ON THE SYNTHESIS OF FINITE SEQUENTIAL MACHINES*

By C. V. Srinivasan and R. Narasimhan
(*Tata Institute of Fundamental Research, Bombay*)

Received December 10, 1958
(Communicated by Dr. R. K. Asundi, F.A.SC.)

Abstract

Starting from the studies of Kleene and Mealy on sequential machines, in this paper is presented a formalism which, in a sense, unifies their treatments. From the specification of the required machine behaviour in terms of events and associated output states, a uniform procedure is given for obtaining a transition table and from that a minimal machine, whenever such a complete reduction is possible. The various steps of the synthesis procedure are so stated that they can be easily programmed on a computer.

1. Introduction

In recent years, sequential machines have been extensively studied under the guise of nerve nets,[1] switching networks[2, 3, 4] and automata.[5] For the present purpose, we take as our point of departure the presentations of Kleene[1] and Mealy[4] and make free use of the concepts introduced by them wherever convenient. In several respects these two papers complement each other and, together, go a long way towards the solution of the synthesis problem, as we shall presently explain.

A sequential machine is described in terms of a finite set of input states, a finite set of output states and a finite set of internal states. It is assumed that the *present output state* (O_t) and the next *internal state* (S_{t+1}) are uniquely determined by the *present internal state* (S_t) and the *present input state* (I_t). Here, t is the time parameter and it is understood that the machine is synchronous, in that, changes in the states take place at discrete instants of time t_i ($i = 1, 2, 3, \ldots$).

* Since we shall be concerned only with finite sequential machines in this paper, we shall henceforth drop the attribute 'finite'.

When the first draft of this was ready, we received a copy of the paper "Analysis of Sequential Machines—I" by D. D. Aufenkamp and F. E. Hohn. Aufenkamp and Hohn are also concerned with the development of a formalism in terms of which the analysis of a sequential machine can be readily mechanised. However, their approach is entirely different from ours. We are grateful to Professor Hohn for making a copy of the paper available to us.

The problem of analysis of a given sequential machine is one of a *complete* description of its behaviour. This can be done by either of two equivalent methods:

(1) The first method makes use of a Transition table or a state diagram which specifies exhaustively for each pair of present internal—present input states, what the associated pair of next internal—present output states is. This is the approach taken by Mealy.

(2) The second method makes use of the notion of events. Events are input state sequences of finite length. For definiteness, assume that the sequential machine starts with a distinguished internal state. Then its complete behaviour is described by giving its output for each possible event. This is the method followed by Kleene. In his paper, he has answered *in principle* the basic question, what classes of events are representable by sequential machines? The formalism, developed with this in view, allows him to suggest a solution to the synthesis problem, *viz.*, how to construct a sequential machine to represent a given event? However, his solution to this is mostly of academic interest.

Neither of these two methods by itself seems to be adequate enough to answer satisfactorily the synthesis problem. In the synthesis of sequential machines one has three major aspects to contend with, *viz.*,

(*a*) specification of the required behaviour,

(*b*) translation of the specification to a state diagram and the reduction of equivalent internal states, and

(*c*) translation of the reduced state diagram to hardware.

Mealy, in extending the works of Huffman and Moore, confines his consideration almost exclusively to aspects (*b*) and (*c*), *i.e.*, given a transition table or, equivalently, a state diagram, he develops procedures to reduce it to its minimal form and to realise it in terms of circuitry. Kleene, on the other hand, is concerned primarily with aspect (*a*) of the problem, *i.e.*, the development of adequate concepts to describe (and hence, equivalently, to specify) the machine behaviour in terms of its input states and output states.

Thus, it would seem natural to look for a unified treatment which would exploit equally methods 1 and 2 described earlier and, hence, treat adequately all the 3 aspects of the synthesis problem. Clearly, a prerequisite for such a unification is the availability of a simple procedure which will enable one to go from a description of machine behaviour by method 2 to a description of its behaviour by method 1.

In this paper we present a formalism which enables one to proceed from a specification of the required machine behaviour in terms of events and associated outputs to a transition table. From this initial transition table a reduction table is formed which, together with a reduction algorithm, allows one to arrive at the minimal equivalent machine whenever such a complete reduction is uniquely possible. We do not consider in this paper aspect (*c*) of the synthesis problem which calls for the realization of the reduced machine in terms of hardware.

The formalism is developed such that the various steps in the synthesis procedure can be readily mechanised or programmed on a computer. This is an essential aspect of the problem in practice, since, without some such mechanical assistance, the synthesis of large, complex sequential machines will be almost impossible.

2. CONVENTIONS

Following the customary practice, we shall assume that the input, output and internal state organs of the machine are binary state devices, and refer to them as flip-flops. Thus, let the sequential machine have n input flip-flops and m output flip-flops. Then there are 2^n possible input states and 2^m possible output states. Call the former set I and index its members $(I_0, I_1, \ldots, I_{2^n-1})$; call the latter set Φ and index its members $(\phi_0, \phi_1, \phi_2, \ldots, \phi_{2^m-1})$.

A k-sequence is a sequence of input states of length k. For a given k, every k sequence can be written as a k-tuple whose components read from left to right constitute the input states in that k-sequence. The 1-sequences are the individual input states.

For each fixed k, the k-tuples are ordered according to the lexicographic ordering of their components. Thus, for each k, the first k-tuple is $(I_0, I_0, \ldots, I_0)$ and the last $(I_{2^n-1}, I_{2^n-1}, \ldots, I_{2^n-1})$. There are 2^n 1-tuples, 2^{2n} 2-tuples and in general, 2^{kn} k-tuples.

The integer N is a given parameter for the problem. N is the maximum length of an input sequence identified by the sequential machine as an event. Hence N is a measure of the finiteness of the memory of the machine.

The set of input sequences of lengths k, for $k = 1, \ldots.$ N, has

$$\mathcal{J}(\mathrm{N}) = 2^n + 2^{2n} + \ldots. + 2^{\mathrm{N}n} = \left(\frac{2^n\,(2^{n\mathrm{N}} - 1)}{2^n - 1}\right)$$

members. Call this set $\mathcal{J}$ and index its members with natural numbers (starting from 1). In each set of k-tuples, the indexing follows the ordering defined earlier. The k-tuples themselves are ordered according to their lengths, starting with the 1-tuples and ending with the N-tuples.

3. The Specification Problem

(*a*) *Aspect.*—It is assumed that the sequential machine always starts from a distinguished initial internal state denoted by S_0. The required behaviour of the machine is now specified by the assignment of outputs from the set Φ to the elements i of $\mathcal{I}$. Thus, if ϕ_a is assigned to i, it means that starting from the initial internal state S_0, the input sequence i terminates with the machine in the output state ϕ_a.

In addition to the above assignments, certain end conditions will have to be prescribed. This is necessary because it was assumed that the sequential machine could identify k-sequences only for $k \leqslant N$. Hence, for completeness, its behaviour for k-sequences with $k > N$ should be specified. (As we shall see, in practice, it is sufficient to specify for $k = N + 1$ in addition to $k \leqslant N$.) This is done, most naturally, by giving explicit rules which identify k-sequences, for $k > N$, with the elements i of $\mathcal{I}$ in a unique manner.

Several cases will have to be distinguished according as the assignment of outputs is complete or partial.

Case 1.—An output is assigned to every element i of $\mathcal{I}$. This is the case of the 'exhaustive' specification and admits of a complete reduction to a minimal machine uniquely.

Case 2.—Outputs are assigned to some elements of $\mathcal{I}$ and it is specified that the assignment of outputs to the remaining elements of $\mathcal{I}$ are not of consequence (and, hence, can be done arbitrarily). The unassigned sequences are the so-called "Don't-care" sequences, and, clearly, any particular solution will depend upon the way outputs are assigned to the 'Don't-care' sequences. Each such completion of assignment will reduce case 2 to an instance of case 1.

Case 3.—Outputs are assigned to some elements of $\mathcal{I}$ and, in addition, auxiliary constraints are prescribed as holding between the elements of $\mathcal{I}$. These constraints may be of the following type:

(*a*) The elements i and k of $\mathcal{I}$ are *indistinguishable* or *equivalent*. By this is meant the following: The same output is assigned to i and k and, in addition, the subsequent behaviour of the machine is identical in the 2 cases. Under these circumstances, it would be natural to say that i and k refer to the same event.

(*b*) Certain input sequences are forbidden, *i.e.*, either they do not occur as inputs to the machine or, if they do, the machine stops, say.

(*c*) Other specialised constraints, if any.

Clearly cases 1, 2 and 3 are not independent. In particular, the exhaustive specification of case 1 contains implicitly all the information required to partition the elements of $\mathcal{I}$ into equivalence classes as described in case 3 (*a*). In fact, what we shall be concerned with in the rest of the paper is the determination of a *uniform* procedure to accomplish just this. Also it must be pointed out that where indistinguishability relations [as in 3 (*a*)] are known to exist, the end conditions cannot be presciibed arbitrarily, but must be obviously consistent (*see* § 7 below).

4. THE ASSIGNMENT OF INTERNAL STATES

The specification of the required machine behaviour having been given, the crux of the remaining problem in synthesis is the assignment of internal states with their transitions such that the resultant machine is minimal. Minimality implies that no machine with a smaller number of internal states will be able to satisfy *all* the specified requirements of behaviour. This problem is best handled in 2 distinct steps, as follows:

Step 1.—First, a uniform procedure is given for assigning internal states to input sequences, *i.e.*, to elements i of $\mathcal{I}$.

Step 2.—Next, a transition table is formed and a uniform procedure is given for finding the equivalence among the internal states. This results in a table with a minimum number of internal states satisfying the specification.

To begin with, we shall restrict our attention to case 1, *i.e.*, that of the 'exhaustive' specification, and develop the details of steps 1 and 2 for this case. Later, we shall show how these details can be suitably modified so as to reduce cases 2 and 3 to case 1.

5. CASE 1: 'EXHAUSTIVE' SPECIFICATION

Step 1: *Assignment of internal states.*—It is assumed here that an output from the set Φ has been specified to *each* element i of $\mathcal{I}$. Now, to each i a distinct internal state S_{i+1} is assigned, starting with S_1 for the 1-tuple $i = 0$. S_0 is the distinguished initial internal state. Thus, Table I, the assignment table, is formed as shown in Fig. 1.

Since the set has

$$\mathcal{I}(N) = \frac{2^n (2^{nN} - 1)}{2^n - 1}$$

members, Table I will have that many rows.

No.	Input sequences (elements i of $\mathcal{J}$)	Output states	Internal states
1	$i=0$	ϕ_{a_0}	S_1
2	$i=1$	ϕ_{a_1}	S_2
3	$i=2$	ϕ_{a_2}	S_3
.	...	...	...
.	...	...	...
$\mathcal{J}$(N)	$i=\mathcal{J}(N)-1$	$\phi_{a\mathcal{J}(N)-1}$	$S_{\mathcal{J}(N)}$

FIG. 1. Table I: The Assignment Table.

Step 2: *Construction of the transition table.*—Using Table I, we wish now to construct Table II, the transition table. The transition table, as shown in Fig. 2, is in the form of a matrix with 2^n columns and $\mathcal{J}(N)+1$ rows. The columns are named $I_0, I_1, \ldots, I_{2^n-1}$ and correspond to the input states with the same names. Similarly, the rows are named $S_0, S_1, \ldots, S_{\mathcal{J}}$ and correspond to the internal states with the same names. Let the cell located at the intersection of row S_i and column I_j be named (S_i, I_j). Each cell (S_i, I_j) contains two entries—one on the left, called the S-entry, which is the name of an internal state, and one on the right, called a ϕ-entry, which is the name of an output state.

The physical significance of the matrix is as follows: The names of the rows refer to the present internal states and those of the columns to the present input states. The S- and ϕ-entries in the cell (S_i, I_j) give the next internal state—present output state pair associated with the present internal state S_i and the present input state I_j.

Each row of Table I contains the names of one internal state and one output state. Each such pair S-ϕ is entered as an S-ϕ entry in one cell of Table II. To begin with, in the first row (named S_0), the S-ϕ pair corresponding to the 1-tuple j (in Table I) is entered in the cell (S_0, I_j). This is done for $j = 0, 1, \ldots, 2^n-1$. Any other cell (S_k, I_j) with $k > 1$ and $j = 0, 1, \ldots, 2^n-1$ is now filled in as follows: In Table I consider the input sequence $i(k)$ which has been assigned the internal state S_k. Form the extended sequence by adding I_j to the extreme right of $i(k)$. Let this input sequence be denoted by $i(k)I_j$. Now, fill in the cell (S_k, I_j), the S-ϕ

pair in Table I corresponding to the sequence $i(k)\,I_j$. In case $i(k)$ is itself an N-tuple, $i(k)\,I_j$ will be a $(N+1)$-tuple and, hence, there will be no row corresponding to it in Table I. However, by applying the proper end condition to $i(k)\,I_j$, it will be identified with one of the rows of Table I uniquely. The S-ϕ pair of that row is now used to fill in the cell (S_k, I_j).

I_t / S_t	I_0		I_1			I_{2^n-1}	
	S_{t+1}	ϕ_t	S_{t+1}	ϕ_t		S_{t+1}	ϕ_t
S_0					. . .		
S_1					. . .		
.	.	.	.	.	. . .	.	.
.	.	.	.	.	. . .	.	.
.	.	.	.	.	. . .	.	.
$S_{\mathcal{T}(N)}$					. . .		

FIG. 2. Table II: The Transition Table.

The form of Table II shows that it is a transition table. The proof that the transition table so obtained does in fact satisfy the specification given follows directly from the explicit rule of construction of the input sequences which correspond to the internal states which are row names of Table II.

Step 2 (*b*): *Construction of the reduction table.*—Referring to Table II, the internal states which are the names of rows, are assigned superscripts according to the following rules. For definiteness, we shall use the Greek letters (in their alphabetical order) for this purpose.

S_i and S_j have the same superscript if, and only if, the ϕ-entries in those two rows, along corresponding columns, are identical.

A uniform procedure for assigning the superscripts may be given. First partition the set of internal states (which are row names) according to their ϕ-entries in the 0th column and assign superscripts. Next, re-partition these subsets according to their ϕ-entries in the 1st column and assign fresh superscripts to the subsets newly formed and so on till the last column is reached. The internal states which belong to the same subset, after the completion of this operation, will have been assigned the same superscript.

In the modified Table II the row names are internal states with superscripts. From this modified Table II, the reduction table, Table III, is constructed, as shown in Fig. 3. The reduction table is in the form of a matrix, with the rows corresponding to the next internal states (written down in order, starting with S_1, together with their superscripts) and the columns corresponding to the input states written in order from left to right. In the cells are written down those internal states (with their superscripts) which associated with the input state which is their column name give rise to the next internal state which is their row name.

I_t / S_{t+1}	I_0	I_1	. . .	I_{2^n-1}
	S_t	S_t		S_t
S_1^{α}			. . .	
S_2^{γ}			. . .	
.	.	.	. . .	
.	.	.	. . .	
$S^{\rho}_{\mathcal{J}(N)}$			. . .	

FIG. 3. Table III: The Reduction Table.

This can be done with the help of the modified Table II as follows: write down in cell (S_i, I_j) of Table III all the row names of Table II whose S-entry in the column named I_j is S_i.

In the reduction process, we wish to merge all internal states which are equivalent to one another, into a single state. Thus when the reduction process is complete, we shall be left with a machine with a minimum number of internal states satisfying the specification.

Clearly, the states (S_{a_1}, S_{a_2}, ..., S_{a_n}) are equivalent if, and only if, they satisfy the 2 conditions:

(1) For every fixed input state I_j, the pair ($S_{a_i} - I_j$) results in the same output state for all a_i, $i = 1, 2, \ldots, n$.

(2) For every fixed input state I_j, the pairs ($S_{a_i} - I_j$) result in the next internal states S_{b_i} such that, either all the S_{b_i} are the same, or are equivalent.

From the way the superscripts have been assigned it is evident that, to begin with, in Table III, all internal states satisfying condition 1 will have the same superscripts. However, in general, such states with the same superscripts will not satisfy condition 2. The iteration procedure outlined below will enable us to modify the superscripts systematically, thus, completing the reduction process.

In Table III, let the columns be ordered from left to right and in each column, the cells from top to bottom. We shall say a row has superscript α if the rowname is a next internal state with superscript α.

The modification in the superscripts is done by the addition to them of indices i which are natural numbers. Thus, a re-partitioning of the set with superscript α_i is done by introducing the superscript α_{i+1}. Similarly for β, γ, etc. The superscripts are ordered according to the following scheme:

$$\alpha, \alpha_1, \alpha_2, \ldots\ldots\ldots, \beta, \beta_1, \beta_2, \ldots\ldots, \gamma, \gamma_1, \ldots.$$

As each new superscript is introduced, it is added at once to this list of superscripts at its proper place.

The following iteration scheme is now used:

(A) Find the first cell in column 0 containing an internal state with superscript α. Let its row have the superscript ω. Find the next cell in the same column with an internal state also with the superscript α but whose row has a superscript different from ω, say ω_1. Change all the superscripts α in that cell, and in all the succeeding cells, in the same column, whose rows have superscripts ω_1, to α_{i+1} (where α up to α_i are already in the list of superscripts). Make corresponding changes in the superscripts of these internal states wherever they occur in Table III (including the names of rows). If no such next cell is found, of course, no modification is to be done. Repeat, now, the process with column 1, then with column 2 and so on up to column $2^n - 1$.

Let A be called a minor cycle.

(B) The minor cycle is repeated using α_1 in place of α and, when this is done, with the next superscript in the list in place of the previous one and so on till the entire list of superscripts is exhausted. Let A and B together be called a major cycle.

(C) The major cycle is iterated till no modification of superscripts takes place in one complete major cycle.

The reduction process is now complete and the internal states having the same superscripts are equivalent and hence can be merged.

The proof that this iteration procedure terminates and does result in complete reduction of the equivalent states is straightforward and we shall not go into the details here. In fact, the procedure given above is essentially a systematization of Mealy's Rule I [pp. 1056 of (4)].

6. Case 2: Specification with 'Don't-Care' Sequences

Here, outputs are specified for some elements i of $\mathcal{J}$ and the remaining elements of $\mathcal{J}$ are specified as 'dont'-care' input sequences. Table I is now constructed, as before, assigning a distinct internal state S_{i+1} to each input sequence i. The ϕ-entries for the 'don't-care' sequences are written in as D.

Thus, when Table II is constructed from Table I, some of the cells will contain D as their ϕ-entries. Before forming the modified Table II, each entry marked D will have to be replaced by a particular ϕ_k. Superscripts are now assigned to the rownames and the reduction process completed as before. Clearly, every such replacement of the D's by the ϕ's will result in a reduced machine satisfying the given specification.

The problem of great interest, in practice, is how to assign the ϕ_k's to the D's so that the reduced machine has a minimum number of internal states. Clearly, not much can be said about this in detail, since the solution for any specific case will depend on the way the D's are distributed among the cells of Table II. However, in general, the aim should be to assign the ϕ_k's so as to minimise the number of distinct superscripts in the modified Table II.

7. Case 3: Specification with 'Input Equivalences'

In this case, outputs are specified to certain elements of $\mathcal{J}$ and, in addition, certain input sequences are specified as equivalent or indistinguishable (in the sense described in § 3), and certain other input sequences are forbidden. There may or may not be 'don't-care' sequences, besides.

It is to be pointed out here that if sequences i and k are equivalent, then for all j, $0 \leqslant j \leqslant 2^n - 1$, the extended sequences iI_j and kI_j will have to be equivalent.

In forming Table I, the outputs are written in wherever they are specified and they are marked D for the 'don't-care' sequences, if any. Distinct internal states are now assigned, in order, starting from S_1 for the input sequence $i = 0$. All the input sequences specified as equivalent are assigned the same internal state. The forbidden sequences are assigned the internal state F. Thus, if sequence i has been assigned the internal state S_k and if

j is the next sequence (in Table I) which is not equivalent to any l for $l < j$, then j is assigned the internal state S_{k+1}.

Let M be the total number of distinct internal states assigned in making up Table I. Table II is now formed with (M + 2) rows and 2^n columns. The columns refer to the input states, as usual, while the rows are named to correspond to the internal states, without any repetition. The first row is named S_0, the ith row, for $2 \leqslant i \leqslant M + 1$ is named S_{i-1} and the last row is named F.

The cells in the row S_0 are filled in as before. The cell (S_k, I_j), for $1 \leqslant k \leqslant M$ is filled in as follows: Let $i(k)$ be the *smallest* input sequence in Table I which has been assigned the internal state S_k. Form the input sequence $i(k)\, I_j$ by adding I_j on the extreme right of $i(k)$. Now, fill in the cell (S_k, I_j), the $S - \phi$ pair corresponding to the sequence $i(k)\, I_j$ in Table I. In case $i(k)\, I_j$ happens to be a $(N + 1)$-tuple, it has to be reduced to a l-tuple, for $l \leqslant N$, with the help of the given end conditions before applying the above rule.

All the cells in the last row named F have F as their S-entry and ϕ_F as their ϕ-entry, where ϕ_F is the output specified for any forbidden sequence. (Note that all forbidden sequences are equivalent.)

It is a straightforward process to verify that the transition table so constructed is complete and consistent, in the sense that every input sequence i occurring in Table I can be built out of Table II starting with S_0 and that it will terminate in the same internal state $S(i)$ as that specified for i in Table I.

Case 3 is now reduced to case 1 and the rest of the reduction process will go through as outlined in § 6 if there are ' don't-care ' sequences, or else as in § 5.

8. CONCLUDING REMARKS

In this paper we have been mainly concerned with the synthesis of sequential machines given their required behaviour in terms of events and associated outputs. The emphasis has been on obtaining a uniform procedure, the details of which can be programmed on a computer with ease. Although the formalism developed here is complete as far as it goes, it should be pointed out that several problems remain open. Here, we shall refer to only two which bear directly on the contents of this paper.

The first is concerned with the problem of specification. Although it follows from Kleene's results that everything that any finite automaton does can be completely described in terms of events and outputs, this does not, in practice, seem to be of much help. Presumably, there are behaviours which are intuitively meaningful but which one does not know how to express in terms of events and outputs. It should be useful to have a uniform procedure for doing this, if not in every case, at least, in a suitably well-defined large class of cases.

The second problem has to do with the design of complex sequential machines. In practice, one tries to visualise a large machine as made up of submachines, each functionally independent to a large extent, and thus reduce the given problem to one of synthesising a large number of smaller machines. In such cases, there may be feedback loops from the output of one submachine to the input of another and problems analogous to those of ' stability ' might arise. It might be useful to consider modifications or suitable extensions of the formalism given here to include these cases.

That both these problems are of great practical importance is shown by the fact that they play a central role in the specification and design of large-scale digital computers. It is of interest to note that digital computers, although seemingly well understood from the point of view of their description as sequential machines, still fall outside the scope of all existing *routine* synthesis procedures, for the most part.

9. Illustrative Examples

In this final section, we give two examples illustrating the concepts and the methods introduced in the paper.

Example 1.—Synthesis of a sequential machine with one input flip-flop and two output flip-flops and whose output is a function of utmost the three previous inputs. The required behaviour is specified exhaustively by means of the assignment table in Fig. 4. (The 2 input states are denoted as 0 and 1 and the 4 output states by 00, 01, 10 and 11 respectively.)

End condition.—If $I_1I_2 \ldots . I_{k-2}I_{k-1}I_k$ is any sequence of length k, for $k \geqslant 3$, then, it is equivalent to the sequence $I_{k-2}I_{k-1}I_k$.

From Fig. 4, one obtains the modified transition table as shown in Fig. 5. The reduction table, at the end of complete reduction, is shown in Fig. 6. Thus, the equivalent states are:

$\alpha: (S_0)$;

$\alpha_1: (S_1)$;

$\alpha_2: (S_4, S_6, S_8, S_{10}, S_{12}, S_{14})$;

$\beta: (S_2)$;

$\beta_1: (S_3, S_7, S_{11})$;

$\gamma: (S_5, S_9, S_{13})$.

No.	$\mathcal{T}$	ϕ	S
1	0	00	S_1
2	1	01	S_2
3	00	00	S_3
4	01	01	S_4
5	10	01	S_5
6	11	10	S_6
7	000	01	S_7
8	001	11	S_8
9	010	00	S_9
10	011	01	S_{10}
11	100	10	S_{11}
12	101	10	S_{12}
13	110	00	S_{13}
14	111	01	S_{14}

FIG. 4. The Assignment Table.

S_t \ I_t	0		1	
	S_{t+1}	ϕ_t	S_{t+1}	ϕ_t
S_0^{α}	S_1	00	S_2	01
S_1^{α}	S_3	00	S_4	01
S_2^{β}	S_5	01	S_6	10
$S_3^{\beta_1}$	S_7	01	S_8	11
S_4^{α}	S_9	00	S_{10}	01
S_5^{γ}	S_{11}	10	S_{12}	10
S_6^{α}	S_{13}	00	S_{14}	01
$S_7^{\beta_1}$	S_7	01	S_8	11
S_8^{α}	S_9	00	S_{10}	01
S_9^{γ}	S_{11}	10	S_{12}	10
S_{10}^{α}	S_{13}	00	S_{14}	01
$S_{11}^{\beta_1}$	S_7	01	S_8	11
S_{12}^{α}	S_9	00	S_{10}	01
S_{13}^{γ}	S_{11}	10	S_{12}	10
S_{14}^{α}	S_{13}	00	S_{14}	01

FIG. 5. The Modified Transition Table.

In Fig. 7 is given the state diagram of the completely reduced machine. It is easily seen that this diagram is not further reducible.

Example 2.—We give here, in terms of our notation, the specification for the ‘reversible counter’ considered by Huffman [pp. 276 of (2)].

S_{t+1} \ I_t	0	1
	S_t	S_t
$S_1^{\alpha_1}$	S_0^{α}	
S_2^{β}		S_0^{α}
$S_3^{\beta_1}$	$S_1^{\alpha_1}$	
$S_4^{\alpha_2}$		$S_1^{\alpha_1}$
S_5^{γ}	S_2^{β}	
$S_6^{\alpha_2}$		S_2^{β}
$S_7^{\beta_1}$	$S_3^{\beta_1}$, $S_7^{\beta_1}$, $S_{11}^{\beta_1}$	
$S_8^{\alpha_2}$		$S_3^{\beta_1}$, $S_7^{\beta_1}$, $S_{11}^{\beta_1}$
S_9^{γ}	$S_4^{\alpha_2}$, $S_8^{\alpha_2}$, $S_{12}^{\alpha_2}$	
$S_{10}^{\alpha_2}$		$S_4^{\alpha_2}$, $S_8^{\alpha_2}$, $S_{12}^{\alpha_2}$
$S_{11}^{\beta_1}$	S_5^{γ}, S_9^{γ}, S_{13}^{γ}	
$S_{12}^{\alpha_2}$		S_5^{γ}, S_9^{γ}, S_{13}^{γ}
S_{13}^{γ}	$S_6^{\alpha_2}$, $S_{10}^{\alpha_2}$, $S_{14}^{\alpha_2}$	
$S_{14}^{\alpha_2}$		$S_6^{\alpha_2}$, $S_{10}^{\alpha_2}$, $S_{14}^{\alpha_2}$

FIG. 6. The Reduced Table.

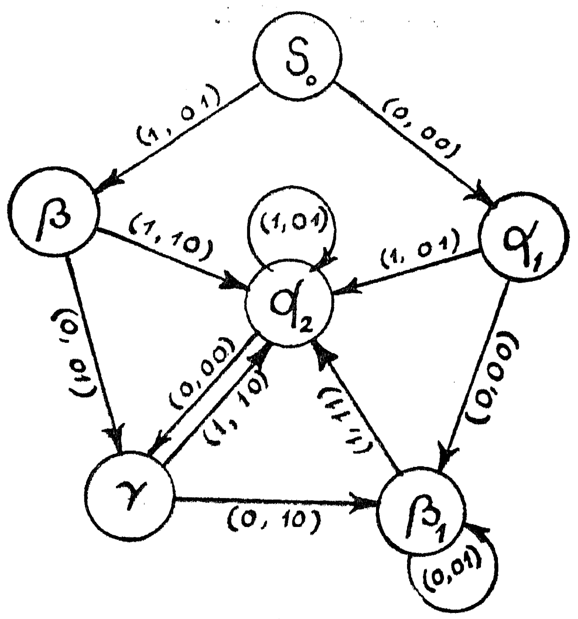

FIG. 7. The State Diagram of the Minimal Machine.

4 input states: I_0, I_1, I_2, I_3

4 ouput states: $\phi_0, \phi_1, \phi_2, \phi_3$

Restraints on input sequences:

(1) For any I_j, $I_0I_j = I_jI_0 = I_j$.

(2) I_3 is forbidden.

Output assignments:

(1) I_0 has the output ϕ_0.

(2) If i is any sequence with output ϕ_{a_i}.

Then iI_1 has output $\phi_{a_{i+1}}$ (Mod. 4).

iI_2 has output $\phi_{a_{i+3}}$ (Mod. 4).

iI_0 has output ϕ_{a_i}

It is easily verified that, with the above, the assignment table can be completely filled in and the reduction process carried through.

REFERENCES

1. Kleene, S. C. .. *Automata Studies*, Princeton Univ. Press, 1956, pp. 3-41.
2. Huffman, D. A. .. *Jour. Frank. Inst.*, 1954, **257**, 161-90, 175-303.
3. Moore, E. F. .. *Automata Studies*, Princeton Univ. Press, 1956, pp. 129-53.
4. Mealy, G. H. .. *Bell Syst. Tech. Jour.*, 1955, **34**, 1045-79.
5. Burks, A. W. and Wang, Hao. *Journal of Association for Computing Machinery*, 1957, pp. 193-218, 279-97.

701-59 Printed at the Bangalore Press, Bangalore City, by C. Vasudeva Rao, Superintendent and Published by The Indian Academy of Sciences, Bangalore