Universality* of Tag Systems With $P = 2$

JOHN COCKE

International Business Machines Corp.,† *Yorktown Heights, N. Y.*

AND

MARVIN MINSKY

Massachusetts Institute of Technology,‡ *Cambridge, Mass.*

1. *Summary*

By a simple direct construction it is shown that computations done by Turing machines can be duplicated by a very simple symbol manipulation process. The process is described by a simple form of Post canonical system with some very strong restrictions.

This system is *monogenic*: each formula (string of symbols) of the system can be affected by one and only one production (rule of inference) to yield a unique result. Accordingly, if we begin with a single axiom (initial string) the system generates a simply ordered sequence of formulas, and this operation of a monogenic system brings to mind the idea of a machine.

The Post canonical system is further restricted to the "Tag" variety, described briefly below. It was shown in [1] that Tag systems are equivalent to Turing machines. The proof in [1] is very complicated and uses lemmas concerned with a variety of two-tape nonwriting Turing machines. The proof here avoids these otherwise interesting machines and strengthens the main result; obtaining the theorem with a best possible *deletion number* $P = 2$.

Also, the representation of the Turing machine in the present system has a lower degree of exponentiation, which may be of significance in applications.

These systems seem to be of value in establishing unsolvability of combinatorial problems.

2. *Tag Systems with Deletion Number* $P = 2$

The Problem of Tag. Let A be a finite alphabet of letters $a_1, \cdots, a_n$. Let W be an associated set of words; for each i, W_i is some fixed string or word of letters of A. Let P be some integer, and consider the following process applied to some initially given string S of the letters.

Examine the first letter of the string S. If it is a_i, then (i) remove from S the first P letters, and (ii) attach to its end the word W_i. Perform the same operation on the resulting string, and repeat the process so long as the resulting string has

* Received May, 1963.

† Watson Research Center.

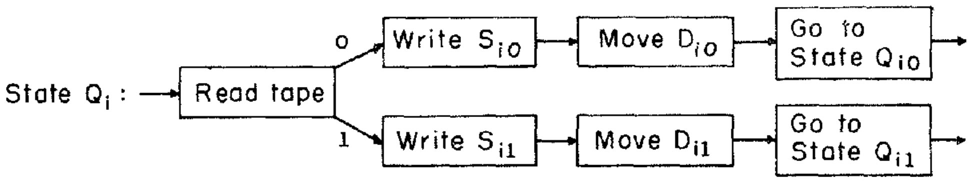
‡ Computation Center and Department of Electrical Engineering.

P or more letters. The "Tag problem," for given A, P and W, is to obtain a decision procedure which, given S, will determine if the process will ever terminate.

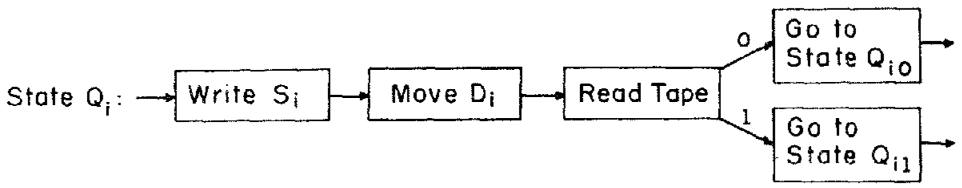
In [1] it is shown that this problem is recursively unsolvable because one can, in effect, simulate the operation of an arbitrary Turing machine through the succession of strings generated by a corresponding Tag system. The present paper accomplishes the same thing, in a much simpler way, with an improved result. We obtain the Turing Machine representation in a Tag system with $P = 2$; that is, the process reads one symbol, erases that and the next, concatenates the corresponding W_i and begins again. It is difficult to imagine a simpler kind of unsolvable word problem.

3. *Representation of a Turing Machine*

The performance of a 2-symbol Turing Machine is usually represented by the procedure [4]:



It is convenient in this paper to use a different but obviously equivalent (and slightly simpler) procedure:

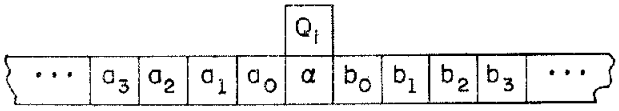


In converting a Turing machine from the first formalism to the second, we have to double the number of states. The advantage is that now the reading operation causes an immediate state-change, and no implicit symbol-memory is required.

In this version, each state is equivalent to a quintuple:

State	Write	Move	Read α If $\alpha = 0$ Go to	If $\alpha = 1$
Q_i	S_i	D_i	Q_{i0}	Q_{i1}

An instantaneous description of a Turing Machine computation must specify the entire contents of the tape, the machine's present location on the tape, and the machine's present internal state. Such a description can be represented as



where α is the digit on the scanned square, the a_i's and b_i's are the digits to the left and right of the scanned square, and Q_i is the machine's state *just after* reading the symbol α. (Hence α is redundant and need not be included in the description.) If we regard the digits a_i and b_i as binary digits we can condense the instantaneous description to a triple:

$$(Q, M, N) = (Q_i, \sum_1^\infty a_i 2^i, \sum_1^\infty b_i 2^i).$$

Since all but a finite portion of the tape contains blanks (0's), the sums are always defined.

Because such a triple is a complete instantaneous description, the machine's structure can be described in terms of the way these triples are transformed from each moment to the next. Suppose, for example, that D_i means "move right." Then

$$M \leftarrow 2M + S_i$$

$$N \leftarrow \left[\frac{N}{2}\right], \quad \text{i. e. the largest integer} \leqq \frac{N}{2},$$

and the new Q is Q_{i0} or Q_{i1} according to whether N was even or odd.

If D_i means "move left," we have only to reverse the roles of M and N in this transformation.

The task is to show how to construct a Tag system equivalent, in a suitable sense, to any Turing Machine. The above shows that we have only to find a way to realize the above transformation involving M, N and the Q_i's.

4. *Construction of the Equivalent Tag System*

Given a two-symbol Turing Machine with states $Q_1, \cdots, Q_i, \cdots, Q_r$, we define a Tag system with symbols x_i, A_i, α_i, B_i, β_i, C_i, c_i, D_{i0}, d_{i0}, D_{i1}, d_{i1}, S_i, s_i, T_{i0}, t_{i0}, T_{i1}, t_{i1} ($i = 1, \cdots, r$). The subscripts i correspond to the internal states of the machine.

For expository purposes, the words (see Section 2) associated with each letter are displayed in boxes interspersed with the steps of an example. The example below shows how the process carried out by the Tag system corresponds to the transformation (see Section 3) of one instantaneous description (Q, M, N) of the Turing Machine to the next. The example illustrates a state which moves right. The other case is obtained *mutatis mutandis.*

The instantaneous description (Q_i, M, N) of the Turing Machine is represented by the letter-string $A_ix_i(\alpha_ix_i)^M B_ix_i(\beta_ix_i)^N$, where the superscripts M and N mean that the bracketed strings are repeated M and N times, respectively.

For typographical reasons we drop the state-subscripts that occur on each letter. Thus the above string is written simply as

$$Ax(\alpha x)^M Bx(\beta x)^N. \tag{1}$$

The Tag words associated with A_i and α_i will be

$$\boxed{\begin{matrix}A \rightarrow Cx\\ \text{OR}\\ A \rightarrow Cxcx\end{matrix}} \quad \text{AND} \quad \boxed{\alpha \rightarrow cxcx}$$

depending on whether S_i is 0 or 1; that is, on whether M is to be converted to $2M$ or to $2M + 1$; This depends only on the internal state. The application of these rules leads eventually to

$$Bx(\beta x)^N Cx(cx)^{M'} \tag{2}$$

where M' is $2M$ or $2M + 1$. Next,

$$\boxed{B \rightarrow S} \qquad \boxed{\beta \rightarrow s}$$

yields, from (2),

$$Cx(cx)^{M'} S(s)^N. \tag{3}$$

Next,

$$\boxed{C \rightarrow D_1D_0} \qquad \boxed{c \rightarrow d_1\,d_0}$$

yields

$$S(s)^N D_1D_0(d_1d_0)^{M'}. \tag{4}$$

The rules for S and s are

$$\boxed{S \rightarrow T_1T_0} \qquad \boxed{s \rightarrow t_1t_0}$$

If N happens to be odd, these rules result in

$$D_1D_0(d_1d_0)^{M'} T_1T_0(t_1t_0)^{N-1/2}. \tag{5_1}$$

But if N is even, the last deletion removes D_1, leaving the string

$$D_0(d_1d_0)^{M'} T_1T_0(t_1t_0)^{N/2}. \tag{5_0}$$

This difference in format corresponds to reading the Turing machine tape and thereby determines (as seen below) the next state! We continue with the case of N odd:

$$\boxed{D_1 \rightarrow A_1x_1} \qquad \boxed{d_1 \rightarrow \alpha_1x_1}$$

This yields a string of the form

$$T_1T_0(t_1t_0)^{N-1/2}A_1x_1(\alpha_1x_1)^{M'}. \qquad (6_1)$$

Next,

$$\boxed{T_1 \rightarrow B_1x_1} \quad \boxed{t_1 \rightarrow \beta_1x_1}$$

yields

$$A_1x_1(\alpha_1x_1)^{M'}B_1x_1(\beta_1x_1)^{N-1/2}. \qquad (7_1)$$

Since $(N - 1)/2$ is the integer part of $N/2$, we have arrived at the next instantaneous description of the Turing machine, in the state Q_{i1} that results when the machine reads a one.

Returning to the case of N even, we introduce the rules

$$\boxed{D_0 \rightarrow x_0A_0x_0} \quad \boxed{d_0 \rightarrow \alpha_0x_0}$$

which yield

$$T_0(t_1t_0)^{N/2}x_0A_0x_0(\alpha_0x_0)^{M'}. \qquad (6_0)$$

Note that the form of this string differs from the form of (6_1) in that the '1' subscripts now occur in even positions and are therefore not seen by the remaining processes! Finally, the rules

$$\boxed{T_0 \rightarrow B_0x_0} \quad \boxed{t_0 \rightarrow \beta_0}$$

produce from (6_0) the final string

$$A_0x_0(\alpha_0x_0)^{M'}\ B_0x_0(\beta_0x_0)^{N/2} \qquad (7_0)$$

all of whose letters are associated with the alphabet of Q_{i0}. Thus the flow of the process is controlled by the odd- or evenness of the string lengths.

The strings (7_0) and (7_1) have the form of the string (1), except for the change in subscript. This transformation is just that required to represent the effect of one cycle of the Turing machine's operation. A very small (4-symbol, 7-state) Universal Turing Machine using this theorem is described in [3].

The length of the longest word W_i is 4 letters, namely, in the "$A \rightarrow Cxcx$" and "$\alpha \rightarrow cxcx$" rules. This maximum can be reduced to 3, as Wang shows in [2]. This can be done directly here but we omit the full construction. The key problem is doubling the number represented by a string, which can be done, for example, by

$$\boxed{\begin{array}{ll} A \rightarrow B & \alpha \rightarrow PPQ \\ B \rightarrow CC & P \rightarrow SSS \\ C \rightarrow SA'x & Q \rightarrow SS \\ \multicolumn{2}{c}{S \rightarrow \alpha' x} \end{array}}$$

which takes $Ax(\alpha x)^n$ into $A'x(\alpha'x)^{2n}$.

Many other related results can be found in [4]. The 2-number representation of the Turing machine tape used in Section 3 appears in [5]. Its use here was suggested by Dana Scott.

REFERENCES

1. Minsky, M. Recursive unsolvability of Post's problem of Tag and other topics in theory of Turing machines. *Ann. Math.* *74*, 3 (Nov. 1961), 437–455.
2. For further results along these lines, see:
   Wang, Hao. Tag systems and Lag systems. To appear.
3. Minsky, M. Size and structure of universal Turing machines using Tag systems: a 4-symbol 7-state machine. In *Proc. Symposium on Recursive Function Theory*, Am. Math. Soc., Providence, R. I., 1962.
4. Shepherdson, J. C., and Sturgis, H. E. Computability of recursive functions. *J. ACM*, *10* (Apr. 1963) 217–255.
5. Wang, Hao. A variant of Turing's theory of computing machines. *J. ACM* *4* (Apr. 1957), 63.