*Reprinted from* IEEE TRANSACTIONS
ON *ELECTRONIC COMPUTERS*
Volume EC-15, Number 4, August, 1966
Pp. 658-659


# Note on Finite-Memory Sequential Machines

JAMES L. MASSEY, MEMBER, IEEE

## INTRODUCTION

The study of sequential machines with finite memory originated with Zadeh and has been continued by Simon[1] and Gill.[2,3] Following Gill,[3] we define a sequential machine by a finite input alphabet $X$, a finite output alphabet $Y$, a finite state-set $S$, a next-state function $\delta$, and an output function $\lambda$, such that for all integers $t$: $x_t \in X$, $y_t \in Y$, $s_t \in S$, and

$$s_{t+1} = \delta(s_t, x_t)$$
$$y_t = \lambda(s_t, x_t).$$

A sequential machine has *memory* $\mu$ where $\mu$ is the least integer such that

$$y_t = f(x_t, x_{t-1}, \cdots, x_{t-\mu}, y_{t-1}, \cdots, y_{t-\mu}), \tag{1}$$

i.e., such that the current output can always be determined from the current input and the $\mu$ preceding input/output events.

Gill[2] has shown than if $\mu$ is finite and the machine is minimal, then

$$\mu \leqq \tfrac{1}{2}(n)(n-1) \triangleq N \tag{2}$$

where $n$ is the number of states in $S$. Gill[3] has shown further that for any $n$, $n$-state machines with $\mu = N$ can be constructed. However, Gill's construction uses an $N$-letter input alphabet which raises the question whether $\mu$ might be limited by the order of the input alphabet $X$ as well as by the order of the state-set $S$.

In this note we show that the order of $X$ cannot strongly limit $\mu$ by exhibiting a class of ternary-input machines with $\mu = N$ for every $n$. For $n \leqq 5$, we show also that the bound of (2) can be attained using only a binary input alphabet. Before proceeding to these tasks, we first point out a correction that must be made in Gill's proof of (2).

## BOUND ON FINITE MEMORY

In proving the bound (2), Gill[4] states that if a minimal machine has finite memory $\mu = m$, then there must exist two distinct states, say $\sigma_{i_1}$ and $\sigma_{j_1}$, and an input sequence $x_1, x_2, \cdots, x_m$ which causes both the state-sequence $\sigma_{i_1}, \sigma_{i_2}, \cdots, \sigma_{i_m}, \sigma_{i_{m+1}}$ with output sequence $y_{i_1}, y_{i_2}, \cdots, y_{i_m}$, and the state sequence $\sigma_{j_1}, \sigma_{j_2}, \cdots, \sigma_{j_m}, \sigma_{j_{m+1}}$ with output sequence $y_{j_1}, y_{j_2}, \cdots, y_{j_m}$ such that

$$\left.\begin{aligned} y_{i_k} &= y_{j_k} \\ \sigma_{i_k} &\neq \sigma_{j_k} \end{aligned}\right\} \quad k = 1, 2, \cdots, m$$
$$\sigma_{i_{m+1}} = \sigma_{j_{m+1}}. \tag{3}$$

Gill argues that (3) is a necessary condition for $\mu = m$ since there must be at least one circumstance in which the state becomes predictable after exactly $\mu$ input/output events, and (3) is the only possible such circumstance. There is a second possibility, however, namely

$$\left.\begin{aligned} y_{i_k} &= y_{j_k} \\ \sigma_{i_k} &\neq \sigma_{j_k} \end{aligned}\right\} \quad k = 1, 2, \cdots, m-1$$
$$\sigma_{i_m} \text{ and } \sigma_{j_m} \text{ totally non-equivalent} \tag{4}$$

where by *totally non-equivalent* we mean that

$$\lambda(\sigma_{i_m}, x) \neq \lambda(\sigma_{j_m}, x) \qquad \text{all } x \in X,$$

i.e., that no input causes the same output for both states. Under condition (4), it must happen that $y_{i_m} \neq y_{j_m}$ and this in turn suffices to determine whether $\sigma_{i_1}$ or $\sigma_{j_1}$ was the initial state. Thus, under (4) the state can become predictable after $m$ input/output events and not before. Conditions (3) and (4) together exhaust the ways that the state can become predictable after $m$ input/output events and not before, since there must be some pair of distinct initial states with a common input/output sequence of length at least $m-1$ or else the initial state could always be determined after at most $m-1$ input/output events.

The reader may verify that the machine defined in Fig. 1 has memory $\mu = 3$ and fails to satisfy condition (3). Condition (4) is satisfied, however, for the initial states $\sigma_1$ and $\sigma_2$ and the input sequence 0, 1 since the final states $\sigma_1$ and $\sigma_3$ are totally non-equivalent. The reader may also verify that when conditions (3) and (4) are both included, the remainder of Gill's proof of the bound (2) is unaltered.

We remark further that the bound (2) applies *whether or not* the machine is minimal. For if a non-minimal machine $M'$ with $n'$ states has finite memory $\mu' > \frac{1}{2}(n')(n'-1)$, then its equivalent minimal form $M$ with $n < n'$ states also has finite memory $\mu = \mu' > \frac{1}{2}(n')(n'-1) > \frac{1}{2}(n)(n-1)$ contradicting the fact that (2) is a valid bound for minimal machines. By similar reasoning, if $\mu = \frac{1}{2}(n)(n-1) = N$ for any $n$-state machine, then that machine must be minimal.

It should be emphasized that, by itself, satisfaction of condition (3) or (4) is *sufficient* only to establish that $\mu \geqq m$. Our approach to finding machines for which $\mu = N$ will be to show first that condition (3) or (4) is satisfied for $m = N$ so that $\mu \geqq N$, and second that the memory is finite and hence $\mu \leqq N$.

| | NEXT STATE SECTION | | OUTPUT SECTION | |
|---|---|---|---|---|
| | 0 | 1 | 0 | 1 |
| $\sigma_1$ | $\sigma_2$ | $\sigma_1$ | 0 | 1 |
| $\sigma_2$ | $\sigma_3$ | $\sigma_3$ | 0 | 0 |
| $\sigma_3$ | $\sigma_1$ | $\sigma_1$ | 1 | 0 |

Fig. 1. Transition table for 3-state machine with memory $\mu = 3$ satisfying condition (4) but not condition (3) for $m = 3$.

## TERNARY MACHINES WITH $\mu = N$

For convenience, let

$$X = \{0, 1, 2\}, \quad Y = \{0, 1\}, \quad \text{and} \quad S = \{1, 2, \cdots, n\}$$

where it will be clear from the context whether an integer belongs to $X$, $Y$, or $S$. Consider the class of such $n$-state, ternary-input machines defined by the transition diagrams in Fig. 2, where for clarity the transitions for each of the three possible inputs are shown separately.

*Lemma 1:* For the machines defined in Fig. 2, condition (4) can be satisfied for $m = N$ with states 1 and 2 as the initial states.

The validity of this lemma follows from the transition chains shown in Fig. 3. The final states are totally non-equivalent as may be seen directly from examination of Fig. 2.

*Lemma 2:* For the machines defined in Fig. 2, $s_{t+1} = 1$ whenever $y_t = 1$.

This follows trivially from the defining diagrams in Fig. 2 since every output of 1 is associated with a transition to state 1.

*Lemma 3:* For the machines defined in Fig. 2, given any pair of distinct initial states, there is no input sequence of length $N$ or

Manuscript received September 27, 1965; revised April 22, 1966. This work was supported by the National Science Foundation under Research Grant GP-2547.
The author is with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, Ind.
[1] J. M. Simon, "A note on memory aspects of sequence transducers," *IRE Trans. on Circuit Theory*, vol. CT-6, pp. 26–29, March 1959.
[2] A. Gill, *Introduction to the Theory of Finite-State Machines*. New York: McGraw-Hill, 1962, pp. 156–159.
[3] ——, "On the bound to the memory of a sequential machine," *IEEE Trans. on Electronic Computers*, vol. EC-14, pp. 464–466, June 1965.
[4] ——, *Introduction to the Theory of Finite-State Machines*. New York: McGraw-Hill, 1962, pp. 161–162.

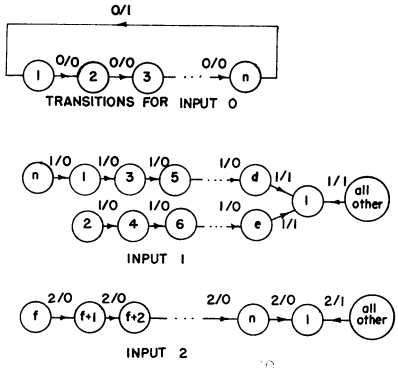


Fig. 2. Transition diagrams for ternary-input machines with memory $\mu = N$. Notes: 1) $d = [n/2]$ if odd, otherwise $d = [n/2]+1$. 2) $e = [n/2]$ if even, otherwise $e = [n/2]+1$. 3) $f = [n/2]+1$, where the brackets denote the integer part of the enclosed number.

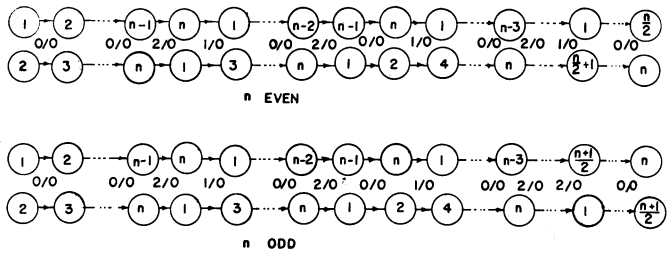





Fig. 3. State transition chains used in the proofs of lemmas 1, 2 and 3.

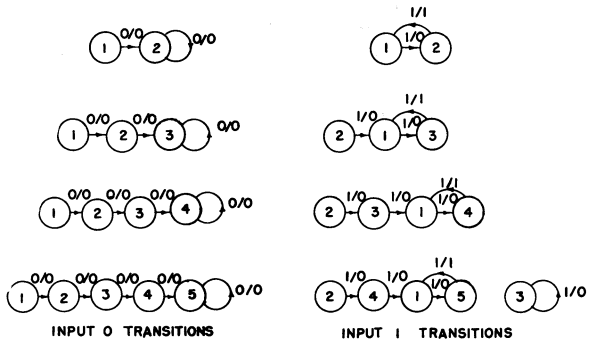


Fig. 4. 2-, 3-, 4-, and 5-state binary-input machines with memory $\mu = N$.

greater that can cause both output sequences to be the all-zero sequence.

*Proof:* Consider the sequence of $N$ pairs of distinct states (1, 2), (2, 3), (3, 4), $\cdots$ formed by taking pairs of states at the same depth in the transition chains of Fig. 3. [The pairs are here considered to be unordered, i.e., (1, 2) is the same pair as (2, 1).] This sequence contains each distinct pair once and only once. All outputs in Fig. 3 are zeros and there is no input letter which can be applied to the final states in both chains to again cause both outputs to be zeroes. We will prove the lemma by showing that any sequence of state pairs, commencing with an arbitrary pair $(a, b)$ of distinct states, obtained by applying an input sequence to both initial states such that *all* outputs are zeros, can be derived from the sequence obtained from Fig. 3 by *deleting* state pairs in that sequence. Thus the input sequence can be no longer than that in Fig. 3 and this length is $N-1$.

Assume now that $(p, q)$ is the first state pair in the sequence commencing with $(a, b)$ whose successor pair cannot be obtained by deleting pairs after $(p, q)$ in the original sequence, i.e., the one obtained from Fig. 3. Let $x_1$ be the input applied to $(p, q)$ in the original sequence and let $x_2$ be the input applied to $(p, q)$ in the second sequence. Clearly, $x_1 \neq x_2$.

Suppose that $x_1 = 1$. Then, from Fig. 3, $(p, q)$ is the pair $(n, r)$ where $r$ is less than the integer part of $n/2$. $x_2$ cannot be 0 since it would then cause an output 1 when applied to state $n$. Also $x_2$ cannot be 2 since it would then cause an output 1 when applied to state $r$. Hence $x_2 = 1$ which contradicts $x_1 \neq x_2$. We conclude that $x_1 \neq 1$. By a similar argument, we can conclude that $x_1 \neq 2$.

We conclude then that $x_1 = 0$ and note from Fig. 3 that neither $p$ nor $q$ can then be state $n$. Since $x_1 = 0$, the successor of $(p, q)$ in the original sequence is the pair $(p+1, q+1)$. But if $x_2 = 2$ *and* the output is 0, it follows from Fig. 2 that the successor of $(p, q)$ in the second sequence is also $(p+1, q+1)$ which contradicts the definition of $(p, q)$. Hence $x_2 \neq 2$. Similarly, if $x_2 = 1$ *and* the output is 0, the successor of $(p, q)$ in the second sequence must be $(p+2, q+2)$ which is always the second successor of $(p, q)$ in the original sequence and thus could be obtained by deleting the pair $(p+1, q+1)$ in the original sequence. Thus, again by contradiction, $x_2 \neq 1$. We conclude that $x_2 = 0$, but this in turn contradicts $x_1 \neq x_2$. The only remaining conclusion is that no $x_1$ exists. Thus the successor of $(p, q)$ in the second sequence can always be obtained by deletion in the original sequence, and the lemma is now proved.

Lemma 1 establishes that the machines defined in Fig. 2 have memory at least $N$. Lemmas 2 and 3 together establish that the memory is finite and hence no more than $N$; for given $N$ past input/output events, either some 1 output occurs which by lemma 2 uniquely identifies the state, or else only 0 outputs occur in which case according to lemma 2 the state is again uniquely determined. Thus, we have proved:

*Theorem:* For every $n$, there exists a ternary-input, binary-output, $n$-state machine with memory $\mu = \frac{1}{2}(n)(n-1) = N$.

## Remarks

The only outstanding question concerning the bound (2) is now whether the bound can be improved for a binary-input machine. For $n \leqq 5$, the binary-input machines in Fig. 4 attain the bound (2) as can be verified exhaustively or by the type of argument used in the proof of lemma 3. For the case $n = 6$, we have been unable to find a binary-input machine with $\mu = N = 15$ and we conjecture that no such machine exists.