10p

Artificial Intelligence Project--RLE and MIT Computation Center
Memo 52--

UNIVERSALITY OF TAG SYSTEMS WITH P = 2

by John Cocke and Marvin Minsky

April 25, 1963

Universality of Tag Systems with P=2

by John Cocke and Marvin Minsky

1. Summary

In the following sections we show, by a simple direct construction, 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. First, the 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 be of 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 non-writing Turing machines. Our 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

2.0. The problem of "Tag". Let A be a finite alphabet of letters $a_1, \ldots, a_n$; and 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

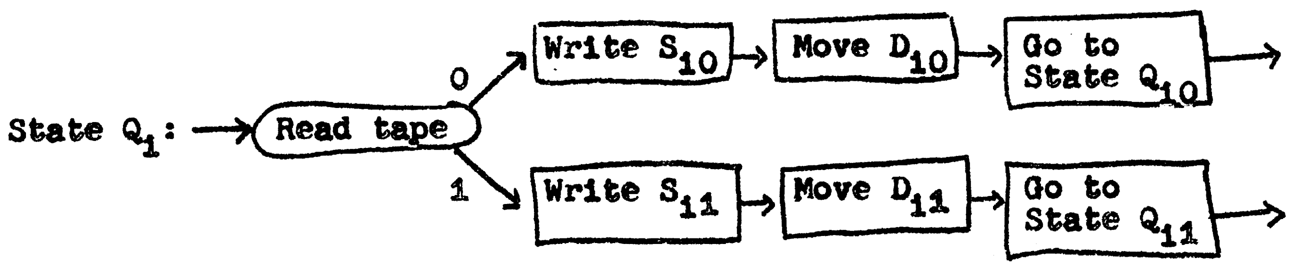
(ii) attach to its end the word W_i.

Perform the same operation on the resulting string, and repeat the process so long as the resulting string has P or more letters. The "Tag problem", for given A, P, and W is to give a decision procedure which, given S, will tell if the process will ever terminate.

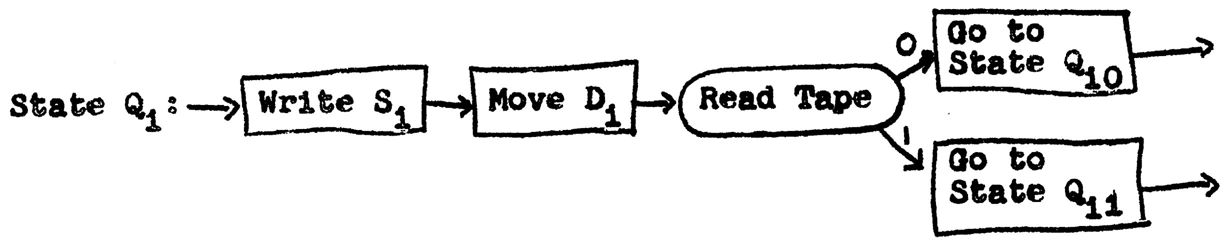
In [1] it is shown that this problem is recursively unsolvable because one can, in effect, simulate the operation of an arbitrary Turing machine through the succession of strings generated by a corresponding Tag system. The present paper accomplishes the same thing, in a much simpler way, with an improved result. Here 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:



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

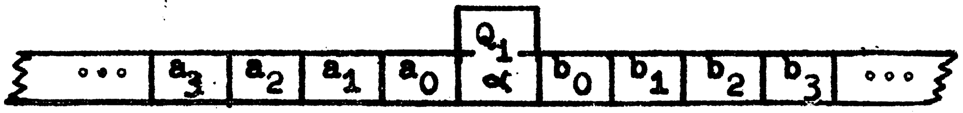


In converting a Turing machine from the first formalism to the second, we have to double the number of states. In a way, however, this new formalism is a little more honest, because in the usual formalism the machine needs an extra memory in which to store the symbol just read, while it writes and then moves. Here, the reading operation causes an immediate state-change, and no implicit symbol-memory is required.

In this version, each state is associated with a quintuple:

State	Write	Move	Read If = 0 Go to	If = 1
Q_1	S_1	D_1	Q_{10}	Q_{11}

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 <u>just after</u> 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:

$$(Q, M, N) = (Q_i, \sum_i^{\infty} a_i 2^i, \sum_i^{\infty} b_i 2^i)$$

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

Since such a triple is a complete instantaneous description, we can describe the machine's structure 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

$$M \leftarrow 2M + s_i$$
$$N \leftarrow [\tfrac{N}{2}], \text{ i.e., the largest integer } \leq \tfrac{N}{2}.$$

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

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

Our 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 $Q_1, \ldots, Q_i, \ldots, Q_r$, we define a Tag system with symbols

$$x_i,\ A_i,\ \alpha_i,\ B_i,\ \beta_i,\ C_i,\ c_i,\ D_{i0},\ d_{i0},\ D_{i1},\ d_{i1},\ S_i,\ s_i,$$
$$T_{i0},\ t_{i0},\ T_{i1},\ t_{i1}. \quad (i = 1, \ldots, r)$$

The subscripts i correspond to the internal states of the machine.

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

The instantaneous description (Q_i, M, N) of the Turing Machine will be represented by the letter-string

$$A_i x_i (\alpha_i x_i)^M B_i x_i (\beta_i x_i)^N$$

where the superscripts M and N mean that the bracketed strings are repeated M and N times respectively.

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

(1) $Ax(\alpha x)^M Bx(\beta x)^N$

The Tag words associated with A_1 and α_1 will be

$$\boxed{\begin{array}{l} A \rightarrow Cx \\ \quad \text{or} \\ A \rightarrow Cxcx \end{array}} \qquad \text{and} \qquad \boxed{\alpha \rightarrow cx}$$

depending on whether S_1 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

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

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

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

yields, from (2),

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

Next,

$$\boxed{C \rightarrow D_1D_0} \qquad \boxed{c \rightarrow d_1d_0}$$

yields

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

The rules for S and s are

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

If N happen to be odd, these rules result in

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

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

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

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:

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

This yields a string of the form

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

Next,

$$\boxed{T_1 \to B_1x_1} \qquad \boxed{t_1 \to \beta_1x_1}$$

yields

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

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_{11} that results when the machine reads a '1'.

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

$$\boxed{D_0 \to x_0A_0x_0} \qquad \boxed{d_0 \to \alpha_0x_0}$$

which yield

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

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

$\boxed{T_0 \rightarrow B_0 x_0}$ $\boxed{t_0 \rightarrow \beta_0}$

produce from (6_0) the final string

$$(7_0) \qquad A_0 x_0 (\alpha_0 x_0)^{M'} \quad B_0 x_0 (\beta_0 x_0)^{2^N}$$

all of whose letters are associated with the alphabet of Q_{10}. 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.

The length of the longest word W_1 is 4 letters, namely, in the "A → Cxcx" rule. This maximum, called "ϵ" by Wang in [2], can be reduced to 3 by replacing the rules associated with A, α, and B (in the $S_1 = 1$ case) by

$\boxed{A \rightarrow Cxc}$ $\boxed{\alpha \rightarrow xc}$ $\boxed{B \rightarrow xS}$

References

[1] M. Minsky, "Recursive Unsolvability of Post's Problem of "Tag" and Other Topics in Theory of Turing Machines," Annals of Mathematics, 74, no. 3, pp. 437-455; November, 1961.

[2] For further results along these lines, see

Hao Wang, "Tag Systems and Lag Systems." To appear in Math. Annalen.

[3] A very small (4 symbol-7-state) Universal Turing Machine, using this theorem is described in

M. Minsky, "Size and Structure of Universal Turing Machines using Tag Systems: a 4-symbol 7-state Machine," Proc. of a Symposium on Recursive Function Theory, Am. Math. Soc., 1962.

[4] The 2-number representation of the Turing machine tape, used in § 3 appears in the paper

H. Wang, "A Variant of Turing's Theory of Computing Machines," J. ACM, 2, p. 63; April, 1957.

Its use here was suggested to us by Dana Scott.

CS-TR Scanning Project
Document Control Form

Date : 11 / 30 / 95

Report # AIM - 52

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☐ Technical Report (TR) ☒ Technical Memo (TM)
☐ Other: ____________

Document Information

Number of pages: 10 (14 - images)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☒ Single-sided or
☐ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☒ Other: MIMEO GRAPH

Check each if included with document:

☐ DOD Form ☐ Funding Agent Form ☐ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: ____________

Page Data:

Blank Pages(by page number): ____________

Photographs/Tonal Material (by page number): ____________

Other (note description/page number):

Description : Page Number:
IMAGE MAP: (1-10) UNH'ED TITLE PAGE, 1-9
(11-14) SCANCONTROL, TRGT'S (3)

Scanning Agent Signoff:
Date Received: 11 / 30 / 95 Date Scanned: 12 / 7 / 95 Date Returned: 12 / 7 / 95

Scanning Agent Signature: Michael W. Cook

Rev 9/94 DS/LCS Document Control Form cstrform.vsd