Minimum-State Sequential Circuits for a Restricted Class of Incompletely Specified Flow Tables*

By E. J. Mc CLUSKEY, JR.†

(Manuscript received June 22, 1962)

This paper is concerned with the problem of obtaining minimum-state sequential circuits for incompletely specified flow tables. Attention is directed to relay-type flow tables in which the only unspecified entries are those which occur because of restrictions on the allowed input-variable changes. For this type of flow table it is shown that a simplified version of the Unger-Paull procedure is sufficient. In particular, only maximum compatibles need be considered in forming the minimum-state sequential circuit.

I. INTRODUCTION

One of the classical problems of sequential circuit theory is that of obtaining a minimum-state sequential circuit satisfying the requirements of a given flow table. When the flow table is incompletely specified, the procedures for obtaining the minimum-state sequential circuit are lengthy and require such extensive enumeration that they are impractical for computer implementation. This paper discusses a restricted type of incompletely specified flow table for which more efficient procedures can be devised. In particular, relay-type flow tables in which the unspecified entries all are present because of a restriction of the manner in which the inputs can change are considered. It is shown that for this type of flow table only the maximal compatibles or compatibility classes need be considered in forming a minimum-state circuit.

II. BACKGROUND

The problem of finding a minimum-state sequential circuit for an incompletely specified flow table has been discussed extensively in

* This paper was presented at the International Symposium on Theory of Switching Systems and Finite Automata, Moscow, U.S.S.R., Sept. 24–Oct. 2, 1962.

† Department of Electrical Engineering, Digital Systems Laboratory, Princeton University. Work supported in part by Bell Telephone Laboratories.

previous papers. The results presented in these papers, particularly that of Paull and Unger,[1] are necessary for the results to be presented here. A brief summary of previous results assumed in this paper will be presented first.

The usual approach to the study of minimum-state sequential circuits involves consideration of which flow tables specify the same external behavior as a given flow table Q. Any flow table which does specify the same external behavior as Q is said to *cover* Q. The usual objective is to formulate a procedure for finding, for any flow table Q, a minimum-state flow table which covers Q. A formal definition of the covering relation among flow tables is:

Definition. A flow table P is said to cover a flow table Q (written $P \supset Q$) if and only if, for each internal state q_i of Q there is an internal state p_j of P such that for any input sequence applied to both tables initially in states q_i and p_j respectively, the output sequences are identical *whenever the output of Q is specified.*

The definition is suitable for flow tables in which each next-state entry is specified but some of the output entries may be unspecified. There is no loss of generality in considering this class of circuits since it has been shown by Narasimhan[2] that all flow tables can be placed in this form. This definition of a flow table covering another flow table induces a corresponding relation between the internal states of the two tables.

Definition. An internal state p_i of a flow table P is said to cover an internal state q_j of a flow table Q (written $p_i \supset q_j$) if and only if, for any input sequence applied to P and Q initially in states p_i and q_j, respectively, the outputs are identical whenever the output of Q is specified.

If flow table P covers flow table Q and P has fewer states than Q, then one state of P must cover more than one state of Q. Whenever two states of a flow table can be covered by a single state of another flow table, the two states must have the following relation:

Definition. Two internal states, q_i and q_j of Q, are *compatible* if and only if, for all input sequences, the output sequence which results when Q is initially in q_i is the same as the output sequence which results when Q is initially in q_j *whenever both outputs are specified.*

Theorem 1. If internal state p_i of P covers both internal states q_j and q_k of Q, then states q_j and q_k must be compatible.

Lemma. If internal state p_i of P covers internal states $q_{j_1}, q_{j_2}, \cdots q_{j_k}$ of Q then states $q_{j_1}, q_{j_2}, \cdots q_{j_k}$ must form a compatibility class; *that is, each pair of the q_{j_1} must be compatible.*

It follows from this that if $P \supset Q$, then each state p_i of P must cover a compatibility class of the states of Q. In addition, the compatibility classes covered by states of P must have the closure property, to be described next.

Definition. The input states of a sequential circuit will be represented by the symbols $\mathbf{x}^0$, $\mathbf{x}^1$, $\cdots$ $\mathbf{x}^r$. The internal states of a sequential circuit will be represented by the symbols s_1, s_2, $\cdots$ s_r.

Definition. The next-state entry specified by a flow table for input state $\mathbf{x}^\alpha$ and internal state s_i will be represented by the symbol $S(\mathbf{x}^\alpha,s_1)$.

Definition. A collection of compatibility classes is said to be *closed* if and only if for each compatibility class $\{s_1, s_2, \cdots s_m\}$, all of the states $S(\mathbf{x}^\alpha,s_1)$, $S(\mathbf{x}^\alpha,s_2)$ $\cdots$ $S(\mathbf{x}^\alpha,s_m)$ are included in a single compatibility class in the collection. This must be true for all choices of α.

Theorem 2. A flow table P covers a flow table Q if and only if:

(A) *each internal state of Q is included in at least one compatibility class of Q that is covered by an internal state of P, and*

(B) *the compatibility classes of Q which are covered by internal states of P form a closed collection.*

There is a procedure whereby for each closed collection of compatibility classes of a flow table Q (with every internal state of Q included in at least one compatibility class) it is possible to obtain a flow table P which covers Q and which contains the same number of internal states as there are compatibility classes in the collection. Thus, a minimum-state flow table which covers a given flow table Q can be formed from a closed collection of compatibility classes of Q containing a minimum number of such classes.

Satisfactory techniques for determining the compatibility classes for a given flow table are known.[1] Actually the maximal compatibility classes can be determined, and all other compatibility classes must be subclasses of these. Presently known techniques for obtaining minimum-state flow tables are inadequate because of the necessity for considering the inclusion of nonmaximal compatibility classes in the closed collection used in forming the covering flow table P.[1] Each subclass of the maximal compatibility classes must be considered, and this number of subclasses can be prohibitively large. The necessity for considering nonmaximal compatibility classes results directly from the closure requirement. The object of this paper is to show that for a certain type of incompletely specified flow table it is always possible to use the maximal compatibility classes in forming a minimum-state flow table. For this type of flow table, the procedure for obtaining a minimum-state flow is very much simpler than in the general case. Moreover, the type of

flow table for which this result holds is the type most often encountered in actual design problems.

III. TYPE A FLOW TABLES

The following discussion applies specifically to flow tables for fundamental mode operation.[3] For the purposes of this paper, a circuit will be said to be operating in fundamental mode if no input is changed until after the circuit has "settled down," that is, until after all internal signal changes have stopped. This type of circuit operation is often referred to as "relay type" or "asynchronous."[4]

It is customary to begin the design of a fundamental mode sequential circuit by writing down a primitive flow table — a flow table in which there is exactly one stable state in each row. For such a table it is possible to associate one of the input states (columns of the flow table) with each internal state, since each internal state is stable for exactly one input state.

Definition. Let P be a primitive, fundamental-mode flow table. Let $s_1^{\alpha}, s_2^{\alpha}, \cdots s_{\sigma}^{\alpha}$ be the internal states of P which are stable for the input state $\mathbf{x}^{\alpha}$; $s_1^{\beta}, s_2^{\beta}, \cdots s_{\tau}^{\beta}$ be the internal states of P which are stable for input state $\mathbf{x}^{\beta}$, etc.

It will be assumed that in a flow table each unstable next-state entry is followed directly by a stable next-state entry — no multiple changes of internal state are allowed. Whether a flow table is of the type considered here, to be called Type A, depends on the mechanism whereby unspecified entries occur in the table. Specifically, a flow table is of Type A if the only unspecified entries are those which arise because of a restriction on which input states can directly follow each given input state.

Definition. A flow table is of Type A if and only if: (a) it is a flow table for fundamental mode operation; (b) it is a primitive flow table; (c) each unstable next-state entry refers to an internal state which is stable for the corresponding input state; and (d) the only unspecified entries are those which occur because of a restriction on the input states which can directly follow each possible input state.

For fundamental-mode flow tables it is common practice to assume that only single changes of input variables are possible. Thus, the input state for which $x_1 = 0$, $x_2 = 0$, cannot be followed by the input state with $x_1 = 1$, $x_2 = 1$. If this restriction is the only source of unspecified entries in the table, then the table is of Type A.

Part (d) of the above definition of Type A flow tables can be restated directly in terms of the pattern of unspecified entries in the table (rather than the mechanism by which they arise). In order to describe

this, it is convenient to assume that the rows in the table are partitioned so that all of the rows which are in the same partition are stable for the same input state and there is one partition for each input state. Actually, if the outputs associated with the stable states are all specified, each partition need only include rows which are all stable for the same input state *and* have the same outputs associated with the stable next-state entry. Part (d) of the definition of Type A flow tables can be considered satisfied if, whenever any row has an unspecified entry for an input state $\mathbf{x}^\alpha$, all other rows in the same partition also have unspecified entries for input state $\mathbf{x}^\alpha$. This condition is actually somewhat more general than the condition (d) given originally, but the theorems are all valid for this more general condition.

For Type A flow tables, the compatibility relation has certain properties which are not generally satisfied for arbitrary flow tables. It is these special properties which form the basis for the simplified procedure to be derived here.

Theorem 3. Let s_i^α, s_j^α, s_k^α, *be three internal states of a Type A flow table P which are all stable for input state* $\mathbf{x}^\alpha$. *If* s_i^α *and* s_j^α *are compatible, and* s_i^α *and* s_k^α *are compatible, then* s_j^α *and* s_k^α *are compatible.*

Proof. By the definition of compatibility, when any input sequence is applied to P the output sequence with P initially in s_j^α will be the same as the output sequence with P initially in s_i^α whenever both outputs are specified. However, because P is a Type A flow table, whenever the output is specified for P initially in s_i^α, the output for P initially in s_j^α will be specified and vice versa. Similar remarks apply to states s_i^α and s_k^α. Thus the output for P initially in s_j^α must always agree with the output for P initially in s_i^α, and the output for P initially in s_k^α must always agree with the output for P initially in s_i^α. Whenever any one of these outputs is specified, all three must be specified; therefore the outputs for P initially in s_j^α and P initially in s_k^α must always agree. This shows that states s_i^α and s_k^α must be compatible. See also Ref. 4, pp. 183–185.

Let the fact that two states p and q are compatible be written symbolically as $p \bigcirc q$. Then for states satisfying the conditions of Theorem 3, the following properties must hold:

(*P*1) $s_i^\alpha \bigcirc s_i^\alpha$ (reflexive)

(*P*2) If $s_i^\alpha \bigcirc s_j^\alpha$ then $s_j^\alpha \bigcirc s_i^\alpha$ (symmetric)

(*P*3) If $s_i^\alpha \bigcirc s_j^\alpha$ and $s_i^\alpha \bigcirc s_k^\alpha$, then $s_j^\alpha \bigcirc s_k^\alpha$ (transitive).

A binary relation which satisfies these three properties is an equivalence relation.[5] The important characteristic of an equivalence relation is that it divides the set of objects on which it is defined into disjoint (nonoverlapping) equivalence classes.

Theorem 4. Let P be a Type A flow table. Let s_i^{α} and s_j^{α} be two internal states of P which are both stable for input state $\mathbf{x}^{\alpha}$, and let s_k^{β} be an internal state of P which is stable for input state $\mathbf{x}^{\beta}$. If s_i^{α} and s_j^{α} are compatible, and s_i^{α} and s_k^{β} are compatible then s_j^{α} and s_k^{β} are compatible.

Proof. For any input sequence, the outputs for P initially in s_i^{α} and for P initially in s_k^{β} must be identical whenever both are specified. However, the output for P initially in s_j^{α} is specified whenever the output for P initially in s_i^{α} is specified, and these outputs must always be the same. Thus, all specified outputs for P initially in s_j^{α} are the same as the corresponding outputs for P initially in s_i^{α}, and the s_i^{α} outputs are the same as the outputs for P initially in s_k^{β} whenever both outputs are specified. It follows from this that the outputs for P initially in s_j^{α} and for P initially in s_k^{β} must be the same when both are specified and hence that s_j^{α} and s_k^{β} are compatible.

Definition. A set of internal states of a flow table P is a maximum compatibility class if and only if (i) every pair of states which are both in the set are compatible, and (ii) there is no other state of P not in the set which is compatible with all of the states in the set.

Theorem 5. Let P be a Type A flow table. Let s_i^{α} and s_j^{α} be two internal states of P which are both stable for input state $\mathbf{x}^{\alpha}$ and which are compatible. Then any maximum compatibility set which includes s_i^{α} must also include s_j^{α} and vice versa.

Proof. Suppose that C is a maximum compatibility class which includes s_i^{α}. If there is any other state in C which is stable for input state $\mathbf{x}^{\alpha}$, say s_k^{α}, then s_i^{α} and s_k^{α} are compatible and s_i^{α} and s_j^{α} are compatible. By Theorem 3, states s_k^{α} and s_j^{α} must then be compatible. Thus s_j^{α} is compatible with all states in C which are stable for input $\mathbf{x}^{\alpha}$. Suppose that there is some state s_h^{β} in C which is stable for some input state β different from α. Then states s_i^{α} and s_h^{β} are compatible and states s_i^{α} and s_j^{α} are compatible. By Theorem 4, states s_h^{β} and s_j^{α} must then be compatible. Thus state s_j^{α} is compatible with all states in C and therefore must be included in C.

Theorem 6. Let P be a Type A flow table. Then any collection of maximum compatibility classes of P for which each internal state of P is included in at least one of the maximum compatibility classes is closed.

Proof. Let $\{s_1, s_2, \cdots s_m\}$ be one of the maximum compatibility classes. Then if the collection of maximum compatibility classes is closed, all of the states $S(\mathbf{x}^{\alpha},s_1)$, $S(\mathbf{x}^{\alpha},s_2) \cdots S(\mathbf{x}^{\alpha},s_m)$ must be included in one of the maximum compatibility classes of the collection. Since P is a Type A flow table, all of the states $S(\mathbf{x}^{\alpha},s_1)$, $S(\mathbf{x}^{\alpha},s_2)$, $\cdots S(\mathbf{x}^{\alpha},s_m)$ must be stable for input state $\mathbf{x}^{\alpha}$. It has been shown that all pairs of these states must be compatible since $\{s_1, s_2, \cdots s_m\}$ is a compatibility

class.[1] By Theorem 5 any maximum compatibility class which includes the internal state $S(\mathbf{x}^{\alpha},s_1)$ must also include $S(\mathbf{x}^{\alpha},s_2)$ $\cdots$ $S(\mathbf{x}^{\alpha},s_m)$. The conditions of Theorem 6 assume that there is at least one maximum compatibility class in the collection which includes state $S(\mathbf{x}^{\alpha},s_1)$. Therefore there must be at least one class in the collection which includes all of the states $S(\mathbf{x}^{\alpha},s_1)$, $S(\mathbf{x}^{\alpha},s_2)$, $\cdots$ $S(\mathbf{x}^{\alpha},s_m)$. From this it follows that the collection is closed.

Theorem 7. Let P be a type A flow table. Then there is at least one minimum-state flow table Q which (a) covers P, (b) contains the minimum number of internal states for any flow table covering P, and (c) for which each internal state of Q covers a maximum *compatibility class of P.*

Proof. There is at least one flow table — P itself — which covers P, and there must be at least one such table containing a minimum number of states. Suppose that R is a flow table containing a minimum number of states and covering P. If each state of R covers a maximum compatibility class of P, the theorem is satisfied. Therefore suppose that each state r_i of R covers a compatibility class C_i of P and that at least one of these compatibility classes is not maximal. Now form a new collection of compatibility classes by replacing each C_i by one of the *maximal* compatibility classes in which it is included. The maximal compatibility class which replaces C_i will be denoted as M_i. The collection of the M_i will (a) contain the same number of classes as the collection of the C_i, (b) include each state of P in at least one M_i, and (c) be closed because of Theorem 6. It is thus possible to form from the M_i a new flow table Q which satisfies all of the conditions of the theorem.

IV. EXAMPLE

In order to illustrate the significance of the theorems, an example of a Type A flow table will be discussed. Table I shows a Type A flow table and the corresponding maximal compatibility classes. States 5 and 10 are the only pair of compatible states which are both stable for the same input state. By Theorem 5, any maximal compatibility class which includes either of these two states (5 or 10) must include both of them. Inspection of Table I(c) shows this to be true. It follows from Theorem 6 that any closure requirements must involve only these two states, and Table I(b) shows this to be true. The formation of a minimum-row flow table which covers Table I(b) requires only that a sufficient number of maximum compatibility classes be chosen so that each internal state of Table I(a) is included in at least one maximal compatibility class. This problem is formally identical to the problem of choosing which prime implicants should be included in a minimal sum

TABLE I — A TYPE A FLOW TABLE

(a) Flow Table

S	x^0	x^1	x^2	x^3	x^4
1	(1), 0	2 , 0	—	5 , 0	—
2	4 , 1	(2), 0	3 , 0	—	—
3	—	2 , 0	(3), 0	5 , 0	6 , 0
4	(4), 1	8 , 1	—	10 , 0	—
5	—	—	3 , 0	(5), 0	7 , 1
6	1 , 0	—	3 , 0	5 , 0	(6), 0
7	4 , 1	—	9 , 0	10 , 0	(7), 1
8	4 , 1	(8), 1	9 , 0	—	—
9	—	8 , 1	(9), 0	10 , 0	6 , 0
10	—	—	3 , 0	(10), 0	7 , 1

S, Z

(b) Implication Table for Determining Compatibity

	1	2	3	4	5	6	7	8	9
2	x								
3	√	√							
4	x	x	x						
5	√	√	x	√					
6	√	x	√	x	x				
7	x	x	x	√	x	x			
8	x	x	x	√	x	x	√		
9	x	x	x	√	x	x	x	√	
10	5, 10	√	x	√	√	x	x	x	x

(c) Maximal Compatibility Classes

A: 4, 8, 9	E: 2, 3
B: 4, 7, 8	F: 2, 5, 10
C: 4, 5, 10	G: 1, 5, 10
D: 1, 3, 6	

for a Boolean function.[6] Therefore, the same techniques can be used. Table II shows a "prime implicant table" for the maximal compatibility classes of Table I. Each row of Table II corresponds to one of the maximal compatibility classes. Each column of Table II represents one of the internal states of Table I. An X is placed in a cell of Table II if the maximal compatibility class corresponding to the row includes

TABLE II—PRIME IMPLICANT TABLE FOR THE MAXIMAL COMPATIBILITY CLASSES OF TABLE I

Maximal Compatibility Classes	Internal States 1	2	3	4	5	6	7	8	9	10
*A				X				X	(X)	
*B				X			(X)	X		
C				X	X					X
*D	X		X			(X)				
E		X	X							
F		X			X					X
G	X				X					X

the internal state corresponding to the column. A sufficient number of rows must be chosen so that each column has an X in at least one of the chosen rows. It follows from this that rows A, B, and D must be chosen, since columns 9, 7, and 6 each contain only a single X. After A, B, and D have been chosen, only columns 2 and 5 do not contain an X in any of the chosen rows. This may be remedied by also choosing row F. Thus the collection of maximal compatibility classes A, B, D and F corresponds to a minimum-row flow table which covers Table I(a). Such a table is shown in Table III.

Inspection of Table II shows that columns 5 and 10 are identical. Any states which are compatible and are stable for the same input state will always have identical columns in the ' prime implicant table" for maximal compatibility classes. It is therefore unnecessary to carry these states along explicitly. Each set of such states can immediately be replaced by a single state (this corresponds to Huffman's merging).[4] The sets of states which are "merged" in this step are exactly the sets of states which must be covered by single states of the new table in order

TABLE III—A MINIMUM ROW FLOW TABLE WHICH COVERS TABLE I(a)

	S	x^0	x^1	x^2	x^3	x^4
(4, 8, 9)	A	(A), 1	(A), 1	(A), 0	F, 0	D, 0
(4, 7, 8)	B	(B), 1	(B), 1	A, 0	F, 0	(B), 1
(1, 3, 6)	D	(D), 0	F, 0	(D), 0	F, 0	(D), 0
(2, 5, 10)	F	B, 1	(F), 0	D, 0	(F), 0	B, 1

S, Z

to insure that closure is satisfied. Thus, closure will always be satisfied as long as these sets of states are identified; i.e., either all members of the set are included in a compatibility class or all members are excluded.

After the collection of maximal compatibility classes which correspond to a minimum-row flow table has been determined, states can sometimes be removed from some of the classes. The advantage of removing states and thereby obtaining nonmaximal compatibility classes is the corresponding introduction of unspecified entries in the minimum-row flow table. Closure will still be satisfied as long as (*i*) only sets of states which were identified previously, or single states which cannot be identified with any other state, are removed; and (*ii*) each state is still contained in one of the remaining compatibility classes. This procedure can be carried out until each state is included in only one of the compatibility classes. In Table III, this could mean the removal of states 4 and 8 from class B.

CONCLUSIONS

It has been shown that for incompletely specified flow tables which satisfy certain very common conditions, greatly simplified procedures for obtaining minimum-state flow tables exist. For this class of tables it should now be possible to develop computer programs which are guaranteed to work for tables with sufficiently large numbers of internal states so that hand techniques are not feasible.

ACKNOWLEDGMENTS

The author would like to acknowledge his indebtedness to E. Eichelberger and S. H. Unger for their comments and to T. H. Crowley for his advice and help in preparing this paper for publication.

REFERENCES

1. Paull, M. C., and Unger, S. H., Minimizing the Number of States in Incompletely Specified Sequential Switching Functions, I.R.E. Trans. on Electronic Computers, **EC-8,** No. 3, September, 1959, pp. 356–367.
2. Narasimhan, R., Minimizing Incompletely Specified Sequential Switching Functions, I.R.E. Trans. on Electronic Computers, **EC-10,** No. 3, September, 1961, pp. 531–532.
3. McCluskey, E. J., Jr., Fundamental Mode and Pulse Mode Sequential Circuits, International Federation for Information Processing Congress, Munich, Germany, Aug. 27 to Sept. 1, 1962. Also Technical Report No. 29, Digital Systems Laboratory, Electrical Engineering Department, Princeton University.
4. Huffman, D. A., The Synthesis of Sequential Switching Circuits, Journal of the Franklin Institute, **257,** No. 3, March, 1954, pp. 161–190; No. 4, April, 1954, pp. 275–303.
5. Birkhoff, G., and MacLane, S., *A Survey of Modern Algebra*, The Macmillan Company, New York, N.Y., 1941.
6. McCluskey, E. J., Jr., Minimization of Boolean Functions, B.S.T.J., **35,** November, 1956, pp. 1417–1444.