructions Only

1. *Description of P-machines.* We now describe a type of machine, the P-machine, designed to simulate Turing machines, and such that its only instructions are print commands. The need for change of direction instructions and for internal states will be obviated respectively by the use of a constantly rotating finite (but indefinitely expandable) circular tape, and by the employment of a variable printing key capable of printing simultaneously either in the two squares consisting of the square under scan and the one to its imme-

([1]) We first encountered the notion of circular tapes in discussion with Mr. Richard Call, who has shown in an unpublished proof (1963) that by employing a circular tape it is possible to rewrite the instructions for Wang's B-machines, eliminating one of the two direction shifts. The idea of circular tapes, however, has long been current in the literature. Cf. Moore [5], Hooper [4], Elgot and Rutledge [2], and Arbib [1].

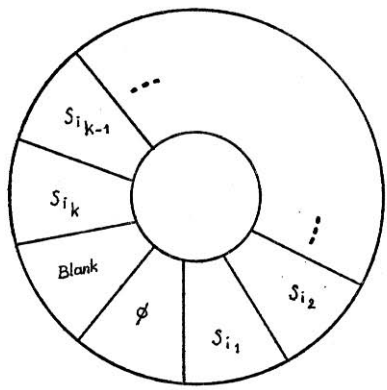
diate left, or else in the two squares consisting of the square under scan and the one to its immediate right.

More exactly, a P-machine is to be conceived as consisting of the following three parts. (1) A circular tape, finite at any moment of the machine's operation, but capable of indefinite expansion in the following manner. At every moment of the machine's operation at which the printing key writes a non-null superscript, the tape expands by the insertion of one blank square immediately to the left of the special symbol 'Ø', or one of its superscripted variants (see below). Further, we suppose that after every printing operation the tape rotates automatically one square in a counter-clockwise direction. This would be equivalent to the printing-scanning head moving uniformly, after every printing operation, one square to the right. It is somewhat easier to talk as if it were this part of the machine which moved, rather than the tape, and we shall do so in the sequel. The other parts are (2) a reading — printing head, the scanner, capable of recognizing the symbol in the square under scan and of either printing in the scanned square and its left-most neighbor, or else in the scanned square and its rightmost neighbor; all of this on the basis of commands from (3) the control element, embodying the print instructions of the P-machine.

2. *Operation of P-machines.* The sole function of P-machines is to simulate the computations of Turing machines assumed to be described in terms of left-shift and right-shift quintuples only (see above). It will simplify our description, however, if we assume that every Turing machine which we wish to simulate with a P-machine has been rewritten as a Turing machine T′ in which the special symbol Ø appears initially on the tape in the square immediately to the left of the input word W, and such that in all subsequent computations, Ø is retained as the left-most symbol. Thus if for input W the machine T produces W′ upon termination, then, the initial word on the tape of T′ will be ØW, and on termination ØW′. Any intermediate word $W_i$ at $t_i$ (the $i^{th}$ moment of T's computation) will have $ØW_i$ as analogue in the computation of T′.

Suppose, then, that we wish to simulate an m — 1 symbol n-state Turing machine T with alphabet $A = \{S_1, S_2, ..., S_{m-1}\}$ (including a symbol for the "blank" square), and states $q_1, q_2, ..., q_n$. Then

first we construct the m-symbol, n-state Turing machine T′ just described. Let us assume that T′ starts in initial state $q_1$ with ØW on its tape, where $W = S_{i1} S_{i2} \ldots S_{ik}$ and the symbol under scan is $S_{i1}$. Then the machine P will begin with $ØS^{1i}{}_1 S^{i}{}_2 \ldots S^{i}{}_m$ on its closed tape, with one blank square immediately to the left of Ø, as follows




P's starting tape

Thus P's initial tape entry differs from that of T′ (apart from the shape of the tapes) only in having $S^{1i}{}_1$ for $S_{i1}$, where the superscript '1' obviously acts as proxy for the machine state $q_1$. We shall show that at any moment of P's operation at which it initiates the simulation of a Turing quintuple, the tape of P will differ from that of T′ in just this fashion, i.e. in being identical save for the square under P's scanner, which will differ from the square under the scanner of T′ only in having the additional superscript 'i' corresponding to the state $q_i$ of T′ at that moment. By inductive assumption then, if at any moment of its operation, initiating the simulation of a quintuple, T′ is in state $q_i$ scanning symbol $S_j$, then P is scanning the symbol $S^{i}{}_j$, $1 \leq i \leq n$, $1 \leq j \leq m$. It is obvious that the alphabet of P must contain $m + mn$ symbols — i.e. the original m symbols of T′, plus mn symbols $S^{i}{}_j$, $1 \leq i \leq n$, $1 \leq i \leq m$. This gives the essence of the P-machine's operation. It remains, of course, to describe the instructions which prescribe this operation.

As already indicated, P has only to simulate left-shift and right-shift Turing quintuples. The right-shift is easy. For every Turing quintuple $q_i\ S_j\ S_k\ R\ q_r$ of T', P will contain the instruction $S^i{}_j,\ S_k{}^r$. This basic instruction of the P-machine tells the scanner to replace the symbol $S^i{}_j$ (scanned say, at time t) with $S_k$, and to print an r superscript in the square to the immediate right. Since the scanner automatically shifts rightward (tape leftward) after every printing operation, at time $t + 1$ the P-machine will be scanning the symbol $S^r{}_x$, corresponding to T' being in state $q_r$ scanning some symbol $S_x$.

Simulation of left-shift quintuples $q_i\ S_j\ S_k\ L\ q_r$ are only a bit more difficult to describe than the right shift quintuples. Here the strategy will be to mark the new state symbol r as a superscript in the square adjacent to the left of the one under scan, then wait for the regular rightward movement of the scanner (assured by "mark-time" print instructions — see below) to bring it back around to that superscripted square. Thus, for every left-shift quintuple $q_i\ S_j\ S_k\ L\ q_r$ of T', P must contain the instructions $S^i{}_j,{}^r\ S_k$. In addition, however, we must include m instructions $S_i$, $S_i\Lambda$ (where '$\Lambda$' stands for the null superscript, or no-print command, and $1 \leq i \leq m$) which allows P to "mark-time", as it were, waiting for the regular rightward movement of its scanner to bring it back around to $S^r{}_x$. Thus P will exactly simulate the computations of T', halting when (and only when) T' does.

We conclude with the following conjecture. It should not prove too difficult to further simplify the instructions for the P-machine, eliminating the necessity for the superscripting operation by allowing a saccadic or tidal scanner motion — e.g. two forward, one back, two forward, one back... etc. Such a uniform two-right one-left movement of the scanner should provide for the necessary rotation of the tape while at the same time allowing for the perhaps rather complex simulation of left-shift Turing quintuples.

*University of Massachusetts* J. W. SWANSON

## REFERENCES

[1] M. A. ARBIB, "Monogenic systems are universal", *The Journal of the Australian Mathematical Society*, vol. 3, Part 3 (1963), pp. 301-306.

[2] C. C. ELGOT and J. D. RUTLEDGE, "RS-machines with almost blank tape", *Journal ACM*, vol. 11, no. 3 (July, 1964), pp. 313-337.
[3] Patrick C. FISHER, "On formalisms for Turing machines", *Journal ACM*, vol. 12, no. 4 (October, 1965), pp. 570-580.
[4] P. K. HOOPER, "Some small multi-tape universal Turing machines", *The American Mathematical Society Notices*, vol. 10, no. 6 (October, 1963).
[5] E. F. MOORE, "A simplified universal Turing machine", *Proceedings ACM* (September 8-10, 1952), photo-offset, Sauls Lithograph Company, pp. 50-55.
[6] E. L. POST, "Recursive unsolvability of a problem of Thue", *The Journal of Symbolic Logic*, vol. 12, no. 1 (March, 1947), pp. 1-11.
[7] Claude E. SHANNON, "A universal Turing machine with two internal states", *Automata Studies* (Princeton, Princeton University Press), 1956.
[8] J. C. SHEPHERDSON and H. E. STURGIS, "Computability of recursive functions", *Journal ACM*, vol. 10 (1963), pp. 217-255.
[9] A. M. TURING, "On computable numbers, with an application to the Entscheidungsproblem", *Proceedings London Mathematical Society*, ser. 2, vol. 42 (1936-37), pp. 230-265; Correction, *ibid.*, 43 (1937), pp. 544-546.
[10] Hao WANG, "A variant to Turing's theory of calculating machines", *Journal of the Association for Computing Machinery*, vol. 4 (1957), pp. 63-92.