SCIENCE BULLETIN
CHAO-TUNG UNIV.
Vol.3,No.2(1969)

On the Minimization of the Number of States in Incompletely Specified Sequential Machines

C. C. Yang* and H. S. Fung*

Abstract-The algorithm of a systematic minimization method given in this paper uses tree idea for generating all compatibility classes and uses Meisel's tree display to produce all possible minimal solutions of a given incompletely specified sequential machine. An example is given for illustration.

I. Introduction

An important problem in the area of sequential machine is the reduction of the number of states. When a machine is completely specified, this problem has been successfully solved and the minimal solution is unique. When the machine is incompletely specified, however, this problem becomes complicated and the number of its minimal machines may be more than one. Hence,the state reductions of incompletely specified machines have manifested considerable interest and attention in nearly past decade [1-8].

Ginsberg [1] has mentioned that it is not always possible to obtain a minimal machine if the unspecified next states and/or outputs are assigned to fixed specifications. Thus we would not treat any unspecified next state or output as don't-care for state reduction.

Paull and Unger [2] have developed a general theory for incompletely specified machines. Although their method for generating the set of maximal compatibility classes by means of implication table is quite successful, their technique for selecting a minimal closed set of maximal compatibility classes which covers a given flow table involves a partially enumeration and requires the examination of a large number of cases for closure and co-

*The authors are with the College of Engineering, National Chiao Tung University, Hsinchu, Taiwan, Republic of China

vering.

McClusky [4] has presented a simplified state - reduction method. However, his technique is valid for a restricted class of machines.

Beatty and Miller [3,5] have described a programmed algorithm for finding all minimal solutions. But their technique seems tedious.

Grasselli and Luccio [6] have introduced the concept of prime compatibility classes derived from the set of maximal compatibility classes. Their state-reduction technique is the selection of a minimal closed set of prime compatibility classes which covers the given flow table. The selection may be obtained as the solution of an integer linear program or by tabular techniques that are an extension of those used in the selection of prime implicants. The tabular method, however, is complex and may yield a cyclic prime implicant table which requires further techniques for obtaining a solution.

Meisel [7] uses the prime compatibility classes as nodes to construct directed trees from which the shortest closed covering paths correspond to minimal solutions.

Bouchet [8] has developed the idea of distributive lattices and an algorithm for deriving minimal closed coverings from lattices. His method is, however, very complicated.

This paper tries to find all possible minimal solutions with a systematic algorithm in which the set of all possible compatibility classes and their class sets is considered and Meisel's tree display is used. An example is given for illustration.

2. Preliminary Considerations

Before introducing our systematic method for reducing the number of states of an incompletely specified sequential machine, some relevant definitions and theorems concerning sequential machines, compatibility classes, implied compatibility classes and class sets are described as follows:

Definition 1 A sequential machine S is a 5-tuple

$$S = (Q, X, Y, M, N)$$

satisfying the following conditions:

1) Q, X, and Y are finite nonempty sets of states, inputs and outputs respectively,

2) The next state function M maps from a subset D_M of $Q \times X$ into Q,

3) The output function N maps from a subset D_N of $Q \times X$ onto

When $D_M = D_N = Q \times X$, S is called a complete machine. Otherwise, S is incompletely specified. In this paper, every unspecified next state or output in a flow table is denoted by a dash "-".

Definition 2 Two sequences of specified and unspecified outputs are called incomparable if a pair of corresponding outputs of the sequences is discriminated whenever both these outputs are specified.

Definition 3 Two states q_j and q_k in Q are called incompatible, written $q_j \not\sim q_k$, iff incomparable output sequences are yielded for some sequences of inputs applied with a machine S initially in states q_j and q_k. A pair of incompatible states q_j and q_k is called an incompatibility-pair, still denoted by $q_j \not\sim q_k$. Otherwise, q_j and q_k are called compatible, or form a compatible-pair, written $q_j \sim q_k$, iff they are not incompatible.

By this definition, a state is compatible with itself, called self-compatible.

Theorem 1 Two states q_j and q_k are incompatible if the outputs $N(q_j, x)$ and $N(q_k, x)$ are incomparable for some input x in X whenever both these outputs are defined by the output function N.

Theorem 2 Two states q_j and q_k are incompatible if the next states $M(q_j,x)$ and $M(q_k, x)$ are incompatible for some x in X whenever both these next states are defined by the next state function M.

If $M(q_j, x)$ and $M(q_k, x)$ are incompatible for some x in X, then so are q_j and q_k by Theorem 2. We say that the incompatibility of q_j and q_k is implied by the incompatibility of $M(q_j, x)$ and $M(q_k, x)$.

Definition 4 A tree is a branching structure which consists of a set of branches and a set of nodes. The nodes constitute the endpoints of the branches. The starting node is called the root of the tree and each terminating node is called a leaf.

Definition 5 A compatibility tree is a two-branching structure growing from top to bottom and satisfying the following conditions:

1) The root is labeled by the set of states

$$Q = \{q_1, q_2, \cdots\cdots, q_n\}.$$

The set is used as the root name and is simply represented by $\{1, 2, \ldots, n\}$.

2) Two new nodes are grown from a node which contains incompatible states. For instance, if the node $T_i = \{j_1, j_2, \ldots, j_r, j_{r+1}, \ldots, j_s\}$ for $s \leq n$ contains incompatible states q_{j_r} and $q_{j_{r+1}}$, then the grown nodes are respectively $T_{i_1} =$

$\{j_1, j_2, \cdots, j_{r-1}, j_{r+1}, \cdots, j_s\}$ and $T_{i_2} = \{j_1, j_2, \cdots, j_r, j_{r+2}, \cdots, j_s\}$.

3) Step 2) continues until a node containing no incompatible states. This node is called a leaf.

4) If nodes are labeled by identical sets, then only one node is retained and all others are crossed for eliminating redundancy.

5) If a leaf contains a single state, it is crossed.

Definition 6 A compatibility class C_j is a set of states

$$C_j = \{j_1, j_2, \cdots, j_k\}, \quad 1 \leq k \leq n$$

such that all the members of C_j are pairwise or self compatible.

Each uncrossed leaf of a compatibility tree is obviously a compatibility class in accordance with Definitions 5 and 6. Thus we have the following theorem:

Theorem 3 An uncrossed leaf of a compatibility tree is a compatibility class. Any subset of this class is also a compatibility class.

Definition 7 Given a compatibility class

$$C_j = \{j_1, j_2, \cdots, j_k\}, \quad 1 \leq k \leq n$$

The compatibility class implied by C_j and x_t in X is the set

$$C_{j,t} = \{M(j_1, x), M(j_2, x), \cdots, M(j_k, x)\}$$

where each member is defined by M. Otherwise, it is deleted from $C_{j,t}$.

Definition 8 The class set P_j corresponding to the compatibility class C_j is the set of implied compatibility clasees $C_{j,t}$ for all x_t in X such that each $C_{j,t}$ satisfies the following conditions:

1) $C_{j,t}$ contains at least two members
2) $C_{j,t} \not\subset C_j$
3) $C_{j,t} \not\subseteq C_{j,s}$ if $C_{j,s} \in P_j$

If every $C_{j,t}$ does not satisfy these conditions, then $P_j = \phi$ where ϕ means empty.

3. Closed Covering Tree

In order to find the minimal state-reductions of an incompletely specified sequential machine, we shall introduce a directed graph called closed coverigg tree whose construction is based on the following conditions:

Definition 9 The number of occurrences of a state in the set of compatibility classes is called the index of the state.

Definition 10 A subset C of the set of compatibility classes covers the given flow table iff every state of the machine is con-

tained in at least one member of C.

Definition 11 A subset C of the set of compatibility classes is closed iff for all C_j in C and all x_t in X, each compatibility class implied by C_j and x_t is contained in at least one member of C.

Definition 12 Let $r, b_{i_1}, n_{i_1}, b_{i_2}, \ldots, n_{i_{k-1}}, \ell_i$ be a node-branch alternating sequence of a tree where r, b_{i_1} through b_{i_k}, n_{i_1} through $n_{i_{k-1}}$ and ℓ_i denote respectively the root, k branches, k-1 nodes and a leaf of the tree. This sequence represents a path of a tree. The length of a path is the number of branches of the path. If every node represents a compatibility class and the nodes (including root and leaf) satisfy the closure and covering properties of Definitions 10 and 11, then the associated path is called a closed covering path. A shortest closed covering path is one whose length is the shortest among all closed covering paths.

Definition 13 A closed covering tree is a branching structure, here growing from left side to right side, which satisfies the folowing conditions:

1) Each node is labeled by certain proper subset of Q (and so is the root or a leaf). This subset is simply represented by $\{ j_1, j_2, \ldots, j_r \}\ 1 \leq r < n$ and used as node name. This subset is also a compatibility class.

2) Establish the root of a tree. A compatibility class containing a state q_i of least index is chosen as the root. If there are several such classes, then there exist the same number of trees. If q_j is also of least index, a choise between q_i and q_j is arbitrarily made.

3) Determine the nodes grown from the root.

a) If the corresponding class set of the root is empty, this implies that only covering problem has to be considered. Since the root must be a compatibility class which is a proper subset of Q, there are some states being not contained in the root. Thus we choose arbitrarily a state of minimum index from among the excluded states, and all compatibility classes being equal to and containing the chosen state grow from the root as first level nodes.

b) If the corresponding class set of the root is nonempty, we consider firstly the closure problem. When the class set has a single member, all compatibility classes being equal to or containing this member grow from the root as first level nodes. If the class set has two members C_i and C_j, then all compatibility classes

being equal to or containing one of these members, say C_i, grow from the root as first level nodes. Some of the compatibility classes being equal to or containing another member C_j become second level nodes provided that the first level node from which the second level nodes are grown is not a leaf and the compatibility class of a second level node is different from the root and the first level node along a path. If the class set has more than two members, similar procedure can be followed.

4) Determine the $(i+1)$ st level nodes grown from an ith level node for $1 \leq i \leq n-2$ where n is the number of states.

a) If an ith level node associates with an empty class set or a nonempty class set each of whose members has been contained in the root or any lower level node, this implies that the closure property has been satisfied but may be not so the covering property. Thus we have to consider the covering problem. Whenever both properties have been satisfied along a path, the ith level node becomes a leaf and the path is a closed covering path. Otherwise, a similar step like 3a) can be followed to establish the $(i+1)$ st level nodes.

b) If an ith level node associates with a nonempty class set some of whose members are not contained in the root and/or any lower level node, then a similar step like 3b) can be followed.

5) Determine the termination of construction. After all ith level nodes have been grown, each of these nodes is checked by the closure and covering properties along its path to determine whether this node is a leaf or not. Whenever an ith level node becomes a leaf, all $(i+1)$st level nodes are not needed to be grown. Therefore the procedure for constructing the tree terminates.

It should be noted that a shortest closed covering path corresponds to a minimal solution of the state-reduction problem. Thus when we construct a closed covering tree, any path of length longer than that of a shortest closed covering path can be ignored. A closed covering tree constructed in this manner is said to be partially complete. By this construction method, the length of a longest closed covering path is at most n-1 where n is the number of nodes including the root and the leaf of the path. In this case, the states of the given machine are not reduced.

4. An Illustrative Example

An incompletely specified sequential machine whose states are being reduced is shown in Table I.

By Theorem 1, the set of incompatibility pairs are found to be

$$\{q_1 \not\sim q_5,\ q_1 \not\sim q_6,\ q_3 \not\sim q_5,\ q_4 \not\sim q_5,\ q_4 \not\sim q_6\}$$

By Theorem 2, the only incompatibility-pair $q_2 \sim q_6$ is implied by $q_3 \sim q_5$. Other possible incompatibility pairs implied by $q_2 \sim q_6$ must be examined. Since no such pairs are implied by $q_2 \sim q_6$, the process terminates. It is obvious that this process has finite steps and will finally terminate since the machine considered in this paper has only a finite number of states.

By Definition 5, a compatibility tree for Table I is shown in Fig. 1.

By Theorem 3, the set of compatibility classes is {{1,2,3,4}, {2,3,4},{2,3},{3,6},{2,4},{2,5},{5,6},{1,2,3},{1,2,4},{1,3,4},{1,2}, {1,3},{1,4},{3,4},{1},{2},{3},{4},{5},{6}} whose first seven members correspond to the leaves of the compatibility tree shown in Fig. 1 and whose last thirteen members are the compatibility classes obtained by Definition 6. By Definition 7, the implied compatibility classes are shown in the central four columns of Table II. By Definition 8, the class sets are shown in the rightmost column of Table II.

The index of each state is determined by counting the number of occurrences of that state in the set of compatibility classes. Table III shows the indices of all states.

By Definitions 10 and 11, the subset {{1,2,3,4},{5,6}} of the set of compatibility classes satisfies the covering property but is not closed. The subset {{1,2},{3,4},{3,6}} is closed but the state q_5 is not covered. The subset {{1,2},{3,4},{5,6}} satisfies both closure and covering properties and it will be seen that this subset constitutes a minimal solution of the state reduction problem. The set of all compatibility classes C_1 through C_{20} is obviously closed and covers the given flow table.

By Definition 13 and Table III, we may choose the compatibility classes {5},{2,5} and {5,6} as the roots, since q_5 is one of the two states of least index. Thus there are three trees to be constructed.

Fig. 2 shows a partially complete closed covering tree in which no path of length one satisfies the covering property so that second level nodes must be grown. Since there are three closed covering paths of length two whose leaves are marked by stars, third level nodes are not needed to be grown and the construction of the tree is terminated so that a partially complete closed covering tree is yielded. In Fig. 2 the broken line arrows indicate that higher level nodes can be grown when a complete closed covering tree is constructed.

Similarly, when {5,6} and {5} are used as the roots of trees,

the corresponding partially complete closed covering trees are shown in Figs. 3 and 4 in which the length of each shortest closed covering path is also two.

Figs. 2,3 and 4 show that there are five shortest closed covering paths of length two which constitute the five minimal closed coverings of the given machine. Therefore the minimal solutions of the state reduction problem for the given machine are as follows:

$\{\{2,5\}, \{3,6\}, \{1,2,4\}\}$,
$\{\{2,5\}, \{3,6\}, \{1,3,4\}\}$,
$\{\{2,5\}, \{3,6\}, \{1,4\}\}$,
$\{\{5,6\}, \{1,2\}, \{2,3,4\}\}$,
$\{\{5,6\}, \{1,2\}, \{3,4\}\}$,

5. Discussion

The method presented in this paper uses a compatibility tree for derivation of compatibility classes and used closed covering trees for obtaining all possible minimal solutions. The method is simple and systematic. We obtain five minimal solutions for the problem used as the illustrative example. The results obtained by use of other state-reduction techniques are shown in Table IV.

6. Bibliography

1. S.Ginsburg, "A Synthesis Technique for Minimal State Sequential Machines," IRE Transactions on Electronic Computers, Vol. EC-8, No. 1, pp. 13-24, March, (1959).
2. M. C. Paull and S. H. Unger, "Minimizing the Number of States in Incompletely Specified Sequential Switching Functions," IRE Transactions on Electronic Computers, Vol. EC-8, pp. 356-367, Sept., (1959).
3. J. Beatty and R. E. Miller, "Some Theorems for Incompletely Specified Sequential Machines with Applications to State Minimization," AIEE Proceedings of the Third Annual Symposium on Switching Circuit Theory and Logical Design, pp. 123-136, Sept., (1962).
4. E. J. McCluskey, "Minimum State Sequential Circuits for a Restricted Class of Incompletely Specified Flow Tables," Bell System Technical Journal, Vol. 41, pp. 1759-1768, Dec., (1962).
5. J. Beatty and R. E. Miller, "An Approach to State Minimization for Incompletely Specified Sequential Machines," IBM Research Report, RC-1055, Sept., (1963).

6. A. Grasselli and F. Luccio, "A Method for Minimizing the Number of Internal States in Incompletely Specified Sequential Networks," IEEE Transactions on Electronic Computers, Vol. EC-14, pp. 350-359, June, (1965).
7. M. S. Meisel, "A Note on Internal State Minimization in Incompletely Sequential Networks," IEEE Transactions on Electronic Computers, Vol. EC-16, No. 4, pp. 508-509, Aug., (1967).
8. A. Bouchet, "An Algebraic Method for Minimizing the Number of States in an Incomplete Sequential Machine," IEEE Transactions on Electronic Computers, Vol. C-17, No. 8, pp. 795-798, Aug., (1968).

Table I. Flow Table Taken From [2]

Present State	Next State				Output			
	x_1	x_2	x_3	x_4	x_1	x_2	x_3	x_4
q_1	-	q_3	q_5	q_2	-	1	1	1
q_2	q_5	-	-	-	0	-	-	-
q_3	q_6	q_6	-	-	0	1	-	-
q_4	-	-	q_2	-	-	-	1	-
q_5	-	q_6	q_1	q_4	-	0	0	1
q_6	q_3	-	q_2	q_3	0	-	0	1

Table III. Indices of States

State	Index
q_1	8
q_2	9
q_3	9
q_4	8
q_5	3
q_6	3

Table II. Compatibility Classes, Implied Compatibility Classes and Class Sets

Compatibility Class C_j		Implied Compatibility Class $C_{j,t}$				Class Set P
		x_1	x_2	x_3	x_4	
C_1	{1,2,3,4}	{5,6}	{3,6}	{2,5}	{2}	{{2,5},{3,6},{5,6}}
C_2	{1,2,3}	{5,6}	{3,6}	{5}	{2}	{{3,6},{5,6}}
C_3	{1,2,4}	{5}	{3}	{2,5}	{2}	{{2,5}}
C_4	{1,3,4}	{6}	{3,6}	{2,5}	{2}	{{2,5},{3,6}}
C_5	{2,3,4}	{5,6}	{6}	{2}	-	{{5,6}}
C_6	{1,2}	{5}	{3}	{5}	{2}	ø
C_7	{1,3}	{6}	{3,6}	{5}	{2}	{{3,6}}
C_8	{1,4}	-	{3}	{2,5}	{2}	{{2,5}}
C_9	{2,3}	{5,6}	{6}	-	-	{{5,6}}
C_{10}	{2,4}	{5}	-	{2}	-	ø
C_{11}	{3,4}	{6}	{6}	{2}	-	ø
C_{12}	{3,6}	{3,6}	{6}	{2}	{3}	ø
C_{13}	{2,5}	{5}	{6}	{1}	{4}	ø
C_{14}	{5,6}	{3}	{6}	{1,2}	{3,4}	{{1,2},{3,4}}
C_{15}	{1}	-	{3}	{5}	{2}	ø
C_{16}	{2}	{5}	-	-	-	ø
C_{17}	{3}	{6}	{6}	-	-	ø
C_{18}	{4}	-	-	{2}	-	ø
C_{19}	{5}	-	{6}	{1}	{4}	ø
C_{20}	{6}	{3}	-	{2}	{3}	ø

Table IV. Results of Four Minimization Methods

Crasselli-Luccio's Minimization Method	{ {3,6}, {2,5}, {1,2,4,} }
Meisel's Minimization Method	{ {3,6}, {2,5}, {1,2,4} } { {3,6}, {2,5}, {1,3,4} }
Bouchet's Minimization Method	{ {3,6}, {2,5}, {1,2,4} } { {3,6}, {2,5}, {1,3,4} } { {5,6}, {1,2}, {2,3,4} } { {5,6}, {1,2}, {3,4} }
Systematic Minimization Method	{ {3,6}, {2,5}, {1,2,4} } { {3,6}, {2,5}, {1,3,4} } { {3,6}, {2,5}, {1,4} } { {5,6}, {1,2}, {2,3,4} } { {5,6}, {1,2}, {3,4} }

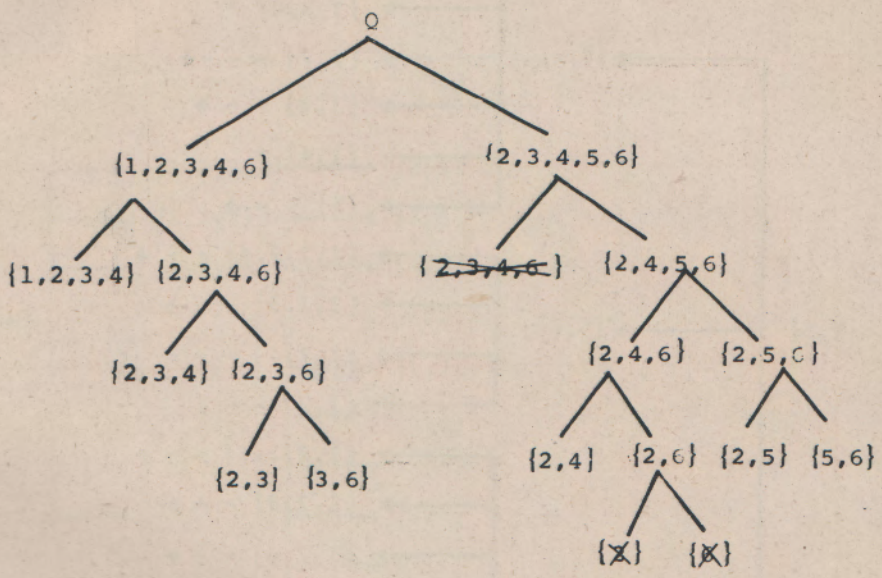

Fig. 1. A compatibilit tree

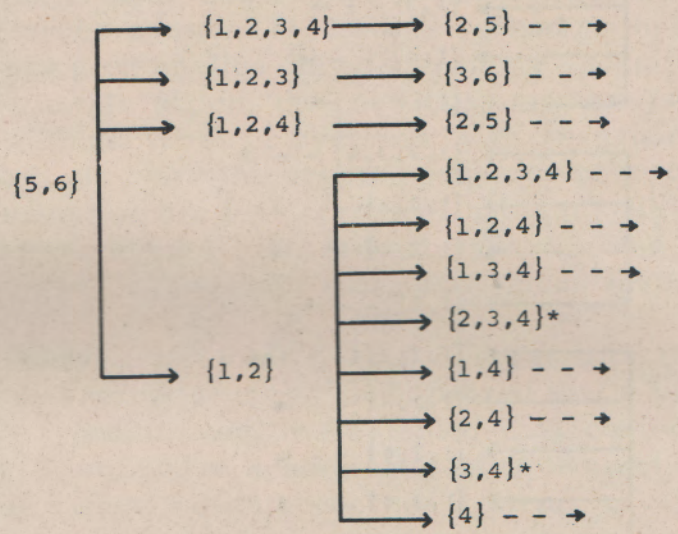

Fig. 3. Partially complete closed covering tree
—— use {5,6} as the root

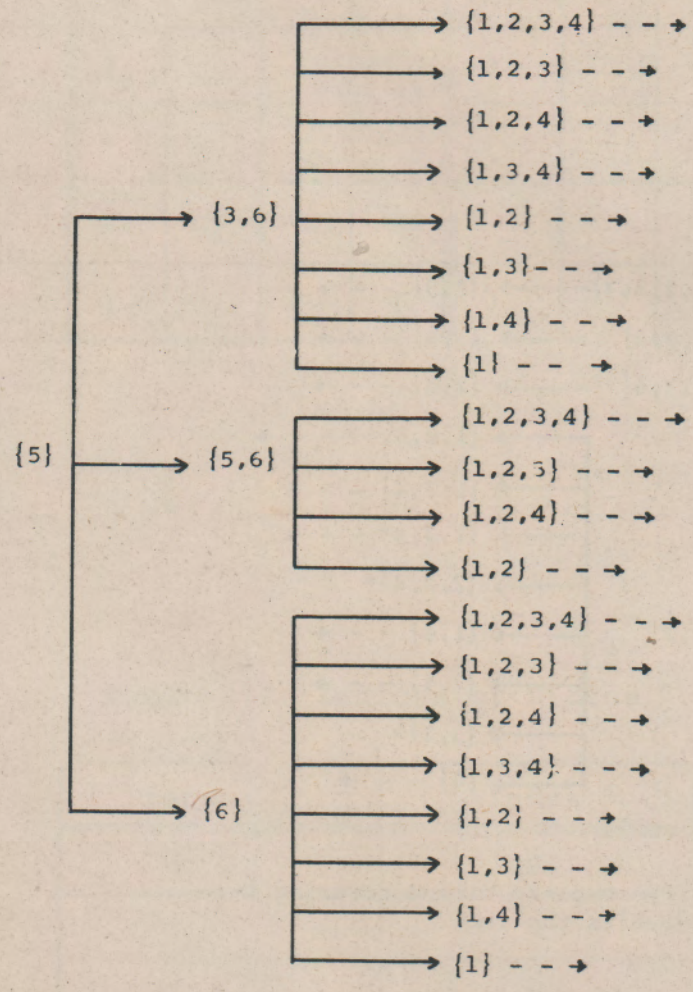

Fig.4. Partially complete closed covering tree—— use{5}as the root