IFAC

INTERNATIONAL FEDERATION
OF AUTOMATIC CONTROL

WARSZAWA 1969

Finite Automata and Switching Systems

Fourth Congress of the International Federation of Automatic Control
Warszawa 16–21 June 1969

TECHNICAL SESSION
27

Organized by
Naczelna Organizacja Techniczna w Polsce

INTERNATIONAL FEDERATION OF AUTOMATIC CONTROL

Finite Automata and Switching Systems

TECHNICAL SESSION No 27

FOURTH CONGRESS OF THE INTERNATIONAL
FEDERATION OF AUTOMATIC CONTROL
WARSZAWA 16 – 21 JUNE 1969

Organized by
Naczelna Organizacja Techniczna w Polsce

Contents

Wydawnictwa Czasopism Technicznych NOT - Polska

Zakład Poligraficzny WCT NOT. Zam. 120/69.

THE APPLICATION OF REGULAR EXPRESSIONS TO THE SYNTHESIS OF COMPLEX ASYNCHRONOUS SEQUENTIAL MACHINES

Eric DACLIN Jean-Paul PERRIN
Nicole BREAUD Michel DENOUETTE

1. INTRODUCTION

Prior to proceeding to the essential items /vital subjects/ of the present paper it seems to us very important to formulate properly the discussed problem. The problem under consideration consists in a synthesis of the transition table characterizing a sequential automatic control systems. From this point of view two kinds of problems can be discerned. The first group of problems embraces problems of characterization of the inputs and outputs of such systems /in the practice, we have usually to do with a large number of inputs and outputs and all these outputs are of an asynchronous type/. The second problem which is dependent to some extent on the first one, is the choice of a suitable algorithm of synthesis rendering it possible to solve the problems quickly enaugh and in a relatively simple and economic way.
In authors opinion, this last aspect is equally important and for this reason it will be discussed in some detail in the present paper after some short general remarks on the first two problems.

1. 1. Formulation of the problem

By automatic sequential industrial systems we will mean here as well as in the further considerations the automatic systems which can be classified as machine tools /or transfer machines/ and digital control computer. These systems are, generally speaking, characterized by a large number of input and output data and consist usually of relays /in the most cases of the "limit switch" type/ and push-buttons as far as inputs are concerned and electric motors - as far as output are concerned. It must be here noted that not all combinations of inputs are realizable from the physical point of view for instance by reason of the geographical distance.

Simultaneously, let us also note /we will return to this problem later on/ that not all input sequences are possible: generally, an automatic industrial installation, operates under the influence of a relatively small number of the input sequences which for the most part repeat peridically /in a cyclic manner/. It is worth mentioning that in the present paper we are not interested in the other class of the industrial systems which may be generally termed "automatic systems with counting"/ this group of systems covers, for instance, lifts /hoists/ with memory/. In the other words, it may be said that we are interested only in such asynchronous sequential systems whose operation can be described by a finite number of relationships between the input and output values these relationships being, in a general case, of a cyclic type.

From what we have said it is evident that the output and input values of such systems are asynchronous. However, it may also be seen that such systems have considerable dimensions. Consequently, it is important that the applied method of synthesis permits us to consider a case of asynchronous machines of a large degree of complexity. It should also render it possible to decompose /divide/ these machines into sub-machines of smaller dimensions, if possible. This decomposition is not aimed at the simplification of the Boole'an equations describing the operation of the entire system as is usually the case for the synchronous sequential systems but is rather aimed at a simplified realization by means of sub-assemblies and, in the present case, at facilitated simulation on a digital computer or logical simulator. We will explain now the choice of the method of synthesis of the table.

1.2. Choice of the method

The medium size of the machines to be realized involves an automatic synthesis by means of computer and application of a procedure which ensures a sound compromise between the time of operation and capacity of memories of the computer. Finally,

this procedure must be applicable in the case of asynchronous controls accurring in a cyclic manner.

As far as the synthesis of the tables is concerned, it is well known that there are two types of methods, and namely methods which don't assume, a priori, the reduction of the size of the table /vide [1,2] / and the methods which reduce the size of the table during the synthesis /vide [3,4] /. It is evident /vide 4/ that a considerable drawback of the latter methods consists in that they not necessarily results in a minimal machine. In spite of this, we have to focus our interest in the latter methods since in the case when we have to do with sub-machines, the size of such sub-machine is, generally speaking, small enaugh to be able to find a minimum solution. Moreover, in the course of calculations performed by a digital computer the gain of the place in the memory resulting from the application of these methods, is a considerable advantage.

The second characteristic feature of the described method consists in the possible application of this method in the case of cyclic asynchronous sequences. In our opinion, the regular expressions are best suited to the description of such relations [4,6]. . For the sake of better comprehension we have decided to apply a slightly modified Glushkov's synthesis method [4,13]. .

1.3. Plan

We will present first the method of synthesis of the tables as derived from the method by Glushkov which will have been applied in the further considerations, and we will see how it is possible to obtain decomposition of corresponding machines, the latter point being illustrated on suitable examples. As all the relevent theoretical proofs may be found in the respective literature cited in the bibliography, we have presented only the essential results, the purpose in so doing being the desire not to complicate our present paper more than it is absolutely necessary.

2. Indexing of regular expressions - synthesis of a table

2.1. General remarks

As for the next part of our paper we will take for granted that the reader is thoroughly acquainted with the methods of synthesis of the tables by Glushkov /the principles of this method and examples of application can be found in [4.13] and [14] /.

We should like to outline briefly some problems concerned with the application of the above mentioned method to the construction of an asynchronous sequential machine.

The first problem consists in an increased number of the required indices, which turn out to be necessary. In effect, it may be seen that each time an asynchronous control appears, the application of two various indices is theoretically necessary /in consequence of the mere fact that it is written in the form $X_i \left\{ X_i \right\}$ /. The second problem /which is also the problem of complexity/ is even more specific for the type of machine which we try to build.

We have to do with the cycles. This means that we will be forced to introduce a great number of indices or internal states /with the aid a universal event, I, or double iteration/ which will prove redundant during the final stage of the synthesis.

In the present section we will discuss two problems and, strictly speaking we will present results thus far obtained. The reader will find proofs of these theorems leading to the formulation of an algorithm of synthesis for the table of an asynchronous machine in the references 8 and 14.

2.2. Expressions of type P_i^{1*}

The regular expressions $R = pp^*$ can be written, accordance with the assumed convention, in the form p^{1*}. If P is a vector /sequence of length 1/ P^{1*} is an asynchronous control /vide 7 , 8 and 13 /. We will index p by attributing to it one fundamental index. However, we are well aware of the fact that the place, i, thus obtained corresponds to a stable

state and, consequently, we write index j in the field (P, j) of the Moore's table. In the case, if p is a sequences of length greater than 1, stability has a global character. This means that if $P = ABC \ldots N$ the fundamental /basic/ index associated with N will be a pre-fundamental /prebasic/ index associated with A. Beginning from this moment, the indexation of the regular expressions describing the machine will follow strictly the Glushkov's scheme with a sole exception concerning the stability of each of attained states.

2.3. Universal event. $I = i^x$

The number of indices which appear in the course of making use of a universal event i^x is reduced in an identical way with due regard being paid to the fact that in the expression $R = i^x aF$ /where a has unit length/ all indices appearing at i^x are equivalent /they correspond to the state of repose of the machine which is realized by R/. It is only one fundamental /basic/ index connected with a that plays another role since it indicates that we are in the sequence aF.

If, during indexation, to i^x was attributed index 1, it is clear that this index must be repeated in all fields of the given table. On completion of the above mentioned operation, further course of indexation will be identical as in the case of the non-modified Glushkov's method.

2.4. Example

We will now give an example illustrating advantages obtained due to application of two rules discussed previously. The problem consists in building an asynchronous m chine with the three inputs X_1, X_2, and X_3 /which can be combinations of variables X_i, $X_j \ldots\ldots X_m$/ and one output Z.
$Z = 1$ for each command X preceded by X_1 and X_2.

2.4.L. Non-modified Glushkov's method

We can write:

$$R/Z/ = / X1 + X2 + X3/^X \; X1 \; X1 \; ^x \; X2 \; X2 \; ^x \; X3 \; X3$$

Upon indexing R /we will assume notation $\{\ \}$ for iteration with the purpose of a graphical representation of the iteration

in as simple a way as possible/.

$$R/Z/ = |\{|x_1| + |x_2| + |x_3|\}|x_1|\{|x_1|\}|x_2|\{|x_2|\}|x_3|\{|x_3|\}|$$
$$\begin{matrix} 0\,0 & 1\,0 & 2\,0 & 3\,0 & 1\,1 & 4\,1 & 5\,5 & 6\,5 & 7\,7 & 8\,7 \end{matrix}$$

$$\begin{matrix} 1 & 1 & 1 & 1 & 4 & 4 & 6 & 6 & 8 & 8 \\ 2 & 2 & 2 & 2 & & & & & & \\ 3 & 3 & 3 & 3 & & & & & & \end{matrix}$$

The table of transitions /non-reduced/ is presented in fig. 2.1. It contains nine internal states.

2.4.2. Modified method

We may write:

$$R/Z/ = i^{x} \; x_1^{1*} x_2^{1*} x_3^{1*}$$
$$1\;1 \quad 2 \quad 3 \quad 4$$

and the obtained table has identical form as presented in fig. 2.2. It has only four internal states.

2.5. Reduction of the number of required indices.

In order to reduce, at the beginning, the number of indices applied when preparing the table of the system, we will use only the rules of similarity and correspondence as given by Glushkov. Let us recall these rules.

R_1: Similarity – Two places are similar, if they depend only upon identical assemblies of prefundamental /pre-basic/ and final places.

R_2: Correspondence – Two corresponding places are place of different regular expressions or of different terme included in the same pair of parentheses to which lead identical sequences beginning from the initial place or from place situated directly in front of the parenthesis.

Let us also note that in order to reduce the number of indices encountered in a regular expression one can make use of only the conditions either R1 or R2 at one time: it is not possible to make use of both conditions R1 and R2 simultaneously.

The application of condition R_2 presents no difficulties /is quite easy/ contrary to the application of R_1 which is of far more delicate nature. In particular, /14/ decomposition of terms in a regular expression into the individual factors creates usually serious difficulties. It is also advantageous to make use of a more restricting condition R_1 rather than to use directly R_1.

R_1 Equality. Indices of two places can be identified in the following conditions:

A. The indices should be attributed to the fundamental places of the same input vector.

B. They should not be indices of the final places of various regular expressions.

C. The corresponding fundamental places for each input vector which can be put in the pre-fundamental places with the considered indices should be:
   - designated with identical indices,
   - fulfil the conditions C and B.

This renders possible to determine directly the table of transitions which in the case of the systems under consideration has only one stable state in a line /14/.

3. DECOMPOSITION OF ASYNCHRONOUS SEQUENTIAL MACHINES

3.1. Formulation of the problem

The problem of decomposition of the sequential machines such as for instance machines which we try to realize, may by considered in two ways. The first way consists in the application of purely sequential techniques, relative /however slightly different from/ the techniques as presented by HARTMANN AND STEARNS.

The second way may be reduced, upon finishing the synthesis of the table and coding the table, to searching for the

decomposition of the obtained Boolean functions. On the other hand, let us note that both above mentioned procedures seem rather hardly applicable in the discussed case,due to a considerable size of the systems under consideration and a comparatively large number of indeterminations in the table on the level of considerations on the partition /vide[11] / and, subsequently, owing to a large number of non-specified inputs which manifest themselves by the presence of so many void /empty/ places in the respective Boolean matrices /vide [12]/.

Consequently, we tried to combine both above mentioned operations.

At the very beginning we have considered the problem of the sequential decomposition on a level of the regular expressions defining the system. Later, having obtained various sub-machines into which the entire automatic system can be eventually divided, we tried to obtain as quick as possible the equations of these sub- machines by considering as the most essential factors, the speed and ease of obtaining of these equations.

3.2. Sequential decomposition.

3.2.1. Notation

We will assume that the regular expressions characterizing operation of the machine which should be realized are known. If controls of this system are physical variables /a_1, ...a_n/. and outputs - /z_1, ... z_p/ then we will designate each from the full monomials a_1, a_2, ... a_n with the letter A_j for the sake of simplification of the expression. Letter A_j will therefore represent the corresponding asynchronous control A_j^x.
In a similar way, each full monomial z_1, z_2, ...z_p is designated by letter Z_k. A_j will be called symbolic inputs and Z_k - will be termed symbolic outputs.

The entire automatic control system which should be realized is thus represented by a series of regular expressions of the form $Z_i = F\ /A_1, .. A_m/$ for i = 1,N, where N is the number of the symbolic outputs and m denotes the number of the

symbolic inputs.

3.2.2. Independent machines

Classes of disjoint inputs

At the beginning let us divide the entire system into disjoint sub-machines. To this aim we must define previously these sub-machines. In an arbitrary way we have assumed the following principle:

For each of the symbolic outputs Z_i we establish the class of the symbolic inputs C_i /A_1, ..., A_m/ which produces output Z_i. Consequently, we have N classes of the above mentioned type. Next, let us combine all these classes in pairs. Each time when the intersection of the both classes C_i and C_j is not equal to zero, a new class $C_{ij} = C_i \cup C_j$ is formed and C_i and C_j are eliminated from the table of comparison. The presented procedure is then repeated and new classes are formed. Finally, by proceeding in this way, we arrive at /$M \leq N$/ input classes C_1 associated in conformity with the method of generation, with M classes of outputs C_1 and Γ_1 /including the symbolic inputs and outputs. We have then:

$$C_k \cap C_l = \emptyset \qquad \forall_k \neq l$$

$$\Gamma_i \cap \Gamma_j = \emptyset \qquad \forall_i \neq j$$

$$i, j, k, l = 1 \text{ to } M$$

Therefore, it may be seen that the machine to be realized is divided into independent sub-machines M_1, ... M_m /independent in a sense that they have no common inputs/ and may be presented not necesarily in the form as in fig. 3.1. but in the form given in fig. 3.2. It is worth mentioning that when we remain on the level of the symbolic inputs and outputs of the system we will gain probably nothing, both from the point of view of complexity as well as from the point of view of simplification of the equations. In effect, the inputs belong further to the alphabet /A_1, A_2, ... A_m, B_1, ...B_n/., where B_i represent void /impossible/ combinations a_1, a_2... a_n.

Simplification of the equations of sub-machine M_i containing inputs $/A_{i1}, A_{i2}, \ldots A_{ik}/$ may be achieved by attributing the value 0 to the functions which should be calculated for all imputs A_j such that A_{jn} $/A_1, \ldots A_m/$ and value Ø at inputs B_i that is by proceeding in a similar way as in the case of a general /global/ synthesis.

3.3. Combinatorial decomposition

3.3.1. Principle of the method

We aim at as great a facilitation of work connected with writing of the logical equations of the individual sub-systems as possible. From this point of view it is absolutely essential that the number of inputs of sub-machines be as small as possible /only in this case the reduction of the entire system will be realized quickly enaugh/. This aim can be achieved by assuming that the real inputs $/a_1, a_2, \ldots a_n/$ are divided into two classes and namely class which remains unchanged during operation of the sub-machine M_i and class wich changes it's value in the course of work of the machine. To some extent it may be said that changes involved in M_i are independent, of the inputs belonging to the first from the above mentioned classes.
Consequently, if the internal variables and outputs M_i can be expressed by the function of the inputs of the second from the above mentioned classes then a considerable economy of time and place will be obtained, since the number of columns in each table will be considerably reduced.

3.3.2. Case of outputs

For the same reason as previously mentioned, and namely for the sake of as far simplification of the conceptual work as possible, we will confine out considerations in the first place to the symbolic inputs. It is generally possible, with the use of an OR system, to deduce each real output from the symbolic outputs in conformity with the scheme of the types as shown in firg. 3.3.

3.3.3. Case of inputs

Let us consider sub-machine M_i having symbolic inputs

$A_{i1}, A_{i2}, \ldots A_{il}$ /whereas each A_{ij} is a full monomial on the basis $/a_1, a_2, \ldots a_m/$. By using the method of succesive comparisons between A_{ij} two classes of the real outputs can be obtained as mentioned above. Let $/a_{i1}, a_{i2}, \ldots a_{ip}/$ be a sub-machine of invariant inputs and let $/a_{v1}, a_{v2}, \ldots a_{vn}/$ /a + p = m/ represent the class, of variable inputs. Should the symbolic outputs M_i be $Z_{i1}, Z_{i2}, \ldots, Z_{ir}$ and the internal variables $Y_{i1}, Y_{i2}, \ldots, Y_{is}$ then it may be written:

$$Z_{i1} = F_{i1} /a_{v1}, a_{v2}, \ldots, a_{vn}, y_{i1}, y_{i2}, \ldots y_{is}/$$

$$Z_{i2} = F_{i2} /a_{v1}, a_{v2}, \ldots, a_{vn}, y_{i1}, y_{i2}, \ldots y_{is}/$$

$$Z_{ir} = F_{ir} /a_{v1}, a_{v2}, \ldots, a_{vn}, y_{i1}, y_{i2}, \ldots y_{is}/$$

$$y_{i1} = G_{i1} /a_{v1}, a_{v2}, \ldots, a_{vn}, y_{i1}, y_{i2}, \ldots y_{is}/$$

$$y_{i2} = G_{i2} /a_{v1}, a_{v2}, \ldots, a_{vn}, y_{i1}, y_{i2}, \ldots y_{is}/$$

$$y_{is} = G_{is} /a_{v1}, a_{v2}, \ldots, a_{vn}, y_{i1}, y_{i2}, \ldots y_{is}/$$

Mowever, it has to be born in mind that the above mentioned equations are only satisfied for the given combinations of $a_{i1}, a_{i2}, \ldots a_{ij}$.

Taking into account the above mentioned conditions we have two possible solutions.

We can produce a Boolean product of each Z_{il}, Y_{jk}, by means of a combination a_{ij} characterizing M_i /however this may prove rather troublesome/ or to try to reduce still more the combination a_{ij}. In this connection two procedures can be considered. The first procedure consists in realizing, a partial minimalization of the members $\pi \tilde{a}_{ij}$, by means of the conventional /classical/ methods and making use of all the empty /void/ combinations which are not present on any of the symbolic inputs/ $A_1, A_2, \ldots, A_m$/. The second way consists in the testing if switching on and off of M_i is performed under the influence /effected/ of one from variables belonging to the set a_{ij}. If both these variables are designated as $\tilde{a}_{il}$

and $\tilde{a}_{i2}$, this influence corresponds to the change in the value of $\tilde{a}_{i1}$ from 0 to 1 when passing from the symbolic input A_{ij} /of sub-machine M_i/ to the symbolic input A_{i1} /symbolic input which switches on M_i/, and to the change in the value a_{ir} from 1 to 0 when the symbolic input A_{ij} becomes the symbolic input A_{kl} /switching on M_k/. If the machine M_i is to be switched on several times at subsequent instants or if switching on /or switching off/ of this machine depends on variables a_{vk} only, then the application of the method as described above is impossible and the method of partial minimalization of a_{ij} must be applied.

An overall layout of the machine M can be presented as in fig. 3.4. where the variables of switching off and on of each sub-machine have been presented separately.

4. Example of application. Setting of blades on a steel tube

4.1. Problem

It is our aim to mount blades on an U-shaped steel tube. For this purpose the tube has been fastened in three vices /fig. 4.1./ pressed by three hydraulic cylinder V_1, V_2, V_3.

The working cycle consists of the following operations:

1. Switching on of circuit-breaker, pressing of V_1
2. Action on m. Presing of V_2
3. At the end of pressing of V_2 /with the use of the stop a/ pressing of V_3
4. At the end of pressing of V_3 /with the use of the stop b/ switching in of V_4 releasing the part from the store
5. At the end of the stroke of V_4 /with the use of a retractable stop/ withdrawal of V_4 and start of V_5
6. Contact on d /switching contactor/ releasing of V_2
   Contact on e releasing of V_3
7. Contact on limit switch f /moving stroke by stroke in the course of mounting of the blades by means of a system with ratched wheel connected rigidly with V_5/.
   Withdrawal of V_5.
8. Contact reversed on e: pressing of V_3

9. Contact reversed on d: pressing of V_2
10. At the end of the stroke of V_5 /with the use of g/: start for taking the next blade /return to 4/.

4.2. Establishment of regular expressions

The table describing the operation of the system is presented in fig. 4.2. The column on the right, hand side represents the symbolic outputs. To exhaust the problem under consideration let us note that fig. 4.3. presents a combination /diagram/ built from OR circuits, which enables the symbolic outputs to be transformed into the real ones.
On the basis of the table presented in fig. 4.2. conclusion can be drawn as to the working cycle of the entire system /attention should be paid to the presence of a transient state/ AB C B D Z_o Z_1 Z_2 Z_3/.
Ten regular expressions of the symbolic outputs have the following form:

$R(Z_1) = AB$

$R(Z_2) = AB/C + CB/$

$R(Z_3) = ABCBD + ABCBDE(F + RF)$

with $R = FENGHJKLKLPHMNE$

$R(Z_4) = ABCBD(E + ER^{1*})$

$R(Z_5) = ABCBDE(F + R^{1*}F)(E + EN)$

$R(Z_6) = ABCBDE(F + R^{1*}F)EN(G + GH)$

$R(Z_7) = ABCBDE(F + R^{1*}F)ENGH(J + JK)$

$R(Z_8) = ABCBDE(F + R^{1*}F)ENGHJK(L + LK)$

$R(Z_9) = ABCBDE(F + R^{1*}F)ENGHJKLK(P + PH)$

$R(Z_{10}) = ABCBDE(F + R^{1*}F)ENGHJKLKPH(M + MN)$

These regular expressions render it possible to formulate the scheme of decomposition into four sub-machined presented

in fig. 4.4. /a. scheme: B: tables of phases and outputs/.

4.3. Decomposition into real controls

As far as the complexity of the entire system is concerned, we have gained nothing /even in the case if the symbolic inputs and outputs/ and have even complicated some equations as compared a general solution to the problem /we do not attach here with the relevent equations in order not to complicate the text/.

Now we will consider, subsequently, four sub-machines and present their equations.

a/ Sub-machine ML

Variable inputs: I, m. New table are presented in fig. 4.5.a ML is switched on as soon as g passes from 0 to 1. M1 is switched off when a passes from 0 to 1.
From the tables presented in fig. 4.5.a. it may be derived:

$$Z_1 = I\,\bar{m}\,\bar{y}_1$$
$$Z_2 = Iy_1 + m$$
$$Y_1 = Iy_1 + m$$

b/ Sub-machine M2

Invariants: d, e, F, I, m : /fig. 4.5.b/
M2 is switched on either when a passes from 0 to 1 /Z_3, Z_4, Z_5/ or when d passes from 1 to 0 /Z_{10}/ and switched off when d passes from 0 to 1.
The equations have the following form:

$$Z_3 = c + \bar{b}\,\bar{y}_2$$
$$Z_4 = b\,c\,\bar{y}_2$$
$$Z_5 = a\,\bar{c}\,y_2$$
$$Z_{10} = \bar{g}\,\bar{y}_2$$
$$Y_2 = c + ay_2$$

c/ Sub-machine M3

Invariants: I, m, c, d, e, f, g /tables: fig. 4.5.e/

M3 is switched on either when d passes from 0 to 1 /Z_6/ or when e passes from 1 to 0 /Z_9/
and switched off either when e passes from 0 to 1 or when d passes from 1 to 0.

Corresponding equations:

$$Z_6 = b\,\bar{y}\,z$$
$$Z_9 = \bar{b} + \bar{a}y_3$$
$$Y_3 = \bar{b} + \bar{a}y_3$$

5. Conclusion

Several problems concerned with the synthesis of complicated asynchronous machines which were discussed above can be regarded only as the first approximation of the problem under consideration. On the other hand, it seems that these remarks in the existing form should facilitate and thus speed up the entire process of formulation of a logical scheme, because they afford the possibility of omitting tedious and cumbersome transformations of the tables of large dimensions. On the other hand, should the results obtained be not absolutely minimum in the strict sense, it can be shown that a decomposition into sub-machines enables easier simulation and facilitates repairs in all cases when the loops encountered in a logical scheme are referred to sub-assemblies of small dimensions.

Consequently, the described procedure can be fully applicable only in the case when it can be extended also on the case when the symbolic inputs may not be divided into disjointed classes. In consequence, there remains still plenty of work to be done both in this field as well as in the field of a more precise formulation of the principles of operation of a sub-machine.

Finally, as far as the problem of synthesis of the table is concerned, still much is to be gained by the reduction of the dimensions so as to obtain directly a diminished table.

	X_1	X_2	X_3	Z
1	2	3	4	0
2	5	6	4	0
3	2	3	4	0
4	2	3	4	0
5	5	6	4	0
6	2	7	8	0
7	2	7	8	0
8	2	8	9	1
9	2	3	9	1

Figure 2-1

	X_1	X_2	X_3	Z
1	2	1	1	0
2	2	3	1	0
3	2	3	4	0
4	2	1	4	1

Figure 2-2

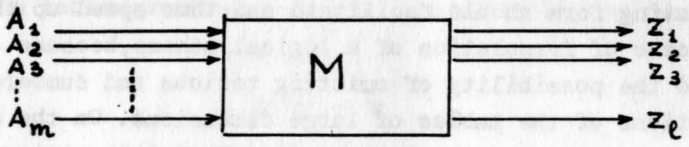


Figure 3-1

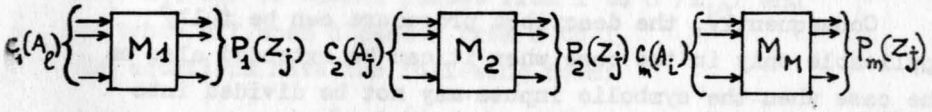


Figure 3-2

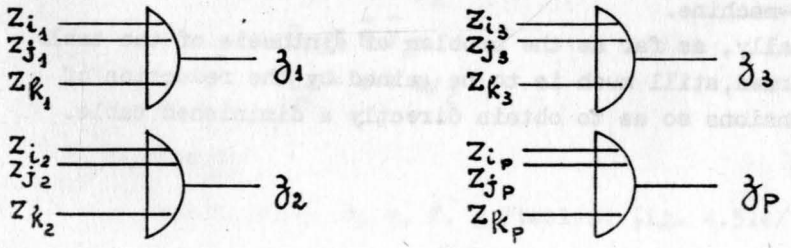


Figure 3-3

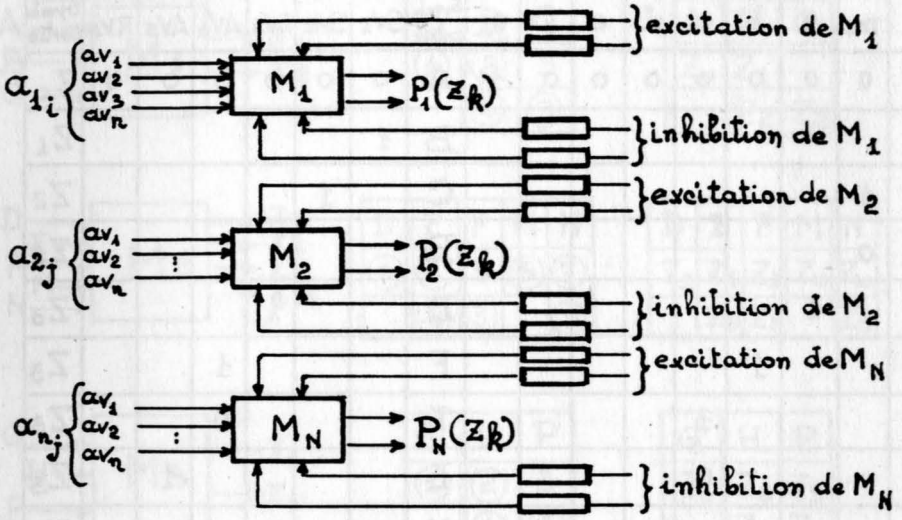


Figure 3-4

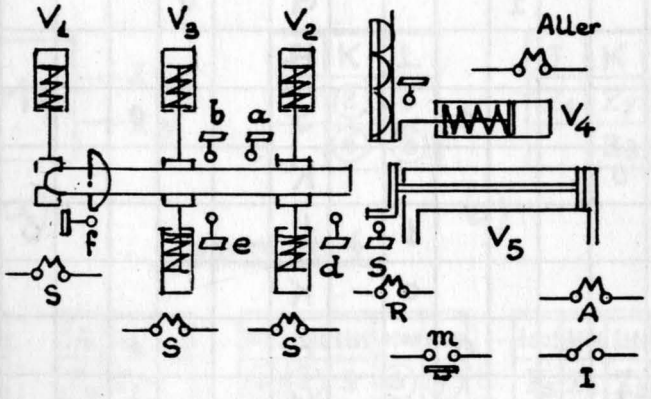


Figure 4-1

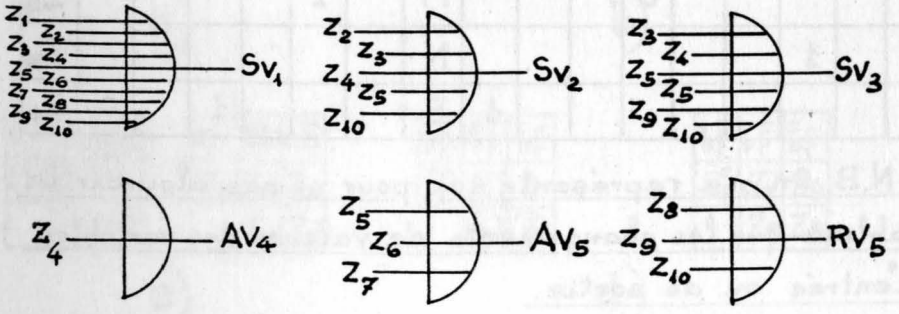
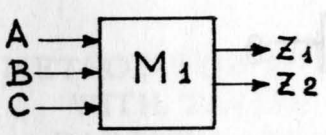


Figure 4-3

I	m	a	b	c	d	e	f	g	Symb. entrée	SV1	SV2	SV3	AV4	AV5	RV5	Symb. Sortie
0	0	0	0	0	0	0	0	1	A	0	0	0	0	0	0	Z_0
1									B	1						Z_1
	1								C		1					Z_2
	0								B							Z_2
		1							D			1				Z_3
			1						E				1			Z_3
				1					F				0			Z_3
				0					E					1		Z_5
								0	N							Z_5
					1				G		0					Z_6
		0							H							Z_6
						1			J			0				Z_7
			0						K							Z_7
							1		L					0	1	Z_8
							0		K							Z_8
						0			P			1				Z_9
			1						H							Z_9
					0				M		1					Z_{10}
		1							N							Z_{10}
								1	E				1		0	Z_4

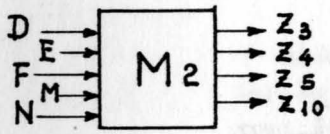
N.B. On n'a représenté ici, pour ne pas alourdir le tableau que les changements de valeurs des variables d'entrée ou de sortie.

Figure 4.2



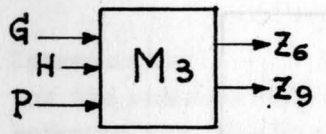
A	B	C
①	②	3
	④	③

A	B	C
Z_0	Z_1	
	Z_2	Z_2



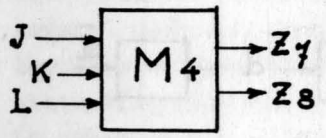
D	E	F	M	N
①	②	3	⑥	⑦
	④	③		⑤

D	E	F	M	N
Z_3	Z_4	Z_3	Z_{10}	Z_{10}
	Z_5	Z_3		Z_5



G	H	P
①	②	3
	④	③

G	H	P
Z_6	Z_6	Z_9
	Z_9	Z_9



J	K	L
①	②	3
	④	③

J	K	L
Z_1	Z_7	Z_8
	Z_8	Z_8

a) b)

Figure 4-4

$\bar{I}\bar{m}$	$I\bar{m}$	Im
①	②	3
	④	③

$\bar{I}\bar{m}$	$I\bar{m}$	Im
Z_0	Z_1	
	Z_2	Z_2

1001	1101	1111	0100	1100
①	②	3	⑥	⑦
	④	③		⑤

1001	1101	1111	0100	1100
Z_3	Z_4	Z_3	Z_{10}	Z_{10}
	Z_5	Z_3		Z_5

a) b)

ab	$\bar{a}b$	$\bar{a}\bar{b}$
①	②	3
	④	③

ab	$\bar{a}b$	$\bar{a}\bar{b}$
Z_6	Z_6	Z_9
	Z_9	Z_9

$b\bar{f}$	$\bar{b}\bar{f}$	$\bar{b}f$
①	②	3
	④	③

$b\bar{f}$	$\bar{b}\bar{f}$	$\bar{b}f$
Z_7	Z_7	Z_8
	Z_8	Z_8

c) d)

Figure 4-5

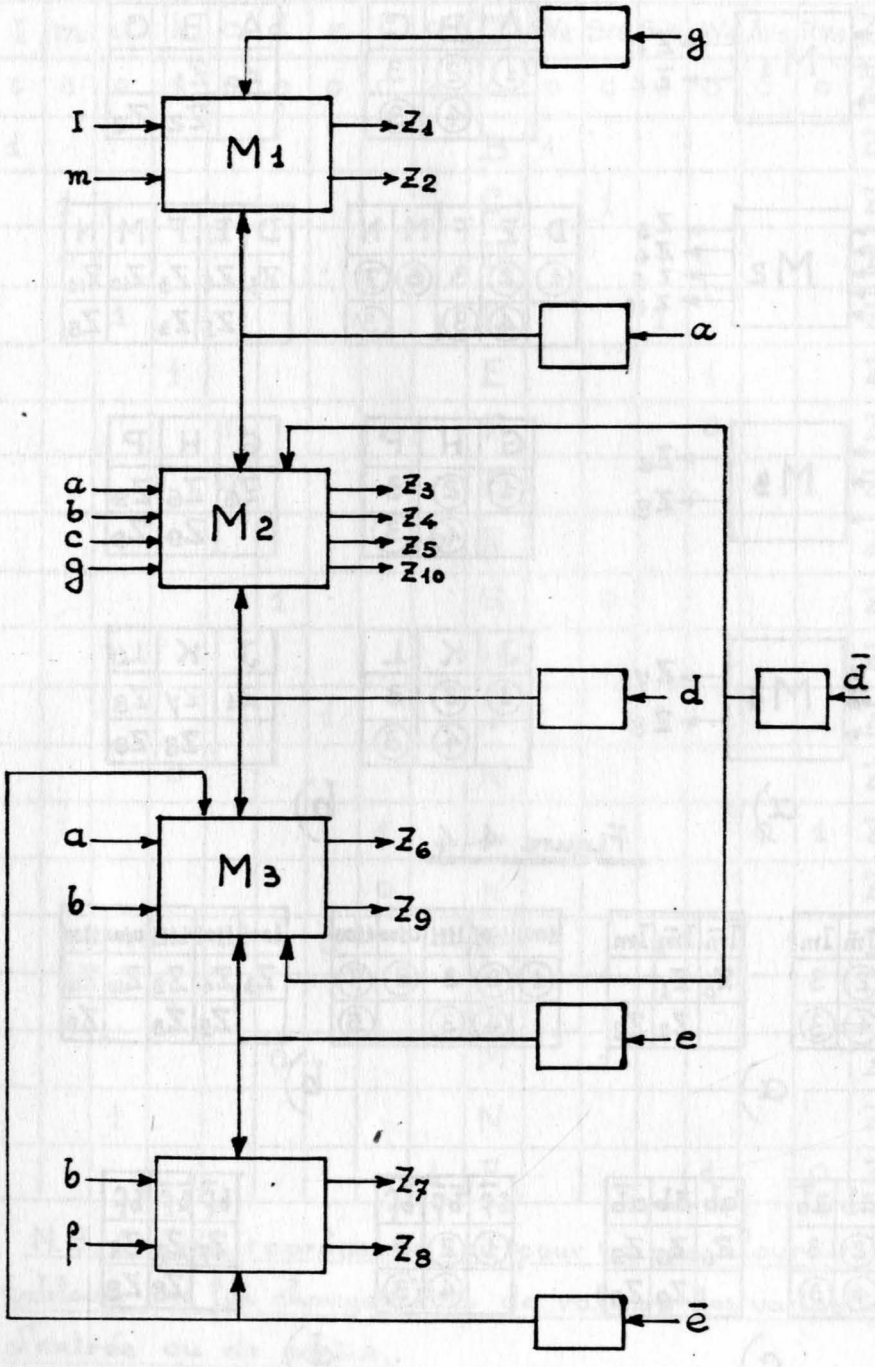


Figure 4.6

METHOD FOR STATE REDUCTION OF AUTOMATA WITH TAKING INTO ACCOUNT TECHNICAL PARTICULARITIES OF SYNCHRONOUS AND ASYNCHRONOUS OPERATIONAL MODES

Hans Joachim Zander
German Academy of Science
Institute for Automatic Control
Dresden, GDR

Introduction

For the construction of industrial control devices asynchronous automata are mainly used. However for the state minimization of asynchronous automata the methods[1,2,3,4,5] especially developed for synchronous automata are not suited. Therefore several authors tried to take into account certain particularities of asynchronous automata with the methods known for state minimization of synchronous automata or tried to develop in other ways special methods for minimization of asynchronous automata[6,7,8,9,10,11]. In this connection also such asynchronous automata are of great importance, in which the transitions to the following states especially depend on the change of the input signals[8,13].

Beginning with a structure analysis in this paper the different kinds of synchronous and asynchronous automata used in the fields of control and computation are intended to be considered from a unified point of view. According to these considerations a method is developed, which allows to determine systematically for any kinds of automata all the solutions with a minimal number of internal states. The technical particularities of the different kinds of automata appear in different conditions under which two of their states are incompatible.

1. Operational modes of automata

1.1 Basic considerations

In order to take into account all the kinds of automata used in the fields of control and computation and to be able to consider them from an unified point of view it is necessary to define the term "automaton" some more completely than it is generally the use in the theory of automata[1].

Definition 1:
An automata $A = A(\mathfrak{A}, \mathfrak{X}, \mathfrak{Y}, g, f)$ is determined by the set $\mathfrak{A}$ of its internal states A, the set $\mathfrak{X}$ of its input combinations X, the set $\mathfrak{Y}$ of its output combinations Y, by a transition function g and an output function f. The transition function g is a mapping of the set $\mathfrak{A} \times \mathfrak{E}$ into the set $\mathfrak{A}$ with $\mathfrak{E} = \mathfrak{X} \cup (\mathfrak{X} \times \mathfrak{X})$. The output function is called f_1, if it is a mapping of the set $\mathfrak{A}$ into the set $\mathfrak{Y}$. The output function f is called f_2, if it is a mapping of the set $\mathfrak{A} \times \mathfrak{E}$ into the set $\mathfrak{Y}$. An automaton, that works according to an output function f_1, is to be called an A-automaton and an automaton, that works according to an output function f_2, is to be designated as AX-automaton.

By enlarging the domain of definition of the functions g and f_2 to $\mathfrak{A} \times \mathfrak{E}$ with $\mathfrak{E} = \mathfrak{X} \cup (\mathfrak{X} \times \mathfrak{X})$ also such kinds of automata are taken into account as contain, in addition to memory elements and logic elements, also differentiating elements[8,13]. Such kinds of automata have great importance in the fields of control

In dependance on whether the transitions fixed by the transitio function g are caused by a special clock pulse or by the input combination itself one distinguishes between a clocked and an unclocked operational mode (usually one finds for that the term synchronous and asynchronous operational mode). For a classification of the different kinds of automata used in the fields of control and computation the following characteristic feature are specially important.

1.2 α-unclocked and ß-unclocked operational mode

As to the exciting combination E_g, by which for instance the transition into the following state according to the transition function

$$^{k+1}A = g(^{k}A, {}^{k+1}E_g) \qquad (1)$$

is determined, it is necessary to distinguish between two specially important kinds of automata.

Definition 2:
If especially

$$^{k+1}E_g \equiv {}^{k+1}X \qquad (2)$$

we call the automaton an α-unclocked automaton. We designate an automaton as a ß-unclocked automaton, if the exciting combination $^{k+1}E_g$ is one-to-one assigned to the two input combinations ^{k}X and ^{k+1}X by a function h as follows:

$$^{k+1}E_g = h(^{k}X, {}^{k+1}X) = {}^{k+1}X', \qquad (3)$$

The dynamic input combination $^{k+1}X'$ can be regarded as a term the value of which depends on two sequential assignment mappings of the input variables. This state of affairs can technically be realized by means of differentiating elements[13].

1.3 Dynamic and static operational mode

On the strength of technical facts the behaviour of an automaton furthermore depends on the ratio between the effective duration of the signals used for changing the memory elements and certain parameters of these memory elements. If one intends to change a binary element from a stable state to another, one has to convey to it a certain energy by a set or a reset signal. In order to abstract from the technical facts in a degree sufficient

for the considerations to be made in this paper let us suppose, that for changing the memory elements in general exciting signals of a determined amplitude-time-surface Q are necessary. The actually existing amplitude-time-surface of the exciting signals belonging to $^{k+1}E_g$ is designated with F. If $F \geq Q$, the memory elements can be changed over whereby the new state is stored according to $^{k+1}A = g(^{k}A, {}^{k+1}E_g)$. In fig. 1 this situation is illustrated in a timing diagram for a special sequential circuit. By a signal x_E the memory element is excited (x_s = threshold value), so that after reaching the necessary voltage-time-surface Q the output signal z under idealized conditions abruptly changes its level. At the end of the transition period from $^{k+1}E_g$ and the new formed state ^{k+1}A new exciting signals originate which in the case of $F \geq 2Q$ are able to change the memory elements over for a further time. In the example according to fig. 1 the considered memory element itself is changed, although according to the problem it is to be kept excited also further on.

So the different behaviour of automata as mentioned above may result from the ratio between the real amplitude-time surface F of the exciting signals and the required amplitude-time surface Q. In this connection a dynamic and a static operational mode is defined here.

Definition 3:
If the relation

$$Q \leq F < 2Q \tag{4}$$

holds for the amplitude-time surface F of the exciting signals, we speak of a dynamic operational mode, and if

$$F \geq 2Q \tag{5}$$

we speak of a static operational mode.

If a stable state is to prevail in the tact interval k+1 you must prevent that the exciting signals formed in compliance with the exciting combination ${}^{k+1}E_g$ and the new state ${}^{k+1}A$ can change the memory elements over again. With a dynamic operational mode the effective length of the exciting signals is limited according relation (4) by using technical resources so, that a further changing-over of the memory elements is impossible.

In a static operational mode it is necessary, that to the exciting combination ${}^{k+1}E_g$ and the state ${}^{k+1}A$ formed according to ${}^{k+1}A = g({}^{k}A, {}^{k+1}E_g)$ the state ${}^{k+1}A$ is assigned again, so that no further transition takes place and the state ${}^{k+1}A$ is stable in the tact interval k+1:

$$ {}^{k+1}A = s({}^{k+1}A, {}^{k+1}E_g) \qquad (6) $$

The function s is designated here as stability function. It represents a restriction within the transition function and describes the stable states in the static operational mode, as in first line you find it in "asynchronous" automata (paragraph 2.2).

2. Combined operational modes of automata

In the paragraphs 1.1 to 1.3 some technical particularities were compared, that are especially important for state minimization, however, were not sufficiently taken into account in the hitherto existing synthesizing methods. If one logically combines the listed classification features one gets a number of different kinds of automata. Here only four of the examined kinds of automata shall be described, for which then in paragraph 3 the conditions for the incompatibility of their states will be given.

2.1 Dynamically clocked AX-automata ("synchronous" automata)

For dynamically clocked AX-automata usually also designated as Mealy-automata the following system of transition and output functions is valid:

$$^{k+1}A = g(^{k}A, {}^{k}X) \tag{7}$$

$$^{k+1}Y = f_2(^{k}A, {}^{k}X) \tag{8}$$

As a basis for the technical realization of this kind of automata you can take the block diagram according fig. 2. In this case when a clock pulse i_t arrives the output combination ^{k+1}Y is formed from the preceding state ^{k}A and the input combination ^{k}X still being present from the tact interval k. ^{k+1}Y is defined only during the width of the clock pulse (pulse output signals). At the trailing edge of the clock pulse the new state ^{k+1}A originates from ^{k}A and ^{k}X, which will be stored in the memory block S. In order to limit the effective width of the exciting signals (dynamic operational mode) the differentiating element D is added (N = inverter).

2.2 Statically α-unclocked AX-automata ("asynchronous" automata)

For statically α-unclocked automata, which usually and strictly speaking are designated as "asynchronous" automata, the following system of transition and output functions holds:

$$^{k+1}A = g(^{k}A, {}^{k+1}X) \tag{9}$$

$$^{k+1}Y = f_2(^{k+1}A, {}^{k+1}X) \tag{10}$$

Because of the static operational mode there exists the stability function

$$^{k+1}A = s(^{k+1}A, {}^{k+1}X) \tag{11}$$

In fig. 3 the appertaining block diagram is shown.

2.3 Dynamically and statically ß-unclocked AX-automata

To dynamically ß-unclocked AX-automata applies:

$$^{k+1}A = g(^{k}A, {}^{k+1}X') \tag{12}$$

$$^{k+1}Y = f_2(^{k+1}A, {}^{k+1}X) \tag{13}$$

The dynamic input combination $^{k+1}X'$ can be realized by means of differentiating elements[13]. For a dynamic operational mode you must dimension the time constants of the differentiating elements in such a manner, that the amplitude-time surface F satisfies the relation (4). For a static operational mode the stability function is:

$$^{k+1}A = s(^{k}A, {}^{k+1}X') \tag{14}$$

Fig. 4 shows the pertaining structural scheme.

3. Conditions for incompatibility of states

To be able to determine the compatible or the incompatible states of the different kinds of automata it is necessary to define the notion "compatibility" in a more general way. Strictly speaking the usual definition of the notion compatibility[2] is applicable only to dynamically clocked automata (look par. 2.1).

Here we start from the fact that in the course of the synthesis certain secondary combinations the elements of which correspond to the output signals of memory elements are assigned to the

internal states A. To have need for that of as few memory elements as possible and as few secondary combinations as possible, respectively, you can try to assign one and the same secondary combination to as many states as possible without changing the over-all behaviour of the automaton considered from outside. With these reflections we obtain for the notion of compatibility of states the following definition applicable to all considered kinds of automata:

Definition 4:
Two states A_i and A_j of an automaton are compatible ($A_i \simeq A_j$), if one and the same secondary combination can be assigned to them without, however, an outside change of the functional behaviour of the automaton with reference to sequences of input and output combinations belonging to one another. Otherwise the states A_i and A_j are incompatible ($A_i \not\simeq A_j$).

Starting from this definition the conditions for incompatibility of two states can be derived for any type of automata by taking into consideration the transition and output functions and, if necessary, the stability functions. For the kinds of automata described in paragraph 2 these conditions are put together in table 1. As to the manner of notation we refer to fig. 5. In special papers[12,13] it was shown for several kinds of automata, how these conditions can be derived by considering the transition functions, the output functions and the stability functions.

4. Unconditionally and conditionally incompatible states

If the conditions 1,2 or 4 (look at table 1) are satisfied it results that two states A_i and A_j are "unconditionally" incompatible, that means $A_i \not\simeq A_j$. On the other hand by means of the conditions 3,5 and 6 one only obtains the statement that two states are "conditionally" incompatible, that means that they are incompatible if two other states A_u and A_v are incompatible.

For this conditional relation let us introduce the following notation:

$$A_i \not\equiv A_j \;/\; A_u \not\equiv A_v$$

In the following the pair of states $\{A_i, A_j\}$ is called starting pair S_B and the pair $\{A_u, A_v\}$ is named sequential pair F_B.

If the sequential pair $\{A_u, A_v\}$ of two conditionally incompatible states A_i and A_j is contained in the set $\mathcal{V}$ of the pairs of unconditionally incompatible states it follows:

$$(A_i \not\equiv A_j / A_u \not\equiv A_v) \wedge (A_u \not\equiv A_v) \Rightarrow (A_i \not\equiv A_j) \qquad (15)$$

Then the pair $\{A_i, A_j\}$ must be added to the set $\mathcal{V}$ of the pairs of unconditionally incompatible states. In this manner it is possible by iteration to determine completely the set $\mathcal{V}_{max}$ of all pairs of unconditionally incompatible states of an automaton with a given basic structure. Thus the set $\mathcal{L}$ of the pairs of conditionally incompatible states which one get by reason of the conditions 3,5 or 6 will be reduced to $\mathcal{L}_{min}$. $\mathcal{L}_{min}$ contains only the pairs of conditionally incompatible states which cannot be identified as pairs of unconditionally incompatible states. In paragraph 6 it is shown that the set $\mathcal{L}_{min}$ that is not further taken into consideration in other methods has an important part here in determining the minimal collections of compatibility classes.

If in a static operational mode by means of condition 5 for two states A_i and A_j the conditional relation $A_i \not\equiv A_j / A_i \not\equiv A_{qj}$ is obtained, there simultaneously exists the conditional relation $A_i \not\equiv A_j / A_j \not\equiv A_{qj}$ by reason of the condition 6 and vice versa (compare table 1). In the algorithms described in par. 5 to find the incompatible states of statically working automata the following rule is used, according to which it is possible to construct one of the two conditional relations from the other one.

Rule 1:

If $A_i \not\subset A_j / A_i \not\subset A_{qj}$ is a conditional relation, $(A_j \neq A_{qj})$, resulting from condition 5 (compare table 1), then $\{A_j, A_{qj}\}$ is the sequential pair of another conditional relation $A_i \not\subset A_j / A_j \not\subset A_{qj}$ about the conditional incompatibility of the states A_i and A_j, as obtained by reason of condition 6.

Example: If for instance one obtains the conditional relation (1,3)/(3,6) by reason of condition 5 one can construct from it the conditional relation (1,3)/(1,6), which one also would obtain by using the condition 6.

5. Algorithms to find the incompatible states

5.1 Formation of transition tables

According to the given conditions of incompatibility (table 1) the incompatible states of any automata can be determined from corresponding transition tables. As the conditions of incompatibility are different for the different kinds of automata it is recommendable to use differently arranged tables in the different cases. Table 2 shows an example for a transition table of type I, as it is in general use for synchronous automata[2].

$A_i(X^i/Y^i)$	00	0L	LL	L0
1 (00/L)				6/0
2 (0L/L)	1/L		5/0	
3 (L0/0)	9/0			
4 (0L/L)	1/L			
5 (LL/0)				7/L
6 (L0/0)	1/L		10/L	
7 (L0/L)	9/0		8/L	
8 (LL/L)		4/L		3/0
9 (00/0)		2/L		3/0
10 (LL/L)		2/L		

table 2

A_i	00	0L	LL	L0
1	1/L			6/0
2	1/L	2/L	5/0	
3	9/0			3/0
4	1/L	4/L		
5			5/0	7/L
6	1/L		10/L	6/0
7	9/0		8/L	7/L
8		4/L	8/L	3/0
9	9/0	2/L		3/0
10		2/L	10/L	

table 3

For minimizing the number of states of certain automata (for example statically α-unclocked automata) it is advantageous to enter according to the stability function (equ. 6) also the "starting" states (from the left-side part of the table) and the corresponding output combinations into the columns of the right-side part of the transition table marked by the input combinations belonging to those states. Table 3 shows an example for such a transition table of type II.

In the case of statically ß-unclocked automata the incompatibility of two states A_i and A_j also depends on the input combinations X^{vi} and X^{vj}, respectively, which directly preceded the input combinations X^i and X^j, respectively. To find out the incompatible states advantageously also in this case in a previous paper[13] so called transition tables of type III were suggested. Here these tables cannot be considered more in detail.

5.2 Algorithm for dynamically clocked AX-automata

Step 1: Formation of a transition table of type I (for example table 2. In this case the entry of X^i and Y^i in the left-side part of the table is not necessary).

Step 2: Comparison of the entries within the columns of the right-side part of the table.
Two states A_i and A_j are incompatible, if in the rows belonging to A_i and A_j, respectively, within the same columns
a) different output combinations are contained (condition 2) or
b) sequential states are entered which are incompatible themselves (condition 3).

Step 3: Interpreting the relation of the conditional incompatibility with taking into consideration the equ. (15).

This algorithm agrees with the algorithm for reduction of

synchronous automata which was given by Paull and Unger[2]. For the example chosen (table 2) the following pairs of incompatible states are obtained (1,5), (1,8), (1,9), (2,3), (2,6), (2,7), (3,4), (3,6), (4,7), (5,8), (5,9), (6,7), (8,9)/(2,4), (8,10)/(2,4).

5.3 Algorithm for statically α-unclocked AX-automata

Step 1: Formation of a transition table of type II (table 3).

Step 2: Comparison of the entries within the columns of the right-side part of the transition table. Two states A_i and A_j are incompatible, if in the rows belonging to A_i and A_j, respectively, within the same columns

a) the output combinations are different (condition 1,2 or 4) or

b) states are entered which are incompatible themselves (condition 3 or 5).

Step 3: Determination of the states being incompatible by reason of condition 6 by means of rule 1.

Step 4: Interpreting the relation of the conditional incompatibility with taking into consideration the equ. (15).

For the chosen example (table 3) there results:
(1,3), (1,5), (1,7), (1,8), (1,9), (2,3), (2,6), (2,7), (2,8), (2,9), (2,10), (3,4), (3,6), (3,7), (4,7), (4,9), (4,10), (5,6), (5,7), (5,8), (5,9), (5,10), (6,7), (6,8), (6,9), (7,8), (7,9), (7,10), (8,9)/(2,4), (8,10)/(2,4).

For the kind of automata considered here E.J.McCluskey[7] has suggested an algorithm in which, however, step 3 is not performed, so that one does not obtain in every case all the pairs of incompatible states. (In the considered example one would not obtain the pairs (4,10) and (7,10)).

5.4 Algorithm for dynamically ß-unclocked AX-automata

Step 1: Formation of a transition table of type I (table 2).

Step 2: Comparison of the entries in the left part of the transition table.
Two states A_i and A_j are incompatible, if for equal input combinations X^i and X^j in the left part of the table
a) the output combinations Y^i and Y^j are unequal (condition 1) or
b) in the rows belonging to A_i and A_j within the same columns of the right part of the table different output combinations are contained (condition 2) or
c) the sequential states entered within the same columns in the rows belonging to A_i and A_j are incompatible (condition 3).

Step 3: Interpreting the relation of the conditional incompatibility with taking into consideration the equ. (15).

For the example chosen one then obtains the following pairs of incompatible states: (1,9), (3,6), (3,7), (5,8), (5,10), (6,7), (8,10)/(2,4).

By using the transition tables of type III one can also formulate a corresponding algorithm for statically ß-unclocked automata[13].

6. Determining the minimal collections of compatibility classes

With taking into account the pairs of incompatible states the set $\mathcal{A}$ of all the states of an automaton can be divided into maximal compatibility classes for the forming of which different methods were suggested by Paull and Unger[2].

For determining the solutions with a minimal number of states generally not all the maximal compatibility classes are needed. Selected subsets of some of the existing maximal compatibility classes mostly are sufficient, which then form a minimal collection of compatibility classes.

Definition 5:
A collection of compatibility classes is designated as minimal collection of compatibility classes,
1. if it covers the set $\mathfrak{A}$ of all automata states,
2. if it is closed in itself and
3. if it contains a minimal number of compatibility classes.

By E.I.McCluskey[7] it was shown how to determine for a restricted class of asynchronous automata the minimal collection of (maximal) compatibility classes by means of a special table. Other methods for synchronous automata were described by Bottke[4] and by Grasell and Luccio[3]. In order to determine the minimal collections of compatibility classes for any kinds of automata the notion for the closeness of a collection of compatibility classes must be defined in a more general way than it was done especially for synchronous automata[2]. In this connection here the following theorem[12] shall be given without proof:

Theorem 1:
A collection of compatibility classes is closed if and only if within this collection for each compatibility class containing the starting pair S_B of a conditional relation $B \in \mathfrak{B}^*_{min}$ with regard to the conditional incompatibility of two states there exists a compatibility class in which the corresponding sequential pair F_B ist contained.

Starting from the definition 5 the following selection expression for determining all minimal collections of compatibility classes can then be formulated by means of theorem 1, whereby closeness is taken into consideration by the first conjunction and the requirement for covering all states by the second one:

$$H = \bigwedge_{B \in \mathfrak{B}^*_{min}} \Big(\bigvee_{Q \in \mathfrak{Q}_B} a_Q \rightarrow \bigvee_{R \in \mathfrak{R}_B} a_R \Big) \wedge \bigwedge_{A \in \mathfrak{A}} \bigvee_{N \in \mathfrak{N}_A} a_N \qquad (16)$$

where the individual symbols have the following meaning:
$\mathfrak{A}$ - set of all states A
$\mathfrak{V}$ - system of all maximal compatibility classes V_{max}

$\mathfrak{V}$ - system of all nonempty sets V, which are partial sets of an element of $\tilde{\mathfrak{V}}$ (system of all compatibility classes)

$\mathfrak{F}^*_{min}$ - set of the conditional relations $B = A_i \not\simeq A_j / A_u \not\simeq A_v$ with $A_u \simeq A_v$

$\mathfrak{y}_B$ - system of all sets $V \in \mathfrak{V}$ which contain $S_B = \{A_i, A_j\}$

$\mathfrak{x}_B$ - system of all sets $V \in \mathfrak{V}$ which contain $F_B = \{A_u, A_v\}$

$\mathfrak{n}_A$ - system of all sets $V \in \mathfrak{V}$ which contain the state A

a - binary variable which expresses whether a set $V \in \mathfrak{V}$ belongs to a collection of compatibility classes or not

For further interpretation the selection expression must be transformed into a disjunctive normal form:

$$H = \bigvee_{C \in \mathfrak{L}} \bigwedge_{w \in \mathfrak{w}_C} a_w \qquad (17)$$

where:

$\mathfrak{L}$ - set of all closed and covering collections of compatibility classes

$\mathfrak{w}_C$ - set of compatibility classes which are contained in a closed and covering collection

All conjunctions $\bigwedge_{w \in \mathfrak{w}_C} a_w$ containing a minimal number of unnegated variables a represent the minimal collections of compatibility classes. Thus, by use of the selection expression according to equ. (16) it is possible to determine systematically and in a closed form all minimal collections of compatibility classes for any automata.

The transformation of the selection expression according to equ. (16) into a disjunctive normal form according to equ. (17) without the use of a computer is very complicated and takes much time, as the system of all compatibility classes in general is rather large already for a small number of states. For this reason still another method shall be given which allows the selection problem to be solved more economically in an iterative way. To this aim there are first selected by use of a selection expression H^* from the system of all maximal compatibility classes those that as a

collection are just sufficient for covering all states of an automaton:

$$H^* = \bigwedge_{A \in \mathfrak{A}} \bigvee_{T \in \mathcal{T}_A} a_T \qquad (18)$$

where:

$\mathcal{T}_A$ - system of all maximal compatibility classes $V_{max} \in \tilde{\mathfrak{V}}$ that contain state A

After transforming the equ. (18) into a disjunctive normal form the minterms with a minimal number of variables represent those collections of maximal compatibility classes that just cover all states of the automaton. By leaving out the states occuring more than once one obtains further covering collections of compatibility classes.

After that in consideration of set $\mathfrak{L}_{min}$ it is to be examined wether the collections are closed according to theorem 1 and consequently represent minimal collections of compatibility classes according to definition 5. If the requirement of closeness cannot be fulfilled for any of these collections one has to proceed in the same manner with such minterms of the disjunctive normal form as contain one variable more. In this way one can quickly determine the minimal collections in most of the practically existing cases.

For the example of a statically α-unclocked AX-automaton mentioned in paragraph 5.3 (table 3) the following system of maximal compatibility classes is obtained in consideration of the unconditional incompatible states

$V_{max\,1} = \{1\ 2\ \ 4\ \ \ \ \ \ \ \ \}$
$V_{max\,2} = \{1\ \ \ 4\ \ 6\ \ \ \ \ \}$
$V_{max\,3} = \{1\ \ \ \ \ 6\ \ \ \ 10\}$
$V_{max\,4} = \{\ 2\ \ 4\ 5\ \ \ \ \ \ \}$
$V_{max\,5} = \{\ \ \ 3\ \ \ \ \ 8\ 9\ 10\}$
$V_{max\,6} = \{\ \ \ 4\ \ \ \ 8\ \ \ \ \}$
$V_{max\,7} = \{\ \ \ \ \ \ 7\ \ \ \ \ \}$

By using the selection expression H^* according to equ. (18) one obtains:

$$H^* = (a_1 \vee a_2 \vee a_3)(a_1 \vee a_4)a_5(a_1 \vee a_2 \vee a_4 \vee a_6)a_4(a_2 \vee a_3)a_7 \wedge$$
$$\wedge(a_5 \vee a_6)a_5(a_3 \vee a_5)$$
$$H^* = a_2a_4a_5a_7 \vee a_3a_4a_5a_7$$

The terms $a_2a_4a_5a_7$ and $a_3a_4a_5a_7$ represent those collections of maximal compatibility classes, which just cover all states of the automaton. Therefrom result the following minimal collections of compatibility classes:

a) {1 4 6 }
{ 2 4 5 }
{ 3 8 9 10}
{ 7 }

b) {1 6 10 }
{ 2 4 5 }
{ 3 8 9 10}
{ 7 }

c) {1 6 }
{ 2 4 5 }
{ 3 8 9 10 }
{ 7 }

d) {1 6 10 }
{ 2 4 5 }
{ 3 8 9 }
{ 7 }

The collection

{1 4 6 }
{ 2 5 }
{ 3 8 9 10}
{ 7 }

is not closed because of (8,9)/(2,4) and (8,10)/(2,4) and therefore it is no solution of the given problem.

7. Concluding remarks

In this paper different kinds of synchronous and asynchronous automata are considered from a unified point of view. Thereby conditions are indicated under which in each case two of their states are incompatible. All minimal collections of compatibility classes can then be determined in a systematic way by use of a

selection expression and in consideration of the incompatible states. The given method can be used for any kinds of automata, if their transition-, output- and stability-functions are known.

By interpreting the conditions for incompatibility it is furthermore possible to compare the different kinds of synchronous and asynchronous automata in a general form with regard to the minimal number of states and memory elements, respectively, necessary for solving a given problem.

References

/1/ Gluschkow, W.M.
Theorie der abstrakten Automaten
(Übers. aus dem Russischen von G.Asser)
VEB Deutscher Verlag d.Wiss., Berlin 1963

/2/ Paull, M.C., S.H.Unger
Minimizing the Number of States in Incompletely Specified Sequential Switching Functions
IRE Trans. on Electr.Comp.EC-8 (1959) Nr.3, S.356-367

/3/ Grasselli,A., F.Luccio
A Method for Minimizing the Number of Internal States in Incompletely Specified Sequential Networks
IEEE Trans. on Electr.Comp. 14 (1965) №.3, S.350-359

/4/ Bottke, G.
Zur Reduktion partieller Automaten
Dissertation Humboldt-Universität Berlin, 1967

/5/ Поттосин, Ю.В.
Сравнительная оценка двух алгоритмов минимизации числа состояний дискретного автомата
Авт. и вычислительная техника,1967,№4,стр.22-28

/6/ Huffman, D.A.
The Synthesis of Sequential Switching Circuits
Journ. of the Franklin Inst. 257 (1954) Nr.3, S.161-190

/7/ McCluskey, E.J.
Minimum-State Sequential Circuits for a Restricted Class of Incompletely Specified Flow Tables
The Bell Syst. Technical Journ. 41 (1962) Nr.6, S.1759-1768

/8/ Лазарев,В.Г., Пийль, Е.И.
Синтез асинхронных конечных автоматов
Изд."Наука",Москва 1964

/9/ Якувайтис, Э.А.
Асинхронные логические автоматы
Изд. "Зинатне",Рига 1966

/10/ Таль,А.А.
Построение статических асинхронных последовательностных машин из типовых ячеек
Авт.и телем.,27/1966/ №12, стр.94-114

/11/ Paull, M.C., G.Waldbaum
A Note on State Minimization of Asynchronous Sequential Functions
IEEE Trans. on Electr.Comp. EC-16 (1967) Nr.1, S.94-96

/12/ Zander, H.J.
Zur Zustandsreduktion ungetakteter (asynchroner) Folgeschaltungen
Elektron.Informationsverarbtg.u.Kybernetik (EIK) 4 (1968) Nr. 4, S.257-278 u. Nr.5, S.285-300

/13/ Zander, H.J.
Besonderheiten bei der Zustandsminimierung ungetakteter (asynchroner) Folgeschaltungen mit Differenziergliedern
messen steuern regeln (msr) 11 (1968) Nr.3, S.91-98

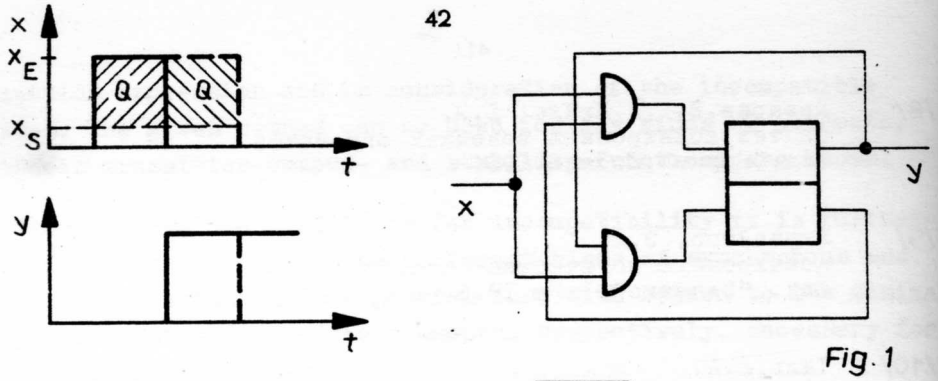


Fig. 1

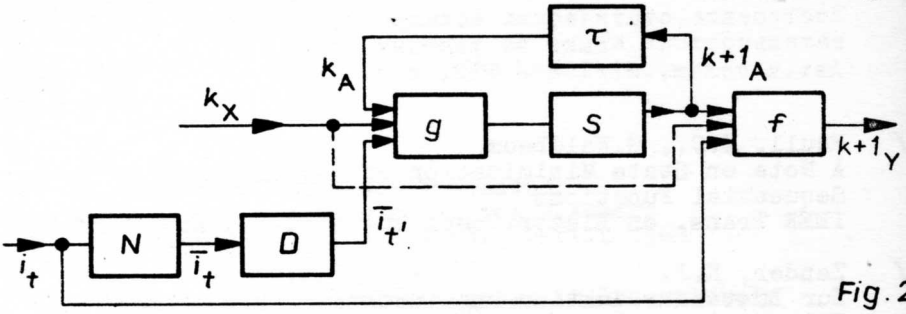


Fig. 2

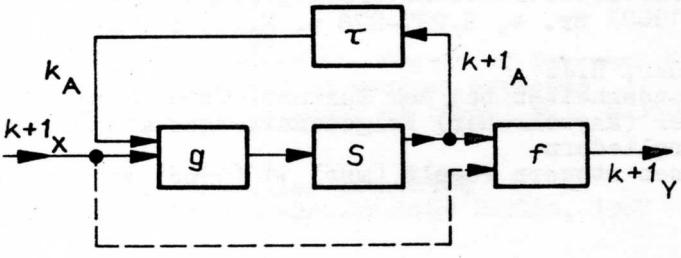


Fig. 3

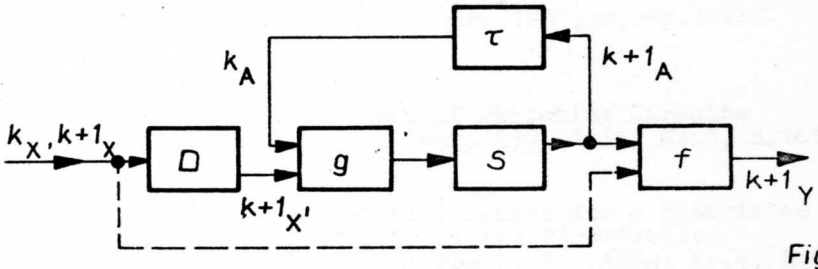


Fig. 4

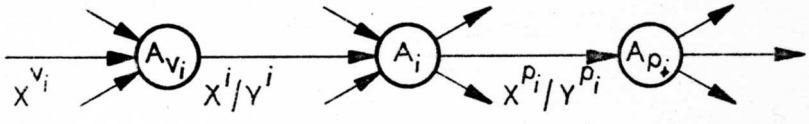


Fig. 5

		Dynamically clocked AX-automata	Statically α-unclocked AX-automata	Dynamically β-unclocked AX-automata	Statically β-unclocked AX-automata
Transitions function		$A_{p_i} = g(A_i, X^{p_i})$	$A_{p_i} = g(A_i, X^{p_i})$	$A_{p_i} = g(A_i, X^{p_i'})$	$A_{p_i} = g(A_i, X^{p_i'})$
Output function		$Y^{p_i} = f_2(A_i, X^{p_i})$	$Y^{p_i} = f_2(A_{p_i}, X^{p_i})$	$Y^{p_i} = f_2(A_{p_i}, X^{p_i})$	$Y^{p_i} = f_2(A_{p_i}, X^{p_i})$
Stability function		—	$A_{p_i} = s(A_{p_i}, X^{p_i})$	—	$A_{p_i} = s(A_{p_i}, X^{p_i'})$
$A_i \neq A_j$, if...	1	—	$X^i = X^j \wedge Y^i \neq Y^j$	$X^i = X^j \wedge Y^i \neq Y^j$	$X^i = X^j \wedge Y^i \neq Y^j$
	2	$X^{p_i} = X^{q_j} \wedge Y^{p_i} \neq Y^{q_j}$	$X^{p_i} = X^{q_j} \wedge Y^{p_i} \neq Y^{q_j}$	$X^i = X^j \wedge X^{p_i} = X^{q_j} \wedge Y^{p_i} \neq Y^{q_j}$	$X^i = X^j \wedge X^{p_i} = X^{q_j} \wedge Y^{p_i} \neq Y^{q_j}$
	3	$X^{p_i} = X^{q_j} \wedge A_{p_i} \not\simeq A_{p_j}$	$X^{p_i} = X^{q_j} \wedge A_{p_i} \not\simeq A_{q_j}$	$X^i = X^j \wedge X^{p_i} = X^{q_j} \wedge A_{p_i} \not\simeq A_{q_j}$	$X^i = X^j \wedge X^{p_i} = X^{q_j} \wedge A_{p_i} \not\simeq A_{q_j}$
	4	—	$X^i = X^{q_j} \wedge Y^i \neq Y^{q_j}$	—	$X^{v_i} = X^j \wedge X^i = X^{q_j} \wedge Y^i \neq Y^{q_j}$
	5	—	$X^i = X^{q_j} \wedge A_i \not\simeq A_{q_j}$	—	$X^{v_i} = X^j \wedge X^i = X^{q_j} \wedge A_i \not\simeq A_{q_j}$
	6	—	$X^i = X^{q_j} \wedge A_j \simeq A_{q_j}$	—	$X^{v_i} = X^j \wedge X^i = X^{q_j} \wedge A_j \not\simeq A_{q_j}$
Conditions for incompatibility of two states					

Table 1

АСИНХРОННАЯ МОДЕЛЬ КОНЕЧНОГО АВТОМАТА

Э.А.Якубайтис, доктор технических наук, профессор
г.Рига, СССР

Введём определения. Назовём правильной систему уравнений, реализация которой даёт автомат, свободный от всех видов опасных состязаний. Код устойчивых внутренних состояний, устраняющий в автомате опасные состязания между промежуточными сигналами, также, назовём правильным. Описывающую автомат систему уравнений, в которой без нарушения её правильности нельзя сократить число операций (дизъюнкций, конъюнкций, отрицаний), будем называть безызбыточной. Поставим задачу нахождения на основе модели автомата, описываемой ниже, всех безызбыточных правильных систем уравнений, обеспечивающих максимальное быстродействие автомата. При введении критерия выбора, из полученного множества систем можно отобрать оптимальные. Наиболее легко задача решается при минимизации памяти автомата.

Представим автомат в виде k инерционных и m примитивных подавтоматов. Инерционный подавтомат q состоит (рис.I) из преобразователя Π_q, содержащего логические элементы, и фильтра Φ_q, осуществляющего фильтрацию импульсов вида $0 \to 1 \to 0$ и $1 \to 0 \to 1$, если их продолжительность менее $\tau_q(t)$.

Как следует из рис.I, инерционный подавтомат q имеет один контур обратной связи, в который входит конечное число логических элементов и фильтр. Примитивный подавтомат v (рис.2) состоит из преобразователя Π_{k+v} и фильтра Φ_{k+v}, но в отличие от инерционного, в нём нет ни одного контура обратной связи.

Инерционные подавтоматы I, ... , k, где k - число контуров обратной связи в автомате (число "элементов па-

мяти") генерируют промежуточные сигналы $X_1, \ldots, X_k$, а примитивные подавтоматы $k+1, \ldots, k+m$ выдают выходные сигналы $Z_1, \ldots, Z_m$, где m – число этих сигналов. $A_1, \ldots, A_n$ – входные сигналы, t – непрерывное время.

Примем, что преобразователи подавтоматов построены на логических элементах, каждый (j – тый) из которых имеет внутреннюю задержку $\tau_j(t)$, являющуюся случайной функцией времени с ограничениями:

$$\tau_э \geq \tau_j(t) \geq 0, \qquad ((I))$$

где $\tau_э$ – заданная конечная величина.

Любой реальный фильтр Φ_i не только осуществляет необходимую фильтрацию заданного сигнала, но и сдвигает этот сигнал на $\tau_i(t)$. Поэтому разделим каждый из имеющихся на рис.I и 2 фильтров на идеальный фильтр ($И\Phi_i$), выполняющий только фильтрацию, и задержку $\tau_i(t)$.

Условимся, что задержка $\tau_i(t)$ является случайной функцией времени с ограничениями:

$$\tau_ф > \tau_j(t) > \tau_{пп}, \qquad ((2))$$

где $\tau_ф$ – заданная конечная величина; $\tau_{пп}$ – время самого продолжительного переходного процесса, происходящего в автомате.

В соответствии со сказанным, на основе рис.I и 2 модель автомата, состоящая из k инерционных и m примитивных подавтоматов, может быть представлена в виде, показанном на рис.3. Как следует из рис.I и 2 автомат, показанный на рис.3 имеет k контуров обратной связи, ибо функции вида $Y_a(t) = f(Y_a(t), \ldots)$, называемые кольцами, запрещены.

Примем, что в рассматриваемой модели автомата при переходе из одного устойчивого полного состояния в другое допускается одновременное измерение значений любого числа промежуточных сигналов.

Наряду с моделью, показанной на рис.3, может быть также использована модель, показанная на рис.4. В этом случае (рис.I и 2) на входы подавтоматов сигналы $Y_1(t)$, ... , $Y_k(t)$ не подаются. Однако, при переходе от модели рис.3 к модели рис.4 резко уменьшается множество безызбыточных правильных систем уравнений, описывающих заданное преобразование. Известны случаи, например, в работе[I], когда число систем уменьшается в сотни раз.

При построении автомата необходимо помнить, что любой критерий его оптимальности всегда должен учитывать затраты на процесс его синтеза. Наличие либо отсутствие необходимых программ, вычислительных машин и главное - стоимость времени проектирования, резко изменяют объём указанных затрат. В связи с этим целесообразно иметь несколько путей, требующих различного времени синтеза автомата. Некоторые из них показаны на рис.5. Они отличаются друг от друга временем синтеза и, естественно, вероятностью получения оптимального результата.

В качестве языка задания автомата удобно использовать, описанный в работе[2] язык графов конечных автоматов. Графом конечного автомата является ориентированный граф, вершины которого отождествляются с устойчивыми полными состояниями, а дуги показывают переходы между этими состояниями. Поэтому всюду ниже под вершиной понимается устойчивое полное состояние. Для нахождения систем уравнений, описывающих автомат, граф целесообразно представлять, как в работе[3], в виде матрицы конечного автомата. Связь таблицы переходов с матрицей конечного автомата проста. Число, записанное в левой части клетки строки β таблицы определяет номер столбца матрицы, в котором в строке β имеется единица. Столбец ρ определяет входные, а λ - выходные состояния в вершинах.

Матрица конечного автомата удобна тем, что её столбец q определяет значения промежуточного сигнала X_q во всех вершинах. Каждая строка матрицы задаёт правильный код соответствующего устойчивого состояния. Например, в строке I матрицы, показанной в таблице I, записано число IOOOOIII, кодирующее вершину I.

Таблица I

	00	0I	I0	II
I	I-00	6-II	7-00	8-0I
2	2-0I	5-00	7-00	8-0I
3	3-I0	6-II	7-00	8-0I
4	4-II	6-II	7-00	8-0I
5	I-00	5-00	7-00	8-0I
6	2-0I	6-II	7-00	8-0I
7	3-I0	5-00	7-00	8-0I
8	4-II	5-00	7-00	8-0I

Таблица переходов

	I	2	3	4	5	6	7	8	ρ	λ
I	I	0	0	0	0	I	I	I	00	00
2	0	I	0	0	I	0	I	I	00	0I
3	0	0	I	0	0	I	I	I	00	I0
4	0	0	0	I	0	I	I	I	00	II
5	I	0	0	0	I	0	I	I	0I	00
6	0	I	0	0	0	I	I	I	0I	II
7	0	0	I	0	I	0	I	I	I0	00
8	0	0	0	I	I	0	I	I	II	0I

Матрица конечного автомата

Код, задаваемый матрицей всегда избыточен и его длина, при желании, может быть сокращена.

Устранение опасных состязаний в конечном автомате выполняется благодаря введению правила инерционности: промежуточный сигнал X_q равен единице в вершине q, в предыдущих вершинах и во всех неустойчивых полных состояниях, имеющих место при непосредственных переходах в вершину q. В остальных вершинах и неустойчивых полных состояниях сигнал X_q равен нулю.

Правильный код еще не определяет правильную систему. Поэтому, учитывая то, что определение правильного кода является не самоцелью, а лишь промежуточным этапом процесса синтеза автомата, целесообразно построить процесс синтеза так, чтобы можно было находить правильные системы уравнений, минуя специальный этап кодирования. Полученные в работах[1,2,4-6] алгоритмы позволяют найти безызбыточные правильные системы уравнений, не выполняя отдельного этапа кодирования.

Нахождение системы уравнений непосредственно по матрице конечного автомата выполняется на основе работы[2] в соответствии с функциями (для упрощения записи время опускается):

$$X_q = \bigvee_{d \in M_q} R_d X_d \ , \qquad ((3))$$

$$Z_v = \bigvee_{\beta \in N_v} R_\beta X_\beta \ , \qquad ((4))$$

где R_d – элементарная конъюнкция входных сигналов $A_1, \ldots, A_n$, равная единице только при входном состоянии ρ_d; R_β – то же, при входном состоянии ρ_β; X_q – промежуточный сигнал, определяемый столбцом q матрицы; X_β – то же – столбцом β; M_q – множество строк матрицы (устойчивых полных состояний), в которых столбец q содержит единицы; N_v – множество строк матрицы, в которых

выходной сигнал $Z_v = 1$. Если столбец e матрицы не содержит ни одного нуля, то $X_e = 1$; $e = d$ либо $в$.

Уравнение вида ((3)) либо ((4)), условимся называть особой дизъюнктивной нормальной формой функции.

Например, на основе ((3)) и ((4)) из матрицы таблицы I имеем:

$$X_1 = \bar{A}_1 \bar{A}_2 X_1 + \bar{A}_1 A_2 X_5 ,$$

$$X_2 = \bar{A}_1 \bar{A}_2 X_2 + \bar{A}_1 A_2 X_6 ,$$

$$X_3 = \bar{A}_1 \bar{A}_2 X_3 + A_1 \bar{A}_2 ,$$

$$X_4 = \bar{A}_1 \bar{A}_2 X_4 + A_1 A_2 ,$$

$$X_5 = \bar{A}_1 \bar{A}_2 X_4 + \bar{A}_1 A_2 X_5 + A_1 \bar{A}_2 + A_1 A_2 , \qquad ((5))$$

$$X_6 = \bar{A}_1 \bar{A}_2 X_1 + \bar{A}_1 \bar{A}_2 X_3 + \bar{A}_1 \bar{A}_2 X_4 + \bar{A}_1 A_2 X_6 ,$$

$$Z_1 = \bar{A}_1 \bar{A}_2 X_3 + \bar{A}_1 \bar{A}_2 X_4 + \bar{A}_1 A_2 X_6 ,$$

$$Z_2 = \bar{A}_1 \bar{A}_2 X_2 + \bar{A}_1 \bar{A}_2 X_4 + \bar{A}_1 A_2 X_6 + A_1 A_2 .$$

Следует отметить, что определение указанным методом безызбыточной правильной системы уравнений позволяет сразу на основе матрицы конечного автомата получать относительно простые результаты, которые при использовании системы совершенных дизъюнктивных нормальных форм функций достижимы лишь после проведения достаточно трудоёмкого процесса минимизации этих функций.

Предельная сложность системы, полученной непосредственно из матрицы, определяется в соответствии с работой[7] цифрами:

$$D=c+m,\quad K=p,\quad O=n,\quad \Phi=c+m,$$

$$B_1=n+2m+p+2c, \qquad ((6))$$

где D , K , O , Φ – соответственно, число дизъюнкций, конъюнкций, отрицаний, фильтров; B_1 – общее число операций (число элементов, каждый из которых имеет любое число входов и любую нагрузочную способность); c – число столбцов матрицы (без учёта столбцов ρ и λ), каждый из которых имеет хотя бы один нуль; p – общее число строк матрицы.

Операция объединения столбцов подматрицы i сводится, в соответствии с работой[5], к произвольному их кодированию α_i сигналами[I)]:

$$\alpha_i=\log_2 \lfloor d_i \rfloor , \qquad ((7))$$

где d_i – число столбцов в подматрице i ; $\lfloor d_i \rfloor$ – округление d_i до ближайшего большего числа из ряда I,2,4,8, ... После объединения столбцов система уравнений, характеризующая автомат, определяется в соответствии с ((3)) и ((4)).

Например, для матрицы, показанной в таблице I, кодирование определяется таблицей 2. В этой таблице в строках столбцов, образуемых столбцами a и b , c записываются любые различные числа.

Таблица 2

	a	b
I	0	0
2	0	I
3	I	0
4	I	I

	c
5	0
6	I

I) Подматрицей называется множество столбцов матрицы, представляющих вершины с одинаковыми ρ .

Из таблицы 2 следует, что столбец a должен быть образован построчной дизъюнкций столбцов 3, 4 матрицы таблицы I; столбец b - построчной дизъюнкцией столбцов 2, 4; столбец c является столбцом 6. В результате указанных преобразований из таблицы I получаем таблицу 3.

Таблица 3

	a	b	c	ρ	λ
I	0	0	I	00	00
2	0	I	0	00	0I
3	I	0	I	00	I0
4	I	I	I	00	II
5	0	0	0	0I	00
6	0	I	I	0I	II
7	I	0	0	I0	00
8	I	I	0	II	0I

В соответствии с ((3)) и ((4)) на основе таблиц I и 3, записываем систему уравнений, получающуюся после объединения столбцов:

$$X_a = \overline{A}_1\overline{A}_2 X_a \overline{X}_b + \overline{A}_1\overline{A}_2 X_a X_b + A_1\overline{A}_2 + A_1 A_2 ,$$

$$X_b = \overline{A}_1\overline{A}_2\overline{X}_a X_b + A_1 A_2 X_a X_b + \overline{A}_1 A_2 X_c + A_1 A_2 ,$$

$$X_c = \overline{A}_1\overline{A}_2\overline{X}_a\overline{X}_b + \overline{A}_1\overline{A}_2 X_a\overline{X}_b + A_1 A_2 X_a X_b + \overline{A}_1 A_2 X_c , \qquad ((8))$$

$$Z_1 = \overline{A}_1\overline{A}_2 X_a\overline{X}_b + \overline{A}_1\overline{A}_2 X_a X_b + \overline{A}_1 A_2 X_c ,$$

$$Z_2 = \overline{A}_1\overline{A}_2\overline{X}_a X_b + \overline{A}_1\overline{A}_2 X_a X_b + \overline{A}_1 A_2 X_c + A_1 A_2 .$$

Максимальная сложность получаемой таким образом системы уравнений определяется в соответствии с работой[5] равенствами:

$$D=\sum_i \log_2 \lceil d_i L+m, \quad K=p,$$

$$O=n+\sum_i \log_2 \lceil d_i L, \quad \Phi=\sum_i \log_2 \lceil d_i L+m, \qquad ((9))$$

$$B_2=n+2m+p+3\sum_i \log_2 \lceil d_i L.$$

Автомат, в котором выполняется неравенство

$$S>\frac{\prod_i \lceil d_i L}{2}, \qquad ((10))$$

где S – число устойчивых внутренних состояний, назовём предельным. Как показано в работе[5], в предельном автомате после объединения столбцов получается система уравнений, обеспечивающая образование минимальной памяти. Далее никакими другими способами в автомате уменьшить память нельзя.

Для автоматов, не являющихся предельными, минимизация памяти выполняется в соответствии с работами[9,10] на этапе "специальной минимизации". Этот этап достаточно трудоёмок. Поэтому им целесообразно пользоваться только в том случае, когда при этом получается существенное упрощение автомата. Сложность автомата после выполнения этапа "специальной минимизации" может быть оценена выражением

$$B_3=n+2m+p+3\log_2 \frac{\lceil sL-1+\log_2 \lceil sL}{2} \qquad ((11))$$

Если $B_3 \ll B_2$, то выполнение этого этапа имеет смысл.

Сущность "специальной минимизации" заключается в необходимости обеспечения различимости вершин с одинаковыми входными состояниями. В соответствии с этим на основе матрицы конечного автомата можно найти наиболее простые элементарные конъюнкции, которые с учётом ограничений, вводимых состязаниями между фильтрами, обеспечивают указанную различимость. В результате, как показано в работе[9], получаются все безызбыточные правильные коды устойчивых внутренних состояний[I)]. При желании из этих кодов выбираются коды, имеющие минимальную длину.

Определение наиболее простых элементарных конъюнкций, представляющих вершины, позволяет в соответствии с работой[I] определить все безызбыточные правильные системы уравнений, описывающие автомат. При этом всегда

$$G_4 \subseteq G_3, \qquad ((12))$$

где G_4 – множество систем, получаемых при использовании упрощённой модели, показанной на рис.4, G_3 – то же, для общей модели рис.3.

Как показано в работе[I], процесс синтеза может быть за счёт потери части вариантов безызбыточных правильных систем значительно ускорен.

В заключение отметим основные результаты, относящиеся не только к рассматриваемой (рис.3), но и к широко исследуемой различными авторами "классической" модели конечного автомата (модели рис.4, в которой все логические элементы безынерционны):

I. Подача на входы преобразователя не только сигналов, снятых с выходов задержек, но и со входов этих задержек, может привести к значительному увеличению множества правильных систем уравнений.

I) Безызбыточным называется код, в котором ни один из разрядов не может быть опущен.

2. Для предельного конечного автомата кроме операции объединения столбцов любая минимизация памяти бессмысленна.

3. Логическое преобразование правильной системы уравнений не обязательно даёт правильную систему уравнений.

4. Не все правильные системы получаются в результате минимизации системы совершенных нормальных форм функций.

Л И Т Е Р А Т У Р А

I. Э.А.ЯКУБАЙТИС. Безызбыточные системы уравнений, описывающие асинхронный конечный автомат. Автоматика и вычислительная техника, № 6, 1968.

2. Э.А.ЯКУБАЙТИС. Асинхронный логический автомат. Автоматика и вычислительная техника, № I, 1967.

3. Э.А.ЯКУБАЙТИС. Метод кодирования внутренних состояний конечного автомата. Автоматика и вычислительная техника,№ 3, 1968.

4. Э.А.ЯКУБАЙТИС. Методика определения систем уравнений,описывающих асинхронный конечный автомат. Автоматика и вычислительная техника, № 6, 1968.

5. Э.А.ЯКУБАЙТИС. Объединение промежуточных сигналов. Автоматика и вычислительная техника, № 5, 1967.

6. Э.А.ЯКУБАЙТИС. Асинхронные логические автоматы. Изд.Зинатне, 1966.

7. А.Ю.ГОБЗНМИС. Два критерия оценки эффективности метода инерционных подавтоматов. Автоматика и вычислительная техника, № 2, 1967.

8. Э.А.ЯКУБАЙТИС. Безызбыточное кодирование асинхронного конечного автомата. Автоматика и вычислительная техника, № 4, 1968.

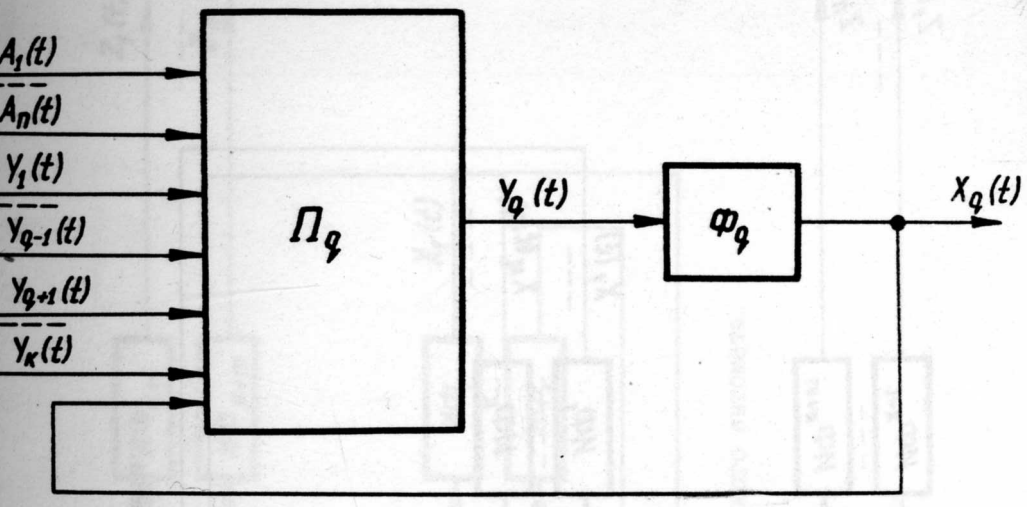


Рис.I. Инерционный подавтомат.

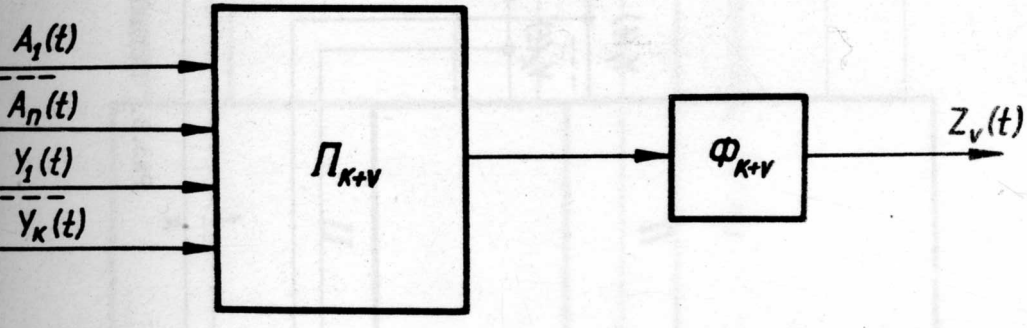


Рис.2. Примитивный подавтомат.

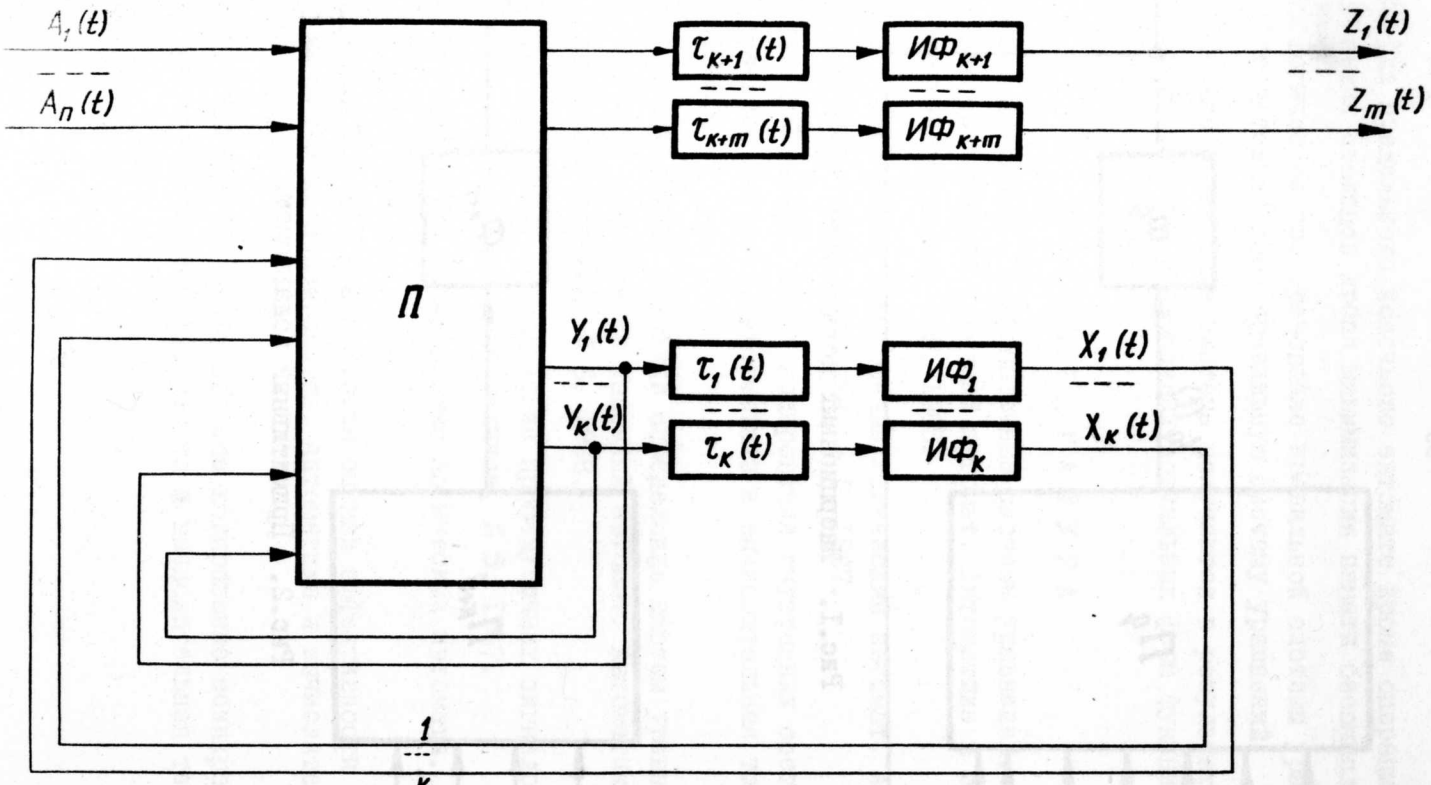


Рис.3. Модель конечного автомата.

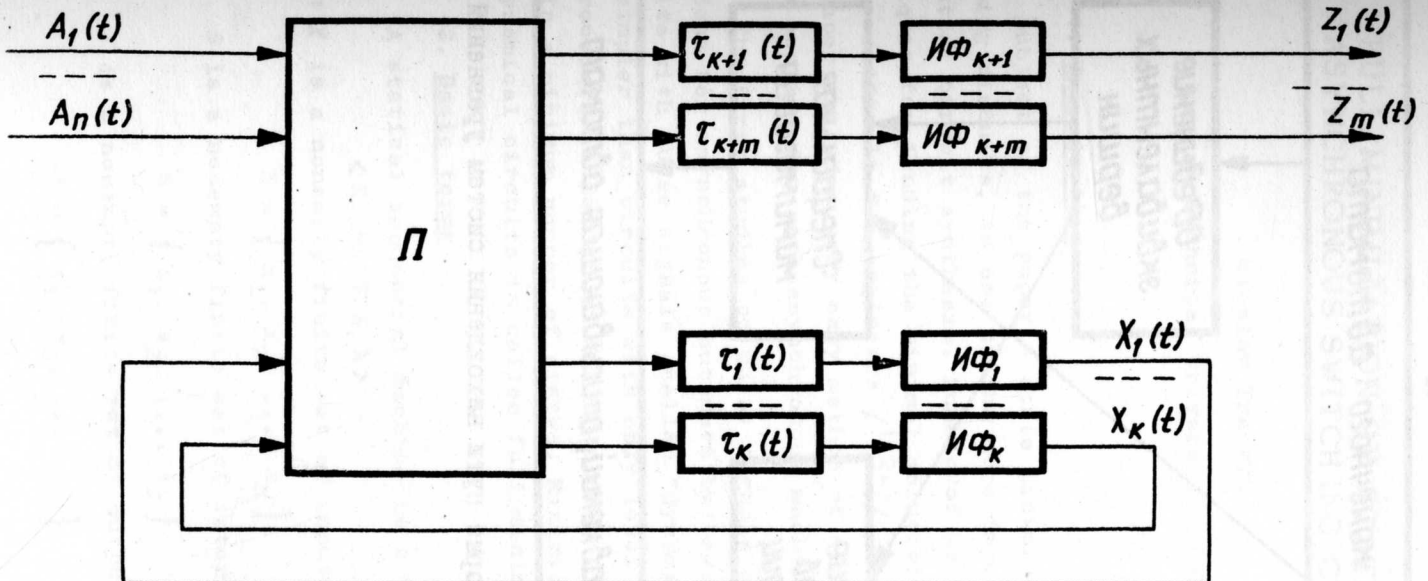


Рис.4. Упрощённая модель конечного автомата.

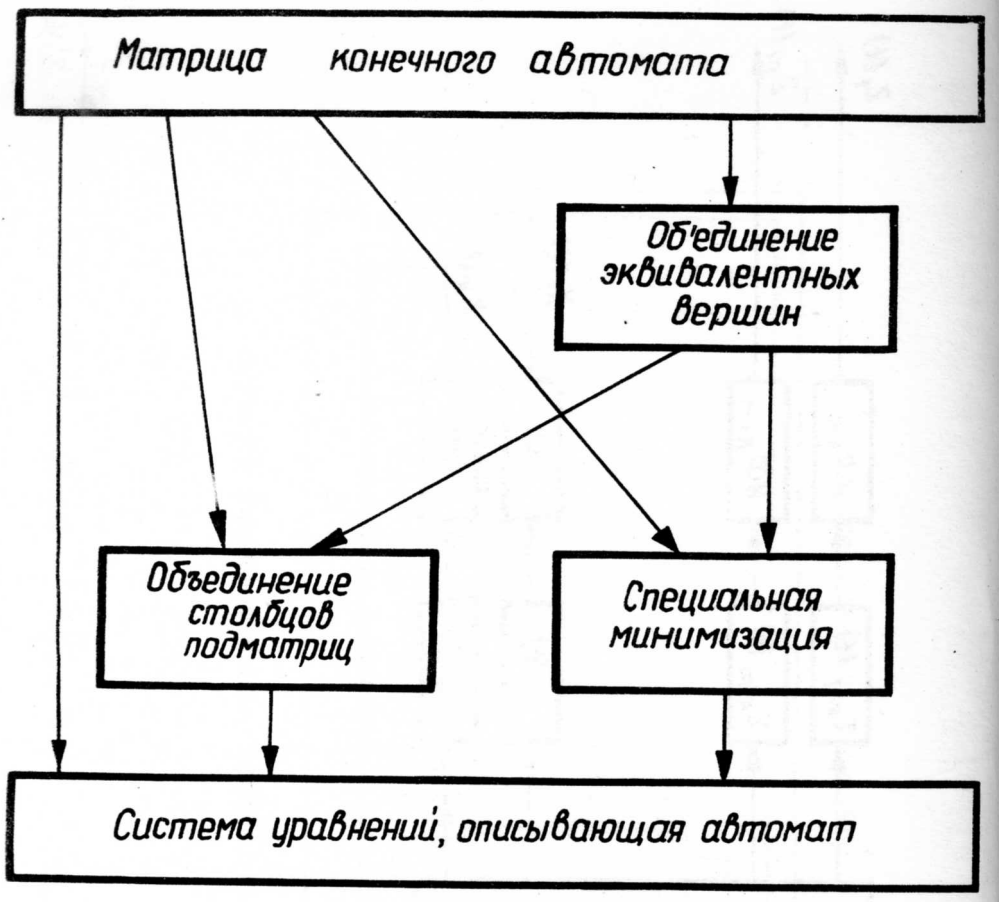


Рис.5. Некоторые пути нахождения систем уравнений.

FULL MINIMIZATION OF STATES FOR ASYNCHRONOUS SWITCHING CIRCUITS

Wiesław Traczyk

Warsaw Technical University, Poland.

1. Introduction

Subject of the paper - state minimization for asynchronous switching circuits, is one of the more important problems of sequential circuit synthesis. Number of papers discussed several techniques for finding the minimal solution but presented methods are too difficult for engineers /[1,2,3]/ or too limited[4,5]. This paper includes very easy method of simplification, which can be applied for each asynchronous machine.

The first studies on pulse signals and their exploitation were done for asynchronous automata by Lazariew and Pijl[3]. Circuits with pulse signals /called "dynamical"/ can be made much simpler than circuits with only level signals /statical/. The procedure is described for obtaining dynamical circuits which contain a minimum number of states. Minimization of both: statical and dynamical circuits is called full minimization of states.

2. Basic terms

A statical sequential machine is a 5-tuple

$$\langle X, S, Y, \delta, \lambda \rangle$$

where: X is a nonempty finite set of inputs x :

$$X = \{ x_1, x_2, \ldots x_N \}$$

S is a nonempty finite set of internal stetes s:

$$S = \{ s_1, s_2, \ldots s_K \}$$

Y is a nonempty finite set of outputs y:

$$Y = \{ y_1, y_2, \ldots y_M \}$$

$\delta : X \times S \rightarrow S$ is next-state function

$\lambda : X \times S \rightarrow Y$ is output function

If function δ or λ is a mapping from a subset of XxS into S or Y than the machine is incomplete.

A dynamical sequential machine is a 5-tuple

$$\langle X, S, Y, \alpha, \beta \rangle$$

where X, S, Y, are defined as before,

$\alpha : X \times X \times S \rightarrow S$ is next-state function

$\beta : X \times X \times S \rightarrow Y$ is output function

A pair /x,s/ determines a total state of the machine. Time interval, in which the corresponding total state does not change is numbered: 1,2,....,t,.... A pair $\frac{x^{t-1}}{x^t}$ determines a dynamical input.

A statical machine can be described by equations:

$$s^{t+1} = \delta\left[s^t, x^t\right] \qquad /1/$$

$$y^t = \lambda\left[s^t, x^t\right] \qquad /2/$$

and dynamical machine - by equations:

$$s^{t+1} = \alpha\left[s^t, x^t, \frac{x^{t-1}}{x^t}\right] \qquad /3/$$

$$y^t = \beta\left[s^t, x^t, \frac{x^{t-1}}{x^t}\right] \qquad /4/$$

Dynamical input is feeled by the elements /for example - differential/ which generete pulse signals. Since in practical cases the output signals are, usually, level, dynamical inputs in /4/ are needless. Than from /4/ we obtain equation /2/ and therefore in this paper we discuss dynamical machines described by equations /3/ and /2/. If change of some internal

state is not dependent on the dynamical input /in dynamical machine/, than for this state equation /3/ passes into 1/. If change of some state s is dependent on the dynamical input, equation /3/ for this state passes into /5/:

$$s^{t+1} = \alpha' \left[s^t, \frac{x^{t-1}}{x^t} \right] \qquad /5/$$

All statical states /x,s,y/ of the machine are defined by the sequence of two-valuable signals; the dynamical state is defined by two such sequences.

Let a basic form of a presentation of the machine will be flow table, which includes /in general case/ "don't care" conditions. As a undefined state /or signal/ we understand an abstractive state /signal/ which has arbitrary value or never exists. Such a state is denoted by /-/.

A pair of internal states /s,s'/ is called incontradictory if both sand s' are the same or one of them is undefined. A mark: $s \overset{n}{=} s'$. For example:

$$/-/ \overset{n}{=} s, \quad s \overset{n}{=} s$$

A pair of outputs /y,y'/ is called incontradictory /$y \overset{n}{=} y'$/ if every pair of correspondent signals from y and y' includes the same elements or one of them is undefined. For example: /0-1/ $\overset{n}{=}$ /-11/

If $P = x^1, x^2, \ldots, x^p$ is a sequence of inputs, than $\delta\left[s^1, P\right]$ denotes a set of all internal states, which can be obtained by the action of the sequence P from the state s^1:

$$\delta/s^1, P/ = \left\{ s^{i+1} \mid s^{i+1} = \delta\left[x^i, s^i\right] , \; i = 1,2,\ldots,p \right\}$$

A sequence P is applicable to a state s if all elements of the set δ /S,P/ are defined.

If for every sequence P which is applicable to both s and s' holds:

$$\lambda/s,P/ \stackrel{n}{=} \lambda/s', P/$$

than the states s and s' are compatible. We denote such states by $s \sim s'$

The states which are mutually compatible give a compatible set. Every such a set can be changed by one state and therefore number of internal states of the machine can be reduced to minimal number of compatible sets. It is relatively easy to obtain a class Γ_{max} /the final class/ of all maximum compatible sets C:

$$\Gamma_{max} = \{C_1, C_2, \ldots, C_c\}$$

That reduces number of internal states to value c. However, for incompletely specified machines /with undefined states/ the sets C can intersect and therefore some of the sets can be removed or reduced. The removal of states is allowed, when the class of sets remains closed and completed.

A class $\Gamma = \{C_1, C_2, \ldots, C_c\}$ of sets C of internal states s is closed if, for every pair /s,s'/ from C_i /$C_i \in \Gamma$/ and every input x /$x \in X$/ for which $\delta/s,x/$ and $\delta/s',s/$ are defined, is

$$(\delta/s,x/, \delta/s', x/) \subseteq C_j \qquad /C_j \in \Gamma/$$

A class Γ is a complete class if

$$S \subseteq C_1 \cup C_2 \cup \ldots \cup C_c$$

A closed, complete class of compatible sets with minimal number of sets /f/ is called a minimal class:

$$\Phi_{min} = \{F_1, F_2, \ldots, F_f\}$$

A minimal class with minimal number of elements in the sets is called a optimal class:

$$\Phi_{opt} = \{F_1', F_2', \ldots F_f'\}$$

Therefore the minimization problem of number of internal states one can consider as a problem of investigation of classes Φ_{min} and Φ_{opt} .

3. Statical machines

The practical investigation of compatible states is based on the theorem, which say that both the states: s and s′ are compatible if:

1. for every $x_i \in X$ a pair $(\delta/s,x_i/ \ , \ \delta/s',x_i/)$ is incontradictory or compatible, and
2. $\lambda/s,x_i/ \stackrel{n}{=} \lambda/s',x_i/$

These conditions one can verify in every column of the flow table and the verification is relatively easy, but condition 1 can do difficulties when the conditional compatibility holds. The compatibility of one pair is then dependent on compatibility of another pair and reduction of states ought to be preceded by examination of close of the classes.

Engineers' methods recommended[6] appliance of the pseudoequivalent states, that means such compatible states which are stable at the same x. However, (as can be easy seen, application of such) states not always gives good results. In this paper the pseudoequivalent states will be used for optimal state minimization.

Theorem 1. Let $\Psi_{max} = \{H_1,H_2,\ldots,H_h\}$ be a class of all maximal sets of pseudoequivalent states, and let $\Gamma_{max} = \{C_1,C_2,\ldots,C_c\}$ be a final class. For each set H_i if $H_i \subseteq C_j$ and if exists such a state s /$s \in H_i$/ which is not an element of another C, then a class Φ_{min} arises from Ψ_{max}, by enlargement Ψ_{max} till a complete class with maximal application of compatibility.

Proof. A set H_i can belong to none of a set F from Φ_{min} if all elements of H_i are in Φ_{min} by means of another H. However, according to the condition of theorem, it is impossible. All elements of H_i are only in C_j, then C_j is some set of Φ_{min} /for example - F_l/ and $H_i \subseteq F_1$. If all sets of pseudo-equivalent states are included in minimal solution, then all conditions of conditional compativility are executed. The remaining states of the machine one can reduce, without additional conditions. The enlargement of the class Ψ_{max}, in this way, till complete class, gives a class with minimal number of states. This completes the proof.

From theorem 1 arises a procedure of minimization. Then one ought to obtain /by discretionary method/ the classes Ψ_{max} and Γ_{max} and to verify a condition of theorem 1. If it is satisfied, we treat the sets of class Ψ_{max} as a single states, and apply the well known merger diagram method[6]. If discused condition is not satisfied, we must examine whether resignation with some set H gives better results. To illustrate this method, consider the examples.

Primitive flow table[3] is presented on figure 1a /instead of is used i/. It is easy to obtain a class:

$$\Psi_{max} = \{1,2;\ 1,3;\ 4,5;\ 6,7\}$$

The pairs 2,3; 8,9; 10,11 are openly incompatible, therefore[2]:

$$\Gamma_{max} = \{1,2,6,7,8;\ 1,3,7,8;\ 4,5,10;\ 5,9,10;\ 6,7,11\}$$

A condition from theorem 1 is satisfied, the merger diagram can be used /figure 1b/. From the diagram we obtain four minimal classes. One of them is

$$\Phi_{min1} = \{1,2,8;\ 1,3;\ 4,5,10;\ 6,7,9,11\}$$

A state 1 is an element of two sets, but can not be removed, then $\Phi_{opt1} = \Phi_{min1}$. The final table of the statical machine is presented on the figure 1c.

By the same manner, from the table of figure 2a, we obtain:

$$\Psi_{max} = \{2,3\}$$

$$\Gamma_{max} = \{1,2;\ 3,4;\ 3,5,7;\ 2,6,8;\ 2,3,6;\ 2,3,5\}$$

The both states 2 and 3 are elements of different sets C, so the theorem 1 is not satisfied. After breaking of the set $\{2,3\}$, the merger diagram /figure 2b/ gives

$$\Phi_{min} = \Phi_{opt} = \{1,2;\ 3,4;\ 5,7;\ 6,8\}$$

The preservation of the set from Ψ_{max} /as it is in engineers technique/ gives the machine with five internal states.

4. Dynamical machines

Additional simplification of the machine is possible if we can apply dynamical states and united with these, pulse signals. It is the consequence of the fact, that additional information, introduced by the dynamical states and readed over by pulse elements, can be used for inscription of new states independently from the level signals in the statical machine.

Theorem 2. When $/s_1,x_i/$ is a stable total state, each unstable state $s_2 = \delta/s_1 x_j/$ can be replaced by a stable state (s_1,x_j) if:

1. the pulse signals are introduced, which write down a state s_2 /from s_1/ when each of x_i changes into x_j,
2. a new state s_1 /at x_j/ is not constrained by the level signals[x]/.

Proof. If at the state $/s_1,x_i/$ the input changes into x_j and, at the same time disappear the level signals, which constrain the state s_1 /condition 2/, then the machine will remain at the state s_1. Into the suitable square of the tabde we can write down the state s_1. Since the level signals does

---

[x]/ It means that signals, which are functions of statical states s and x, does not operate on the memory circuits.

not work, the pulse signals can introduce the state s_2, without obstacles. So each change of x which gives s_2 in the statical machine, gives s_2 in the dynamical machine, too /condition 1/. The operation of the machine does not change, what proofs the theorem.

Second condition of theorem 2 is real when the switching functions are obtained. At the minimization process one can suppose that it is satisfied and only the first conditions is obligatory.

The admissible /by theorem 2/ transformations of the table we can use for its simplification.

If two internal states one can change by one, with introduction of i pulse signals, then such states are called the joint states of i-th rank.

Theorem 3. The state s_1 /stable at x_k/ such that $\delta/s_1, x_j/=s$ and the state s_2 /stable at x_j/ are the joint of i-th rank, if:

1. without consideration of i input signals x_j /i columns of x_j/ is $s_1 \sim s_2$
2. $\alpha\left[s_2, \frac{x_k}{x_j}\right] = s_1$ /if exists/
3. at the change of the state from $/s_1, x_k/$ across $/s_1, x_j/$ till $/s_1, x_j/$ each output change the value not more than once.

Proof. For confirmation that the states s_1 and s_2 are joint it is necessary and sufficient that they are compatible /it arises from definitions/. Without of i columns x_j the states are compatible /condition 1/ and signals of $\alpha\left[s \frac{j x_k}{x_j}\right] = s_1$ can write down the desirable state even at $s=s_1$ as at $s=s_2$ /condition 2/. The output signals are not disturbed /condition 3/, so with i columns the states are compatible, too. It proofs the theorem.

From theorem 3 arises a procedure of minimization of dynamical machines. If the table of the statical machine has two states which are not compatible in consequence of unstable states, we ought to verify if the remaining conditions /2 and 3/ of theorem 3 are satisfied. If it is so, then they are joint and can be changed by one state. Such proofs we must to repeat for all pairs of states, but it is not difficult if we have the minimal table of statical machine.

The investigation of the joint states in flow table is similar as the investigation of the compatible states, but is easier, because the conditions are simpler. Therefore we can use similar auxiliary diagram - the joint diagram, on which the internal states are jointed by such number of lines, as is the rank of joint of the states.

The states which are mutually joint give a joint set. As previously , from the joint diagram we can obtain the minimal class Θ_{min} of the maximal joint sets. For every such a class we can write down the number ρ - the sum of all ranks of pairs which are closed in the sets of the class. Number ρ is also a number of dynamical signals and therefore ρ is a index of circuits complication. If the machine has a few classes Θ_{min} than we must select such a class, for which the index is minimal.

The minimal flow table of dynamical machine we can built from the best class Θ_{min}; every set of the class is then the internal state of minimal machine and the suitable states are obtained by dynamical signals.

Obtaining of a state s_2 bydynamical signal: $\alpha\left[s_1 \quad \frac{x_1}{x_2}\right] = s_2$, we mark in the flow table by small arrow, which begins in the square of total state $/s_1,x_1/$ changes the direction in $/s_1,x_2/$, and ends in $/s_2,x_2/$.

The technique we will show in example.

For the machine with table from figure 2c a pair of states /1,4/ is joint with rank 2, because for change of unstable state 3 /in column x_3/ a pulse signal is necessary, and for change of unstable state 2 /at x_1/ - the second pulse signal is necessary. Likewise - the states 2 and 3 are joint with rank 2, and a simple diagram /figure 3a/ gives synonymously:

$$\Theta_{min} = \{ 1,4;\ 2,3 \}$$

The minimal flow table of dynamical machine is presented on figure 3b. By four pulse signals number of memory elements is reduced to one.

Obtaining of the switching function from the flow table with arrows is not difficult[7].

The proofs of practical application of the method show that number of elements in the dynamical machine amounts 50-70% of elements in the statical machine.

References

1. Ginsburg S.: An Introduction to Mathematical Machine Theory. Addison - Wesley, 1962.

2. Miller R.E.: Switching Theory. J.Wiley, 1965.

3. Łazariew W.G., Pijl E.I.: Sintiez asynchronnych konieČnych awtomatow. 1964.

4. Paull M.C., Unger S.H.: Minimizing the Number of States... IRE Trans.EC 3/1959.

5. McCluskey E.J.: Mnogotaktnyje schiemy... "Tieoria konieČnych i wierojatnostnych awtomatow", 1965.

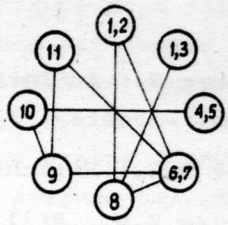
6. Maley G.A., Early J.: The Logic Design of Transistor Digital Computer. Everitt, 1963.

7. Traczyk W.: Projektowanie tranzystorowych układów przełączających WNT, 1966.

a.

S	x_1	x_2	x_3	x_4	x_5	y
1	(1)	4	6	8	–	–
2	(2)	5	–	8	–	0
3	(3)	–	7	8	–	1
4	3	(4)	–	9	10	0
5	1	(5)	–	9	10	0
6	2	–	(6)	–	11	1
7	1	–	(7)	–	11	1
8	1	–	–	(8)	–	0
9	2	–	–	(9)	–	1
10	1	–	–	9	(10)	0
11	1	–	–	9	(11)	1

b.



c.

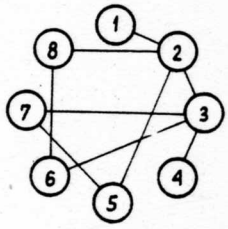
S	x_1	x_2	x_3	x_4	x_5	y
1,2,8 – 1	(1)	3	4	(1)	–	0
1,3 – 2	(2)	3	4	1	–	1
4,5,10 – 3	2	(3)	–	4	(3)	0
6,7,9,11 – 4	1	–	(4)	(4)	(4)	1

Fig. 1

a.

S	x_1	x_2	x_3	x_4	y
1	2	(1)	7	–	0
2	(2)	–	–	6	0
3	(3)	5	–	–	0
4	3	–	8	(4)	0
5	2	(5)	7	–	1
6	3	–	8	(6)	1
7	–	5	(7)	4	0
8	–	1	(8)	6	1

b.

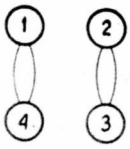


c.

S	x_1	x_2	x_3	x_4
1,2 – 1	(1),0	(1),0	3	4
3,4 – 2	(2),0	3	4	(2),0
5,7 – 3	1	(3),1	(3),0	2
6,8 – 4	2	1	(4),1	(4),1

Fig. 2

a.



b.

S	x_1	x_2	x_3	x_4
1,4 – 1	(1),0	(1),0	(1),1	(1),1
2,3 – 2	(2),0	(2),1	(2),0	(2),0

Fig. 3

ЭВРИСТИЧЕСКИЕ МЕТОДЫ СТРУКТУРНОГО СИНТЕЗА РЕЛЕЙНЫХ УСТРОЙСТВ

Чл-корр.АН СССР М.А.Гаврилов
Институт Автоматики и телемеханики. Москва СССР

I. Постановка задачи

Широкое развитие вычислительных и управляющих машин и устройств дискретного действия/релейных устройств/ поставило, как одну из актуальнейших проблем сегодняшнего дня, проблему синтеза их структур. Основной задачей здесь является построение оптимальной в определенном смысле структуры устройства, удовлетворяющей заданному для него алгоритму функционирования, заданной надежности и заданным структурным свойствам элементов, на которых устройство должно быть реализовано.

Современные задачи синтеза структуры релейных устройств имеют ряд особенностей, которые требуют значительного пересмотра раннее разработанных методов синтеза и существенного развития существующей теории релейных устройств и конечных автоматов. К этим особенностям относятся:

а/Существенное усложнение структурных свойств элементов на которых реализуются структуры релейных устройств; а именно: реализация релейных устройств на т.н."микромодулях", имеющих сложные структурные свойства, и на элементах т.н." однородных сред", в которых должны рассматриваться совокупности элементов со связями между ними.

б/Существенное увеличение"объемов"релейных устройств и усложнение условий их работы. Уже в настоящее время требуется решать задачи синтеза релейных устройств с числом

входо-выходов и числом внутренних состояний порядка сотен.

В области структурного синтеза в настоящее время наиболее развиты и теоретически обоснованы методы, дающие реализацию структур в нормальной /дизъюнктивной или конъюнктивной/ форме, что соответствует реализации структур на элементах типа: "И", "ИЛИ", "НЕ". Эти методы основаны в большинстве случаев на определении из таблиц состояний, с помощью которых задаются условия работы релейных устройств, т.н. "минимальных членов" -таких произведений букв, из которых нельзя исключить ни одной буквы без того, чтобы была нарушена непротиворечивость реализации таблицы состояний, т.е. не было получено такое произведение букв, которое содержится, в состоянии, дающем единицу на выходе, и в состоянии, дающем нуль на нем. Из множества минимальных членов затем путем перебора возможных подмножеств их отбираются подмножества, образующие т.н. "неизбыточные реализации"[I/] и из всех неизбыточных реализаций - также путем перебора - все реализации, удовлетворяющие заданному функционалу минимизации, например имеющие минимальное число элементов /будем называть их "абсолютно минимальными" реализациями/.

Требующийся для этого перебор весьма быстро возрастает. Например, максимальное число минимальных членов составляет:

I/ Неизбыточность здесь понимается в том смысле, что без получения противоречивой реализации нельзя: а/подать 0 или I на вход какого либо элемента, на который ранее не была подана константа и б/ нельзя исключить какой либо части структуры путем подачи выхода выходного элемента какой либо другой части структуры на выход всей структуры в целом или на вход какого либо элемента другой части структуры.

для 5-ти переменных-32, для IO-ти-4,300, для I5-ти-759.488 и т.д. [I]. Число операций, необходимых для определения минимальных неизбыточных реализаций, растет ещё быстрее и нахождение оптимальных реализаций структур для числа переменбольшего I2-I5-ти, становится практически неосуществимым даже для универсальных вычислительных машин. Ещё более громоздкой является реализация структур в скобочной форме и при наличии многих выходов в виде т.н. "связных" структур.I/ Попытки разработки методов, в которых возможен неполный перебор, не привели к существенному уменьшению громоздкости операций синтеза.

Одним из путей устранения указанных трудностей является т.н. "направленный" поиск оптимальной реализации структур, заключающийся в том, что на каждом из этапов синтеза из всех возможных вариантов дальнейших этапов выбирается при помощи соответствующих оценок некоторый вариант, обеспечивающий реализацию структуры в целом,близкую к оптимальной. При этом в дереве решений выбирается лишь один путь, что существенно снижает громоздкость вычислений. Этот путь приводит к реализации тем более приближающейся к оптимальной, чем точнее критерии для оценки сложности реализации структур.

В общем виде задача направленного поиска оптимальной реализации структур может быть сформулирована следующим образом.

I/ "Связанными" или структурами типа "связанного дерева" будем называть структуры, в которых имеются связи между цепями отдельных выходов.

Пусть заданы условия работы релейного устройства с n входами и κ выходами $F(n,\kappa)$ в виде некоторой совокупности таблиц состояний, в общем случае недоопределенных, где функции, реализуемые на каждом выходе F_i $(1 \le i \le \kappa)$, заданы наборами состояний: $M_1^i = \{\alpha_1^i, \alpha_2^i, \dots, \alpha_m^i\}$, дающими значения на выходе, равные I, и $M_0^i = \{\beta_1^i, \beta_2^i, \dots, \beta_p^i\}$, дающими значения на выходе, равные 0 /будем называть: первые "рабочими состояниями и вторые – "нерабочими состояниями"/.

Пусть задана также некоторая совокупность элементов $\Phi = \{\varphi_1, \varphi_2, \dots, \varphi_t\}$. Структурные свойства каждого из элементов φ_j $(1 \le j \le t)$ характеризуются полностью определенной таблицей состояний, в которой подмножество наборов рабочих состояний: $\pi_1^j = \{\psi_1^j, \psi_2^j, \dots, \psi_r^j\}$ и подмножество наборов нерабочих состояний: $\pi_0^j = \{\xi_1^j, \xi_2^j, \dots, \xi_s^j\}$.

Требуется построить структуру релейного устройства, реализованную на этих элементах, близкую к оптимальной с точки зрения заданного функционала оптимальности. Практически в качестве таковых рассматриваются: минимальное число элементов, минимальное число задействованных входов их, минимальный обобщенный "вес" структуры, если каждый из заданных типов элементов, оценивается каким либо весом, отличным от других, и т.п.

Будем называть булеву функцию $\varphi(y_1, y_2, \dots, y_q)$, реализуемую одним элементом φ_j с числом входов $q(\varphi_j)$, на входы которого поданы переменные: $y_1, y_2, \dots, y_q$, "элементарной" функцией относительно набора элементов, содержащих элемент φ_j. Любая булева функция может быть, очевидно, записана в виде некото-

рой композиции элементарных функций, аргументами которых могут быть либо другие функции / в том числе и элементарные/ либо независимые переменные. В соответствии с этим задача реализации структуры $F(n,k)$ на заданном наборе элементов, заключается в представлении булевых функций, характеризующих условия работы отдельных выходов устройства, в виде композиций элементарных относительно заданного набора функций:

$$F_i = \varphi(y_1, y_2, \dots, y_i, x_{i+1}, x_{i+2}, \dots, x_n) = \varphi(f_1, f_2, \dots, f_q)$$

и выбора из всех возможных композиций такой, которая обеспечивала бы реализацию, близкую к оптимальной. Из числа элементов, входящих в состав элементов, реализующих релейное устройство, ряд элементов являются выходными для структуры в целом. Обозначим их через: $y^0_1, y^0_2, \dots, y^0_k$.

В общем виде алгоритм направленного поиска может быть определен, как состоящий из следующих операций.

а/ Выбор из всех функций F_i такой, реализация которой в первую очередь является наилучшей с точки зрения наиболее оптимального использования её частей для реализации всех остальных функций выходов структуры.

б/ Выбор из всех возможных такого элемента y^0_i на выходе каждой из реализуемых функций F_i, который обеспечивал бы наиболее оптимальную реализацию этой функции в целом.

в/ Выбор для этого элемента наиболее оптимальной последовательности реализации функций $f_g \; (1 \le g \le q)$ на его входах и выбор таких функций f_g, которые, обеспечивая непротиворечивую и неизбыточную реализацию функции F_i, осуществляли бы её наиболее оптимально.

После выбора функций f_q для выходного элемента функции F_i, реализуемой в первую очередь, совокупность этих функций рассматривается, как условия работы некоторого многовыходного релейного устройства и указанные операции повторяются иттеративно до тех пор пока функция F_i не будет реализована полностью.

После этого реализуется следующая по порядку функция F_i причем все выходы элементов, входящих в уже реализованную структуру, принимаются как дополнительные независимые переменные, которые рассматриваются при синтезе структуры остальных выходов наравне с первоначальными независимыми переменными. Это приводит к получению структур типа связного дерева, обеспечивающих в ряде случаев получение более оптимальных реализаций.

2. Критерии для оценки оптимальности структур.

Как уже указывалось, оптимальность получаемой при направленном поиске структуры релейного устройства существенно зависит от характера и точности используемых критериев для оценки оптимальности. Задача получения точных критериев является одной из сложнейших в теории релейных устройств. Работы в этой области развиты явно недостаточно. В существующих разработках рассматриваются главным образом критерии для оценки числа элементов или входов при реализации структур в нормальной форме. Предлагаемые критерии имеют асимптотический характер и в ряде случаев не дают надлежащей ориентировки при решении реальных задач синтеза.

Впервые вопрос об оценке сложности одновыходных структур был рассмотрен в работе К.Шеннона [2] . Полученная К.Шенно-

ном асимптотическая оценка была затем улучшена Г.Поваровым [3] и О.Лупановым [4]. Предложенная последним оценка имеет вид: $L(n) \sim \rho \frac{2^n}{n}$ (1), где n -число независимых переменных и ρ -коэффициент, зависящий от типа применяемых элементов и связности структуры. Была также предложена асимптотическая оценка, учитывающая распределение состояний между рабочими и нерабочими: $L(k,n) \sim \rho \frac{\log_2 C_{2^n}^k}{\log_2 \log_2 C_{2^n}^k}$ (2), где k -число рабочих состояний.

Приведенные выше оценки справедливы для полностью определенных функций. Между тем большинство практических условий работы, в особенности при большом числе переменных, характеризуются неполностью определенными таблицами состояний. Исследование оценок сложности реализации структур релейных устройств для этого случая проводилось в СССР в Институте Автоматики и телемеханики /технической кибернетики/. Л.Шеломовым [5,6] был предложен асимптотический критерий: $J \sim N \log_2 N - n_1 \log_2 n_1 - n_0 \log_2 n_0$ (3), где n_1 - число рабочих состояний, n_0 -число нерабочих состояний и $N = n_1 + n_0$. Этот критерий дает оценку числа входов элементов типа "И", "ИЛИ", "НЕ". Оценка числа элементов с учётом связности и ветвлений имеет вид: $L(f) \sim \rho \frac{J}{\log_2 J}$ (4) Протекание критерия в зависимости от N и n_1 приведено на рис.I. Статистическая проверка критерия на УВМ показала, что он достаточно правильно отражает изменение сложности и при числе переменных, не стремящихся к бесконечности /рис.2/. П.Пархоменко [7] предложил оценивать сложность реализации недоопределенной таблицы состояний по числу состояний N. Как показал Л.Шеломов, эта оценка верна в асимптотике. Однако

как показали статистические испытания, точность её уменьшается с увеличением сложности реализации структуры /рис.2/.

Несколько другой характер имеет критерий, предложенный В.Копыленко [8]. Он служит для определения очередности выбора переменных, подаваемых на входы элементов, и имеет вид:

$S = \max\{S_1, S_0\}$ (5) , где $S_1 = \frac{n_1^1 n_0^0}{n_1^1 + n_0^0}$ (6) и $S_0 = \frac{n_1^0 n_0^1}{n_1^0 + n_0^1}$

В этих выражениях: n_1^1, n_1^0 -число рабочих состояний, в которых оцениваемая переменная принимает значение единицы или нуля соответственно, и n_0^1, n_0^0 - число нерабочих состояний, в которых оцениваемая переменная также принимает значение единицы или нуля соответственно. Значения S_1 или S_0 определяют близость функции f_g к реализации её одной буквой /как это легко видеть в этом случае критерии S_1 и S_0 имеют наибольшее возможное значение/. При наличии ограничений на число инверсных значений независимых переменных выбор переменных осуществляется в порядке наибольших значений S_1.

Как будет показано ниже, при направленном поиске встречается необходимость оценки сложности реализации многовыходной структуры в целом. В.Шеломов [7] показал, что вычисление такой оценки сложно. Более простая оценка имеет место в частном случае, когда недоопределенные состояния для всех выходов одинаковы. Она имеет вид: $J(F_1,\dots,F_k) \approx N\log_2 N - \sum_{i=1}^{q} l_i \log_2 l_i$ (8) где: l_i -число одинаковых столбцов матрицы, строки которой сопоставлены функциям F_i , а столбцы состояниям, определяющим функции, в клетках же единицами обозначены рабочие состояния и нулями-нерабочие. Просто вычисляемая оценка по-

лучается также в случае, когда в заданной системе функций, реализующих выходы структуры, рабочие и нерабочие состояния попарно не пересекаются. В этом случае оценка имеет вид:
$\mathcal{J}(F_1,\dots,F_k) \sim (N_1+N_0)\log_2(N_1+N_0) - N_1\log_2 N_1 - N_0\log_2 N_0$ (9) где: N_1-общее число рабочих состояний и N_0 -общее число нерабочих состояний.

В указанной работе Л.Шеломова рассмотрен также вопрос о наиболее оптимальной очередности реализации функций на выходах многовыходной структуры. Эта задача была решена также только для частного случая одинаковых недоопределенных состояний для всех выходов структуры. Процедура определения этой очередности состоит в следующем: а/Определяется разница в величине оценки сложности реализации всей многовыходной структуры в целом и сложности её при поочередном удалении строк матрицы, соответствующих функциям F_i. Показывается асимптотически, что функция, удаление которой дает минимальную разность должна быть реализована в последнюю очередь. б/ Строка, соответствующая этой функции, удаляется из матрицы совсем и указанная операция повторяется до тех пор, пока не будет определена очередность реализации для всех выходов структуры без исключения.

Приведенный выше обзор показывает, что существующие работы по оценкам сложности ~~не~~ лишь частично удовлетворяют необходимости в них. Поэтому при разработке методов синтеза, в особенности при программировании их на УВМ, приходится прибегать к чисто интуитивным критериям, целесообразность которых определяется экспериментально при статистической проверке соответствующих методов синтеза.

3.Методы направленного поиска оптимальных реализаций

При разработке методов направленного поиска оптимальных реализаций структур существенным является возможность учёта различного рода практических требований, а именно:

а/ ограничений по числу входов элементов, по затуханию, по коэффициенту разветвлений выходов, по характеру взаимных соединений и т.п.;

б/ необходимости устранения состязаний в цепях выходов элементов;

в/ возможности синтеза структур с заданной надежностью;

г/ универсальности метода с точки зрения применяемых элементов;

и д/ возможности распространения метода на случай задания условий работы релейного устройства и структурных свойств элементов не в виде таблиц состояний, а в интервальной форме, что особенно важно для многовыходных структур с большим числом входных переменных.

Ниже рассматриваются два метода направленного поиска оптимальной реализации структур, разработанные автором совместно с В.М.Копыленко. Прежде чем переходить к изложению их существа, сделаем некоторые замечания по классификации элементов, на которых реализуются релейные устройства.

Введем понятие "характеристического числа" $[\tau_j]$ элемента j представляющего собой минимальное число входов, подача на которые одинаковых воздействий γ ($\gamma \in \{1,0\}$) однозначно определяет воздействие на его выходе ε ($\varepsilon \in \{1,0\}$).

Будем делить релейные элементы на классы во первых по значению характеристического числа, выделив элементы с $[\tau_j]=1$

/например элементы "И","ИЛИ"/ и элементы с $[\tau_j] > 1$ /например мажоритарные элементы/.Кроме того будем различать элементы с "симметричными" входами,для которых значение выхода не изменяется при перестановке переменных на входах: $f(f_1,f_2) = f(f_2,f_1)$ и с " нессиметричными" входами,для которых это значение изменяется.Будем также различать: элементы с " упорядоченными"входами,в которых значение $[\tau_j]$ для любой совокупности входов одинаково,и с "неупорядоченными" входами,для которых значение $[\tau_j]$ для различных совокупностей входов различны.

А.Метод таблиц реализаций.

Рассмотрим существо этого метода на примере. Пусть задана таблица состояний,представленная на рис.3а,Выделим в ней т.н. "обязательные" буквы[I]/ в табл.рис.3 подчеркнуты/.Очевидно,что некоторые из переменных,в которых нет обязательных букв,могут быть удалены. Применяя критерии:(5),(6),(7) будем вычеркивать поочередно эти переменные,начиная с переменной,имеющие наименьшее значение выбранного критерия,т.е. дающей наиболее сложную реализацию функции на соответствующем входе выходного элемента.Каждый раз будем определять новые обязательные буквы,появившиеся после вычеркивания соответствующего столбца таблицы состояний до тех пор пока все столбцы таблицы не будут содержать обязательные буквы

I/Обязательной называется буква,по которой данное состояние является соседним с каким либо состоянием в противоположной таблице состояний.Вычеркивание этой буквы приводит поэтому к противоречивой реализации и она должна обязательно входить в минимальный член,реализующий данное состояние.

Произведения полученных при этом обязательных букв в каждом из состояний образуют два вида минимальных членов: а/если они не содержаться в противоположной части таблицы состояний,то минимальные члены ядра (см. [8]) и б/ если содержатся,то т.н. "недостаточные" минимальные члены,которые для непротиворечивой реализации должны быть доопределены.Будем рассматривать для простоты реализацию таблицы состояний рис. 3а на элементах: "И","ИЛИ" ,"НЕ" с выходным элементом"ИЛИ".После указанных операций структура примет вид,показаний на рис.4

Построим т.н. таблицу "реализаций",/рис.5а/,в которой строки сопоставлены минимальным членам ядра и недостаточным минимальным членам,а столбцы-рабочим/слева/ и нерабочим /справа/ состояниям. В клетках таблицы проставляются в каждой строке крестики в тех столбцах,состояния которых реализуются данным минимальным членом. Минимальные члены ядра реализуют таблицу состояний непротиворечиво и поэтому содержат крестики только в рабочих состояний. Недостаточные минимальные члены имеют крестики,как в рабочих,так и в нерабочих состояниях.Для элементов "И" и "ИЛИ",имеющих $[\tau_g]=1$, в каждом из столбцов достаточно иметь один крестик.Минимальные члены ядра должны быть в любой реализации структуры. Отметим состояния,реализуемые ими в табл.рис.5а птичкой. Оставшиеся состояния реализуются недостаточными минимальными членами избыточно,поскольку в соответствующих столбцах таблицы реализации имеются по несколько крестиков. Поэтому из множества их нужно выбрать такое подмножество,ко-

торое,во первых,реализовало бы все рабочие состояния неизбыточно и,вовторых,требовало бы наиболее простой реализации доопределяющих функций.

Представим совокупность определяющих функций в виде некоторой одной функции: $F = \bigvee_{g=1}^{K} f_g$ где: K -число недостаточных минимальных членов,входящих в какое либо неизбыточное подмножество их.Число рабочих состояний,которое должна реализовать эта функция будет для всех подмножеств неизменным. Потому для получения наименьшей сложности доопределяющих функций нужно выбрать в соответствием с критерием (3)/см.рис.I/ такое подмножество недостаточных минимальных членов,которое имеет крестики в минимальном числе столбцов нерабочей части таблицы реализации.Алгоритм такого выбора содержат следующие операции: а/выбирается минимальный член,в строке которого имеется наименьшее число крестиков в нерабочей части таблицы реализации,а при нескольких таких членах-имеющий наибольшее число крестиков в рабочей части; б/ из всех остальных строк в рабочей и нерабочей частях таблицы реализации вычеркиваются крестики в столбцах,в которых крестики имеются в выбранной строке; в/из оставшихся минимальных членов снова выбирается член, удовлетворяющий пункту а/,и указанные операции продолжаются до тех пор,пока в таблице реализации не окажутся строки ,имеющие только вычеркнутые крестики,что будет свидетельствовать о реализации всех состояний рабочей части таблицы реализации.

Для рассматриваемого примера таблица реализации примет после этих операций вид,показанный на рис.5 б.В оптимальное неизбыточное подмножество в ней входят минимальные

члены ядра: $\bar{x}_9\bar{x}_3$ и $\bar{x}_9x_7\bar{x}_5$ ... и недостаточные минимальные члены: $\bar{x}_5\bar{x}_3$, $\bar{x}_6$ и $\bar{x}_7$. Из той же таблицы реализации определяются таблицы состояний для доопределяющих функций f_1, f_2 и f_3. В качестве рабочих в них входят рабочие состояния соответствующей строки таблицы реализации с невычеркнутыми крестиками, а в качестве нерабочих - все состояния, имеющие крестики /вычеркнутые и невычеркнутые/ в этой строке в нерабочей части таблицы. /см. рис.5в,5г и5д/. Эти таблицы реализуются с помощью тех же операций, что и первоначальная таблица состояний.

Рассмотренный метод синтеза достаточно прост и эффективен для всех элементов с симметричными входами и $[\tau_3]=I$[1], но существенно усложняется для элементов с $[\tau_3]>I$ /см. [9]/.

Б.Метод переходных таблиц.

В этом методе задача направленного поиска решается в её наиболее общем виде, поскольку он является пригодным для общего случая элементов с $[\tau_3]>I$, с несимметричными и неупорядоченными входами. Пусть условия работы синтезируемого релейного устройства заданы таблицей состояний рис.6б с множеством рабочих состояний M_1 и множеством нерабочих - M_0, а структурные свойства элемента, на котором должно быть реализовано это устройство заданы таблицей состояний рис.6а с множеством рабочих состояний π_1 и множеством нерабочих-π_0.

Образуем две функции: h_i, в которой рабочими состояниями являются π_1 и нерабочими π_0 и функцию q_i, в которой рабочими состояниями являются π_0 и нерабочими π_1. Множество всех ми-

---

1/ В число этих элементов входят: "И","ИЛИ","НЕ","ИЛИ-НЕ","И-НЕ" и условно могут быть включены элементы "сумма по модулю два" и "равнозначность".

нимальных членов функции h_i будем обозначать через $H_i = \{v_1, v_2, \ldots, v_e\}$ и называть "рабочей характеристикой" элемента, а множество минимальных членов функции g_i обозначать через $G_i = \{w_1, w_2, \ldots, w_p\}$ и называть "нерабочей характеристикой" элемента. Легко показать, что для непротиворечивой реализации структуры на выходе какого либо элемента необходимо, чтобы для каждого из состояний входов устройства $\alpha \in M_1^i$ существовал хотя бы один минимальный член $v_k \in M_1^i$, а для каждого из состояний $\beta \in M_0^i$ - хотя бы один минимальный член $w_c \in G_i$.

Основные операции, входящие в алгоритм синтеза, являются следующими: а/ Находятся рабочая и нерабочая характеристики элемента; б/ Строится т.н. "переходная" таблица, имеющая число столбцов, равное числу минимальных членов элемента, и число строк, равное числу строк таблицы состояний реализуемой функции. Слева от переходной таблицы размещается таблица состояний функции, вверху - таблица минимальных членов H_i и внизу таблица минимальных членов G_i /рис.6в/; в/ Производится определение очередности заполнения входов элемента и определение переменных, подаваемых на них; г/ Проводится заполнение столбцов переходной таблицы, определение т.н. "жестких" предписаний и составление таблиц состояний для доопределяющих функций.

В первую очередь выбирается вход элемента, имеющий наименьшее число реализующих его букв/ что дает наибольшую свободу дальнейших операций/, ~~при этом~~ и входящих в наиболее короткие минимальные члены/что обеспечивает наиболее быструю реализацию состояний/. В таблице рис.6в таким входом являет-

ся вход y_1. Переменная, подаваемая на входы, выбираются в соответствии с критериями (6) и (7). В соответствии с их значениями /указаны внизу табл.рис.6б/ выбираем по критерию δ_0 переменную x_4. Осуществляя проверку основной таблицы состояний, отмечаем те из них, которые реализуются этой переменной противоречиво /цифра I в состояниях:3,9,0,8,I2,I4), и для которых должна быть поэтому обеспечена реализация на других минимальных членах.

Вторым выбираем вход y_4, т.к. он полностью реализует минимальный член v_2. Т.к. в y_i этот вход реализует две буквы в членах w_1 и w_2, то в состояниях: 0,8,I2,I4 значения переменной подаваемой на этот вход жестко предписаны, т.е.она комбинацией своих значений должна полностью удовлетворять этим состояниям. Такой переменной является $\bar{x}_4$, однако при использовании её член v_2 не будет реализовать ни одного из состояний. Поэтому заменяем переменную на входе y_4 функцией f_4. Рабочими состояниями этой функции должны быть состояния: 0,8,I2,I4, имеющие жесткие предписания, а нерабочими желательно иметь состояния:2,4,6,I0, которые могут быть реализованы с помощью члена v_2. Реализация функции f_4 показана в табл. рис.6г. После получения структуры соответствующего элемента в табл.6в отмечаются состояния, реализованные членом v_2/цифры 4 в кружках/ и вновь появившиеся жесткие предписания /цифры 4 в состояниях:5,7,II/.

Процесс реализации заканчивается, когда будут заполнены все входы элемента и реализованы все состояния из M_1^i и M_0^i.

Описанный алгоритм достаточно прост и универсален. Некоторые видоизменения его позволяют получать структуры, свобод-

ные от состязаний на выходах элементов и синтезировать структуры при задании их и структурных свойств элементов в интервальной форме.

М.А.Гаврилов

Цитированная литература: I.В.Д.Казаков.Нахождение максимального числа простых импликантов произвольной логической функции переменных. Сб.Автоматическое регулирование и управление. Изд.АН СССР,I960г. 2.C.Shannon.A.Simbolic Analisis of 2. Relay and Switching Circuits.Tranc.of. AYFE 57/I938/,713-723
3.Г.Н.Поваров. О синтезе контактных многополосников. Доклады АН СССР, XCIУ, /I954/, №6, I075-I078.
4.О.Б.Лупанов. Об одном подходе к синтезу управляющих систем принципе локального кодирования. Сб.Проблемы кибернетики.I4 изд.Наука, I965, 3I-II0.
5.Л.А.Шеломов. О функционалах, характеризующих сложность систем недоопределенных булевых функций. Сб.Проблемы кибернетики, вып.,I9. изд.Наука, I967, I23-I39.
6.Л.А.Шеломов. Критерии сложности булевых функций. Сб.Проблемы Кибернетики, вып.I7, I966, 9I-I27.
7.П.П.Пархоменко. Синтез релейных устройств на различных функционально полных системах логических элементов. Автоматика и телемеханика, 25, /I964/, №6, 963-979.
8.В.П.Диденко. К построению частных минимальных форм булевых функций. Сб.Логический язык для представления алгоритмов синтеза релейных устройств. Изд.Наука, I966, I94-203.
9.М.А.Гаврилов, В.М.Копыленко. Синтез структуры бесконтактных релейных устройств. Bull.Math,de la Soc.Sci. Math.Phys.de la RSR,IO /58/,I966,3,227-259

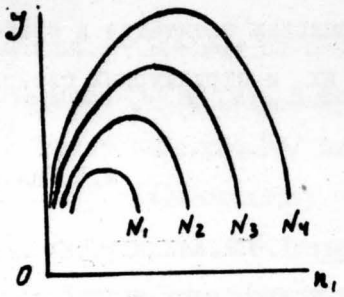


Рис. 1.

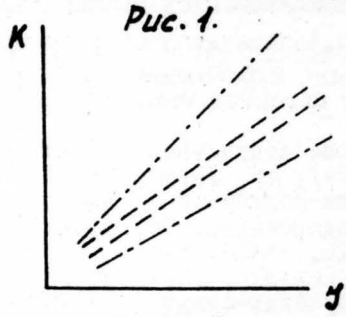


Рис. 2.

—значение сложности по критерию, K-действительная сложность, ——— критерий Л. Шоломова, -·-·-·- критерий П. Пархоменко.

	x_{10}	x_9	x_8	x_7	x_6	x_5	x_4	x_3	x_2	x_1
65	0	0	0	1	0	0	0	0	0	1
123	0	0	0	1	1	1	1	0	1	1
170	0	0	1	0	1	0	1	0	1	0
344	0	1	0	1	0	1	0	0	0	0
360	0	1	0	1	1	0	1	1	0	0
437	0	1	1	0	1	1	0	1	0	1
450	0	1	1	1	0	0	0	0	1	0
615	1	0	0	1	1	0	0	1	1	1
680	1	0	1	0	1	0	0	0	0	0
704	1	0	1	1	0	0	0	0	0	0
808	1	1	0	0	1	0	1	0	0	0
358	0	1	0	1	1	0	0	1	1	0
376	0	1	0	1	1	1	0	1	1	0
391	0	1	1	0	0	0	1	1	1	1
424	0	1	1	0	1	1	0	1	0	0
511	0	1	1	1	1	1	1	1	1	1
559	1	0	0	0	1	1	0	1	1	1
629	1	0	0	1	1	1	1	1	1	0
902	1	1	1	0	0	0	1	1	1	0

а).

x_9	x_8	x_7	x_6	x_5	x_3	
0	0	1	0	0	0	
0	0	1	1	1	0	$\bar{x}_9\bar{x}_3$
0	1	0	1	0	0	
1	0	1	0	0	0	$\bar{x}_6$
1	0	0	1	0	0	$\bar{x}_5\bar{x}_3$
1	1	1	1	0	1	$\bar{x}_7$
1	1	0	0	0	1	
1	0	1	0	0	1	$\bar{x}_9x_7\bar{x}_5$
0	1	1	1	0	0	$\bar{x}_9$
0	1	1	0	0	0	
1	0	1	0	0	0	$\bar{x}_8$
1	0	1	1	0	1	x_9x_3
1	0	1	1	1	0	$x_9x_6x_5$
1	1	1	0	0	1	
1	1	1	0	0	0	x_9x_8
1	1	1	1	0	1	$\bar{x}_7$
1	1	0	1	1	1	$\bar{x}_7$
0	0	1	1	1	1	x_5x_3
1	1	0	0	0	1	

б).

Рис. 3.

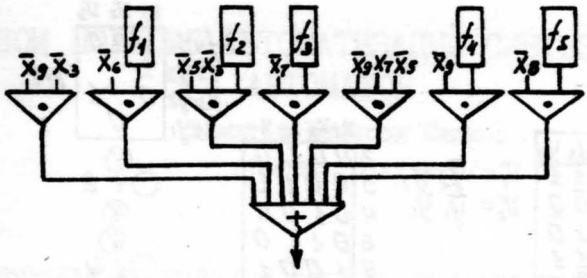


Рис. 4.

	65	123	344	360	437	450	613	680	704	808	358	376	391	424	511	559	629	
$\bar{X}_9\bar{X}_3$	+	+						+	+									4-0
$\bar{X}_6$	+		+			+			+				+					4-1
$\bar{X}_5\bar{X}_3$	+			+		+		+	+	+				+				6-1
$\bar{X}_7$					+			+		+			+	+		+		3-3
$\bar{X}_9X_7\bar{X}_5$	+						+		+									3-0
$\bar{X}_9$	+	+					+	+	+							+	+	5-2
$\bar{X}_8$	+	+	+	+			+			+	+	+				+	+	6-1

а).

	$\bar{X}_9\bar{X}_3$	+	+						+	+								
f_2	$\bar{X}_6$	*		+			*			*				+				2-4,1-1
f_1	$\bar{X}_5\bar{X}_3$	*			+		+		*	*	+				+			3-1,
f_3	$\bar{X}_7$					+			*		*			*	*		+	2-3,1-2,1-1
	$\bar{X}_9X_7\bar{X}_5$	*						+		+									
	$\bar{X}_9$	*	*					*	*	*							+	+	0-2
	$\bar{X}_8$	*	*	*	*			*			*	+	+				+	+	3-4,1-4,0-4

б).

	X_9	X_8	X_7	X_6	X_5	X_3
437	1	1	0	1	1	1
391	1	1	0	0	0	1
424	1	1	0	1	0	0
559	0	0	0	1	0	1

$f_3 = X_5$

в).

	X_9	X_8	X_7	X_6	X_5	X_3
344	1	0	1	0	1	0
391	1	1	0	0	0	1

$f_2 = \bar{X}_8 = X_7 = X_5 = \bar{X}_3$

г).

	X_9	X_8	X_7	X_6	X_5	X_3
360	1	0	1	1	0	0
450	1	1	1	0	0	0
898	1	0	0	1	0	0
391	1	1	0	0	0	1

$f_1 = \bar{X}_3$

д).

Рис. 5.

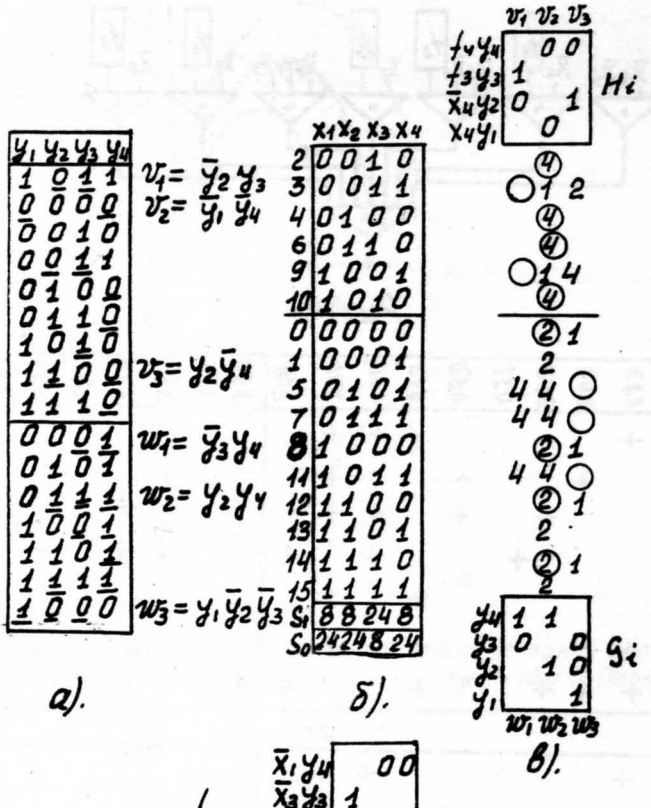

а).
б).
в).
г).
Hi
Gi
$v_1 = \bar{y}_2 \bar{y}_3$
$v_2 = \bar{y}_1 \bar{y}_4$
$v_3 = y_2 \bar{y}_4$
$w_1 = \bar{y}_3 y_4$
$w_2 = y_2 y_4$
$w_3 = y_1 \bar{y}_2 \bar{y}_3$
S_1
S_0
f_4

Рис. 6.

ОБ ОДНОМ ПОДХОДЕ АВТОМАТИЗАЦИИ СИНТЕЗА КОНЕЧНЫХ АВТОМАТОВ

Димитър Борисов Шишков
/София, Болгария/

Автоматизация процесса синтеза конечных автоматов при помощи ЭЦВМ затрудняется отсутствием согласованности во входных - выходных языках известных алгоритмов, решающих задачи отдельных этапов синтеза и возможности применять оптимизационных приемов на каждом этапе с учетом оптимального решения задачи в целом, а также неособенно высокой эффективностью некоторых алгоритмов синтеза.

В настоящей работе рассматривается подход автоматизации процесса оптимального синтеза конечных автоматов, который включает класс алгоритмов, основанных на языке структурных функций возбуждения и выходов[I]. Предложенные алгоритмы имеют аналитический характер, позволяют проводить единую оптимизационную линию в процессе синтеза, приложимы для достаточно широкого класса конечных автоматов /полностью и частично определенных, минимальных и неминимальных автоматов типа Мура, Мили, Мура - Мили/, выполняемых на произвольных функционально полных наборах элементов, а их эффективность сравнима или более высокая, чем эффективность известных аналогичных алгоритмов.

Для общности рассматривается модель конечного автомата типа Мили, определенного на множествах внутренних состояний $A = (a_0, a_1, \ldots, a_{n-1})$, где a_0 – начальное состояние автомата, входных / $X = (x_0, x_1, \ldots, x_{m-1})$, $x = \{0,1\}$/ и выходных сигналов / $Y = (y_0, y_1, \ldots, y_{p-1})$, $y = \{0,1\}$/ и рекурентных соотношений $a(t+1) = \delta[a(t), x(t)]$ и $y(t) = \lambda[a(t), x(t)]$, детерминированных на этих множествах и представляющих функции переходов и выходов автомата.

В качестве абстрактного языка записи условия функционирования автомата предлагается удобная с точки зрения инженерной практики интерпретация автомата в виде произвольной по сложности

и типах используемых элементов структурной схемой.

Переход от структурной схемы к основному для данного подхода языку структурных функций возбуждения и выходов осуществляется в следующей последовательности:

I. Составляются функции возбуждения / $q_i' = q_i'(x_0, x_1, \ldots, x_{m-1}, Q_1, Q_2, \ldots, Q_s)$, где Q_k - функция выхода k-го элементарного автомата в структурной схеме заданного автомата, а $i = 1,2,\ldots,s \geq \lceil \log_2 n \rceil$/ и функции выходов / $z_j = z_j(x_0, x_1, \ldots, x_{m-1}, Q_1, Q_2, \ldots, Q_s)$, где $j = 1,2,\ldots,\gamma \geq \lceil \log_2 p \rceil$ /.

2. Функции возбуждения q_i' / $i = 1,2,\ldots,s$ / приводятся к функциям возбуждения /q_i/ элементов задержки с помощью соотношений, приведенных в табл.I, причем минимизация полученных зависимостей необязательна.

табл.I

Функция возбуждения триггера	Функция возбуждения элемента задержки	Примечание
со счетным входом (q_{si}')	$q_i = \bar{q}_{si}' Q_i \vee q_{si}' \bar{Q}_i$	-
с раздельными входами (q_{oi}', q_{1i}')	$q_i = q_{1i}' \vee \bar{q}_{oi}' Q_i$	$q_{oi}' q_{1i}' = 0$
с дублированными переходами (q_{oi}', q_{1i}')	$q_i = \bar{q}_{oi}' Q_i \vee q_{1i}' \bar{Q}_i$	-
с тремя входами ($q_{si}', q_{oi}', q_{1i}'$)	$q_i = q_{1i}' \vee q_{si}' \bar{Q}_i \vee \bar{q}_{si}' \bar{q}_{oi}' Q_i$	$q_{si}' q_{oi}' = q_{1i}' q_{si}' = q_{oi}' q_{1i}' = 0$

3. Переход к структурным функциям возбуждения F_k / $k = 0,1,\ldots,n-1$ / и выходов Ψ_j / $j = 0,1,\ldots,p$ / выполняется на основании следующих формул:

$$F_k = \langle \tilde{q}_1 \cdot \tilde{q}_2 \ldots \tilde{q}_s \rangle_k , \qquad /1/$$

$$\Psi_j = \langle \tilde{z}_1 . \tilde{z}_2 \ldots \tilde{z}_4 \rangle_j , \qquad /2/$$

где $k = 1,2,\ldots,2^s$, а $j = 1,2,\ldots,2^4$.

В /1/ и /2/ везде полагается $\langle \tilde{Q}_1 . \tilde{Q}_2 \ldots \tilde{Q}_s \rangle_k = M_k$ и $\langle \tilde{x}_0 . \tilde{x}_1 \ldots \tilde{x}_{m-1} \rangle_j = X_j$ / $j = 0,1,\ldots,2^m-1$/, где M_k и X_j конституенты единиц, соответствующих состоянием автомата a_k и структурному входному сигналу X_j.

4. Устраняются недостижимые состояния автомата, полагая

$M_k = 0$, если $F_k = 0$ и

$M_k = 0$, если $F_k = F_k(x_0, x_1, \ldots, x_{2^m-1}, M_k)$.

Если автомат задан некоторым структурным языком, то определяя в соответствии с К.Бержом[2] все переходы /полные состояния $x_j M_k$ / как заходящие, исходящие и петлевые, можно, пользуясь табл.2, получить искомые структурные функции возбуждения и выходов.

Исходными данными для структурного синтеза являются структурные функции возбуждения и выходов автомата и способы кодирования его состояний, заданные в виде множеств двоичных разбиений типа $r_i = \{A_0^i, A_1^i\}$ / $i = 1,2,\ldots,s$ /, где A_0^i и A_1^i соответствуют непересекающимся подмножествам состояний заданного автомата, закодированных состояниями 0 и 1 i-го элементарного автомата / i-го структурного входа или выхода/.

Канонические функции возбуждения и выходов автомата, выраженных в совершенной дизъюнктивной нормальной форме, получаются с помощью соотношений, соответствующих выбранным типам элементарных автоматов /см. табл.2/. Для этой цели в полученных соотношений везде полагается $M_k = \langle \tilde{Q}_1 . \tilde{Q}_2 \ldots . \tilde{Q}_s \rangle_k$, где M_k определяется на основании выбранного варианта кодирования /множества двоичных разбиений состояний автомата $r_1, r_2, \ldots, r_s$ /.

Степень удовлетворения различных требований, предъявляемых к синтезируемым автоматам, определяются выбором подходящего множества двоичных разбиений $r_1, r_2, \ldots, r_s$.

Сущность алгоритма экономичного кодирования состоит в следующем. По фиксированному множеству двоичных разбиений, характеризирующих все неэквивалентные способы кодирования состояний автомата, применяя вышеописанный метод структурного синтеза,

табл.2

Тип /элементарного/ автомата	Структурная функция возбуждения /выхода/	Переходы /состояния/, содержащиеся в структурной функции возбуждения /выхода/	Функция возбуждения /выхода/	Примечание*
Элемент задержки	F_k	Заходящие и петлевые переходы	$q_i = \bigvee\limits_{k \in A_1^i} F_k$	-
Триггер с раздельными входами	F_k	Заходящие и петлевые переходы	$q_{0i} = \bigvee\limits_{k \in A_0^i} F_k \oplus \sum\limits_{l \in A_0^i} q_j M_l$ $q_{1i} = \bigvee\limits_{k \in A_1^i} F_k \oplus \sum\limits_{l \in A_1^i} q_j M_l$ где $K_j M_l \subset F_k$	Все петлевые переходы разметить неопределенным коэффициентом q = 0 или I.
Триггер с тремя входами	F_k	Заходящие и петлевые переходы	$q_{0i} = \bigvee\limits_{k \in A_0^i} F_k$ $q_{1i} = \bigvee\limits_{k \in A_1^i} F_k$	Все переходы разметить неопределенным коэффициентом q = 0 или I. Коэффициенты в одинаковых импликантах функции q_{si} и q_{0i}, а также q_{si} и q_{1i} должны быть ортогональными.
	F_{sk}	Заходящие и исходящие переходы	$q_{si} = \sum\limits_{k \in A_1^i} F_{sk}$	
Триггер со счетным входом	F_{sk}	Заходящие и исходящие переходы	$q_{si} = \sum\limits_{k \in A_1^i} F_{sk}$	-
Триггер с дублированными переходами	F_k	Заходящие исходящие и петлевые переходы	$q_{0i} = \sum\limits_{k \in A_0^i} F_k$ $q_{1i} = \sum\limits_{k \in A_1^i} F_k$	Все петлевые и исходящие переходы разметить неопределенным коэффициентом q = 0 или I.
Мили	ψ_j	Все переходы, отмеченные выходным сигналом y_j	$z_i = \bigvee\limits_{j \in A_1^i} \psi_j$	-
Мура	ψ_j	Все состояния, отмеченные выходным сигналом y_j	$z_i = \bigvee\limits_{j \in A_1^i} \psi_j$	-

* Все неопределенные /условные/ переходы частично определенных автоматов нужно разметить неопределенным коэффициентом q = 0 или I.

строится полная система канонических функций /3/ и /4/:

$$q_i = q_i(x_0, x_1, \ldots, x_{m-1}, M_0, M_1, \ldots, M_{n-1}), \quad i = 1,2,\ldots,2^{n-1}-1 \qquad /3/$$

$$y_j = y_j(x_0, x_1, \ldots, x_{m-1}, M_0, M_1, \ldots, M_{n-1}), \quad j = 1,2,\ldots,2^{p-1}-1 \qquad /4/$$

С помощью соотношения

$$\bigvee_{k \in A} M_k = M_{\ell_1} \vee M_{\ell_2} \vee \ldots \vee M_{\ell_h} = \tilde{Q}_j , \qquad /5/$$

где $\tilde{Q}_j = Q_j$, если $(\ell_1, \ell_2, \ldots, \ell_h) = A_1^j$, $\tilde{Q}_j = \overline{Q}_j$, если $(\ell_1, \ell_2, \ldots, \ell_h) = A_0^j$ и $\tilde{Q}_j = 1$, если $(\ell_1, \ell_2, \ldots, \ell_h) = A_0^j \cup A_1^j = A$, эти функции выражаются в абсолютно минимальной дизъюнктивной нормальной форме. Далее выбирается такая экономичная /с точки зрения технической реализации/ система функций, которая содержит минимальное число неявных аргументов. Это сводится к выполнению следующих требований:

$$\sum_{i=1}^{s} b_i = B_{min} , \qquad /6/$$

$$\prod_{i=1}^{s} r_i = 0 \quad \text{и} \qquad /7/$$

$$\sum_{i=1}^{s} c_i = C_{min} , \qquad /8/$$

где b_i - число необходимых технических элементов /в условных единицах/ для реализации i-ой функции; c_i - число неявных аргументов i-ой функции, а B и C - некоторые целые положительные числа.

При экономичного кодирования без критических состязаний дополнительно вводится множество характеристических разбиений $\tau_{k min} = \tau_{k min}[r_k, \delta(a,x)]$, эквивалентных анализируемым функциям[3]. Эти разбиения определяют требования к совместимости различных функций с целью устранения критических состязаний. Вариант экономичного кодирования без критических состязаний выбирается на основании условий /6/, /7/, /8/ и /9/.

$$\tau_{k min} \leq r_1 \cdot r_2 \ldots r_{k-1} \cdot r_{k+1} \ldots r_s \qquad /9/$$

Описанные методы кодирования нетрудно расспространить для совместного кодирования внутренних, входных и выходных сос-

тояний полностью и частично определенных, минимальных и неминимальных автоматов[4]. В последнем случае в процессе кодирования выявляется целесообразность минимизации числа состояний автомата с точки зрения простоты структуры его реализаций. Это основывается на следующем.

Наличие совместимых состояний у автомата означает, что размерность множества его внутренних состояний может быть уменьшена до некоторой величины $h < n$. Так как совместимым состояниям могут быть присвоены одинаковые коды, условие однозначности кодирования /см. /7// для неминимальных автоматов представится выражением

$$\prod_{i=1}^{s} r_i \geq 0 \qquad /10/$$

Следовательно любой неминимальный автомат может быть представлен системой функции, состоящей из $s_{min} =]\log_2 h[\leq s =]\log_2 n[$ функций возбуждения и $\eta \geq]\log_2 p[$ функций выходов. Чтобы блоки разбиения /10/ включали только совместимые состояния, необходимо и достаточно, чтобы выбранная система функций возбуждения и выходов зависела только от входных сигналов и аргументов $Q_1, Q_2, \ldots, Q_s$.

При такой постановке задача минимизации числа внутренних состояний заключается в отыскании системы функций возбуждения и выходов, для которой $s_{min} =]\log_2 h[$, а задача получения экономичной системы неминимального автомата - в определении такой системы, которая имеет самую экономичную реализацию /при этом необязательно $s = s_{min}$/.

Основные критерии, определяющие выбор оптимального варианта кодирования, можно дополнить другими требованиями: быстродействие и ограничения на числе входов логических элементов[5], минимальная частота срабатывания вторичных элементов, допустимые нагрузки на выходах элементарных автоматов, надежность и др.

Принципиальным недостатком данной группы алгоритмов кодирования является их пригодность для кодирования состояний сравнительно несложных автоматов. Для осуществления оптимального синтеза сложных автоматов при использовании этих алгоритмов удобно применять способы агрегатной декомпозиции автомата в

виде иерархических и каскадных структур[6].

В поисках избежать вышеуказанного недостатка известных алгоритмов кодирования предлагается регулярный метод оптимального структурного синтеза, основанный на прямом использовании структурных функций возбуждения и выходов в качестве канонических. Метод позволяет удовлетворять одновременно большой комплекс требований: отсутствие критических состязаний, быстродействие при использовании двухвходовых логических элементов совпадения, минимальное число инверторов и экономичность.

Функции возбуждения и выходов по данному методу образуются в соответствии с табл.2 и минимизируются по зависимостям

$$M_k = Q_k , \qquad /II/$$

$$x_j \bigvee_{i \in J} M_i = x_j \overline{\bigvee_{k \in K} M_k} , \qquad /I2/$$

где $\overline{\bigvee_{k \in K} M_k}$ представляет ортогональную запись функции $\bigvee_{i \in J} M_i$, а J и K – непересекающиеся подмножества внутренних состояний автомата, причем $J \cup K = A$.

Из соотношений /II/ и /I2/ видно, что при данном методе отпадает необходимость решения задач кодирования, структурного и комбинационного синтеза, как они понимаются в традиционном смысле, что делает метод пригодным для оптимального синтеза автоматов произвольной сложности на любых функционально полных системах технических элементов без необходимости применять декомпозиционные приемы.

В заключении следует отметить, что достоинства предлагаемого подхода автоматизации процесса синтеза конечных автоматов определяются в значительной мере использованием нового структурного языка – язык структурных функций возбуждения и выходов. Разработанные на его основе алгоритмы оптимального синтеза автоматов характеризируются следующими наиболее существенными особеностями:

I. Имеют активный характер /приложимы для оптимального синтеза любых типов автоматов произвольной сложности, реализуемых на произвольных функционально полных наборах элементов/,

2. Их эффективность сравнима или превышает эффективность известных аналогичных алгоритмов.

3. Удобны для использования в инженерной практике /в процессе синтеза отпадает необходимость построения громоздких кодированных таблиц переходов и выходов автомата и учета собственных переходов элементарных автоматов; позволяют проводить оптимальный синтез с учетом различных требований, причем перебор вариантов на этапе кодирования существенно сокращен по сравнению с наиболее расспространенными алгоритмами кодирования; являются достаточно формализованными/.

4. Особенно удобны для автоматизации процесса синтеза конечных автоматов с помощью ЭЦВМ.

Цитированная литература:

I. Е.Н.Вавилов, Д.Б.Шишков - Анализ и синтез автоматов, заданных системой функций возбуждения и выходов - Кибернетика, 1967, № 4.

2. К.Берж - Теория графов и ее применения - Издательство иностранной литературы, Москва, 1962.

3. Д.Б.Шишков - Икономично кодиране на устойчиви крайни автомати - Известия на Института по техническа кибернетика при Българската академия на науките, София, 1967, т.7.

4. Д.Б.Шишков - Некоторые вопросы анализа и синтеза дискретных автоматов, диссертация, София - Киев, 1967.

5. Д.Б.Шишков - Оптимални решения при синтез на бързодействуващи автомати - Техническа мисъл, 1968, № 3.

6. Д.Б.Шишков - Декомпозиция автоматов в виде иерархических и каскадных структур - Доклады Болгарской академии наук, 1968, т.21, № 1.

HOMOMORPHISMS AND CODES FOR SEQUENTIAL MACHINES

by Pierre TISON, Saint-Avold, FRANCE

---

1 - REPRESENTATIVE FUNCTIONS OF SEQUENTIAL MACHINES

Let us associate to the set of two lines i and j of the flow-table of a given sequential machine, a degree of freedom $^{1, 2}$ of which the value will be 1, if we accept the possibility of merging these two lines and 0 otherwise. The aggregation of these variables for every pair of states $\{1, 2\}$, $\{1, 3\}$, ..., $\{2, 3\}$, $\{2, 4\}$, ... gives a primary variable g, called a grouping variable. Its values express some freedom allowed to the lines merging. Thus the value 111010 of g represents for a 4-stated machine the set of the groupings equal to or smaller than $\{124, 13\} : \{12, 13, 4\}$, $\{24, 13\}$, ... $\{1, 2, 3, 4\}$.

Likewise, we associate to each pair $\{i, j\}$ a degree of freedom of which the value will be 1 if and only if we accept the possibility of parting the states i and j, that is to say of not merging the lines i and j. The aggregation of these elementary variables gives a primary variable called the parting variable of which each value expresses some freedom in the parting of the states set. Thus the value 000101 represents the set of the partings equal to or smaller than $\{124, 13\} : \{124, 123\}$, ..., $\{1234\}$.

We call morphism variable the secondary variable constructed by aggregating a grouping variable associated to the present time (beginning of a transition from a state to another state) and a parting variable associated to the next time (end of a transition).

A truth function can express a flow-table in the following way : Let us consider an elementary value of the morphism variable (having only one 1 per component). The 1 of the grouping variable represents the freedom to merge the two corresponding present states, whereas the 1 of the parting variable represents the freedom to part the two corresponding next states. If and only if, these two freedoms are compatible one with the other, the given term is a canonical implicant of the representative function of the machine.

Let us consider for instance the machine M1 given by the flow-table T1 and the value 100-100 of the associated morphism variable. We represent by arrows on figure 1 the transitions from the present states to the next states. The merging of the present states 1 and 2 requires the merging of the next states 1 and 3, what is not inconsistent with the parting of the next states 1 and 2. Therefore 100-100 is not an impliquant. We find through this process the canonical implicator { 100-100, 100-001, 010-100, 010-001, 001-100, 001-010, 001-001 } of F1, representative function of M1.

The set of the canonical implicants implies the representative function of the machine. We can determine whether a value of the morphism variable is or is not an implicant : given any implicant, every merging of present states equal to or smaller than the first component must be compatible with every parting of the next states equal to or smaller than the second component.

We can compute all the implicants of a machine from the canonical implicator by carrying out the algorithm of successive concentrations [3]. We operate, first with respect to the parting component, then to the grouping one. We get likewise at the end of the first step the primitive implicants which are directly deducible from the flow-table. An implicant, prime with respect to its second component, is called a morphene for the machine. The computation of the morphenes for M1 is shown on figure 2.

We get, in addition to the trivial morphenes 000-111, the primitive ones 100-101, 010-101 and 001-111 which are 1-concentrates, the non-primitive ones 110-101, 101-101, 011-101 and 111-101 (2-concentrates). The prime implicants are morphenes (marked * on figure 2).

2 - RESEARCH OF HOMOMORPHISMS [4]

Given an implicant $m(i) = g(i)\ p(i)$, we may compare the groupings $g(i)$ and $p(i)'$ (the grouping $p(i)'$ has the same writting as the parting $p(i)$). Suppose we have $g(i) \geqslant p(i)'$. Let us propose the performing, first on the present states of all the mergings allowed by $g(i)$, second on the next states of all the partitions allowed by the parting $g(i)'$. These two operations are compatible one with the other since $g(i)' \leqslant p(i)$. As a result $g(i)$ is an homomorphic grouping and the supposed condition ($g(i) \geqslant p(i)'$) is sufficient.

Conversely, suppose we have an homomorphism corresponding to the grouping g(j). Let be g(k) an elementary grouping (having only one 1 in its writting) equal to or smaller than g(j) and m(k) = g(k) p(k) the corresponding morphene. We have $p(k) \geqslant g(j)'$ since each 1 in g(j)' is compatible with every 1 in g(j), therefore compatible with g(k). The morphene g(j) p(h) relative to g(j) is the 2-concentrate of all the m(k)'s. p(h) is the product of the p(k)'s. As each term of the product is equal to or greater than g(j)', p(h) is equal to or greater than g(j)'. $[p(h) \geqslant g(j)']$ is a proposition equivalent to $[g(j) \geqslant p(h)']$. Therefore the condition is necessary.

THEOREM 1 - A grouping g(i) gives an homomorphism if and only if the morphene m(i) = g(i) p(i) relative to g(i) is so that $g(i) \geqslant p(i)'$.

Such a morphene is called a homomorphene. It may be computed by 1-concentrating the primitive morphenes.

The algorithm is carried out through successive steps, each of them corresponding to a primitive implicant, taken by increasing rank to form the concentrate. We may cancel, in the course of this computation, each term such that 101000-001111 since it is impossible, from this value to form implicants having a 1 in the second position of the first component by logical sum of values having 1 only after the third position. The presence of 0 in the second position of the second member would impose the 0 on any term ulteriorly formed since we perform a logical product on the second component. We would obtain terms .0.... - .0.... for which it would be impossible to get the relation $g(i) \geqslant p(i)'$, since p(i)' would be .1....

For the machine M2 (table T2) we have 6 primitive implicants (see figure 3).

The computation of homomorphenes is performed through successive steps, each of them being devoted to a primitive term. We form, in the course of any step, new terms by 1-concentration (with respect to the grouping variable) on the pairs { primitive implicant characteristic of the step, one of the implicants known at the beginning of the step }. At the beginning of the computation, we deal with the trivial implicant 000000-111111 which corresponds to the isomorphism. At the end of the

computation, we get the second trivial morphene 111111-..... which corresponds to the total merging of the states.

Figure 4 shows the computation for M2. The 7 homomorphenes are marked (✔). In addition to the two trivial terms, we get 5 homomorphisms corresponding to the groupings {14, 2, 3}, {14, 23}, {124, 13}, {123, 124} and {124, 134}.

The value of the output corresponding to a state may be represented by a term of a primary variable expressing some freedom. 2 states are equivalent if and only if the corresponding values of the output are compatible one with the other, that is to say the product of the corresponding values of the output variable is not null. We get thus an algorithm for determining the states equivalence.

3 - DETERMINATION GRAPHS

A coding variable is a variable which parts the set of states into two disjoined blocks. The corresponding grouping is called its effect. A code is a set of coding variables. Its effect is the product of the effects of its elements. A complete code has for effect the set of the states disjoined each one to the other. A code both complete and irredundant is called an assignment [5].

There are 3 coding variables for machine M1 : c_1 of which the effect is {12, 3} what may be written g(1) = 100, $c_2 \longrightarrow$ {13, 2} $\longrightarrow$ g(2) = 010 $c_3 \longrightarrow$ {1, 23} $\longrightarrow$ g(3) = 001. We have 3 assignments : $\{c_1, c_2\}$, $\{c_1, c_3\}$ and $\{c_2, c_3\}$ (see figure 6). The present value of a coding variable is a Boolean function of the present state. Its next value is a sequential function of the input, that is to say a combinatory function of the present values of the coding variables and of the input. We say that a coding variable is determinable from a code $\{c_j, \ldots, c_k\}$ if and only if we may deduce without any ambiguousness the next value of c_i from the present values of $c_j, \ldots, c_k$ and of the next input.

THEOREM 2 - A coding variable c_i is determinable from a code $\{c_j, \ldots, c_k\}$ if and only if the morphene g(h) p(h) that we form with g(h), product of the effects g(j), ..., g(k) of the coding variables $c_j, \ldots, c_k$ is so that p(h)' is a grouping equal to or smaller than the effect g(i) of c_i.

The condition is necessary. We suppose that the next value of c_i is computable from the present values of c_j, ..., c_k and the next input. Let us show that $p(h)' \leqslant g(i)$. To say that $\left[p(h)' \leqslant g(i)\right]$ is a true proposition is to say that there is in the grouping $p(h)'$ at least one 1 which is not in $g(i)$, that is to say that on the one hand, there are two states s and t of the machine which are separated by c_i and at every time take distinct values and on the other hand, it was impossible to put a 1 in $p(h)$ what means that there is in $g(h)$ a 1 representating the possible mergings of the states u and v, and that, by the same transition in the flow-table, we go from the present states u and v to the next states s and t, what indicates that from the same block of $g(h)$ we go, by the same transition to s and t. Therefore there is no determinability of c_i from $\{c_j, \ldots, c_k\}$ what is inconsistent with the hypothesis and prooves that $p(h)' \leqslant g(i)$.

The condition is sufficient. Let us suppose that the morphene $g(h)\ p(h)$ is such that : $p(h)' \leqslant g(i)$. From two states belonging to the same block of $g(h)$ we may go into the same block of $p(h)'$, therefore a fortiori into the same block of $g(i)$. The next value of c_i combinatorily depends upon the input and the present values of c_j, ..., c_k, what prooves that the coding variable c_i is determinable from the code $\{c_j, \ldots, c_k\}$.

We have thus settled a method to point out to the determinations of the coding variables among themselves : given the determining set $\{c_1, \ldots, c_m\}$, we compute the product $g(1) \cdot \ldots \cdot g(m)$ of the effects which gives the first component $g(h)$. We look for the corresponding morphene $g(h)\ p(h)$ and find each determinable coding variable c_i by testing the relation $p(h)' \leqslant g(i)$.

A coding variable c_i is <u>irredundantly determinable</u> from the code $\{c_j, \ldots, c_k\}$ if and only if it is not determinable from any subset of this code. We write this irredundant determinability by the term i-j...k which will be called a <u>segment</u> relative to i, a n-ary segment if there are n elements j, ..., k in the second member. We get easily the set of segments since, if there is a segment i-j...k, on the one hand there is no segment relative to i which may be got from the preceding one by cancelling any one of the second component elements, and on the other hand, c_i is determinable from $\{c_j, \ldots, c_k\}$.

After the computing of the morphenes, we proceed through successive steps, the $(n+1)^{th}$ step being devoted to the searching of the n-ary segments.

Let us compute for instance the segments relative to M1 (see figure 5).

1) We take no coding variable. The determining set is empty. Its effect is 111, the corresponding morphene 111-101. We have the relation 010 = g(2) which gives the segment 2 - $\emptyset$. The second component is empty and c_2 is determinable from the inputs only.

2) There are 3 possible codes $\{c_1\}$, $\{c_2\}$ and $\{c_3\}$. For the third of them, the relations 000 $\leqslant$ g(1), 000 $\leqslant$ g(2), 000 $\leqslant$ g(3) give 3 possibilities. From the 5 terms which where formed through this step remains only two since 2-1, 2-2, 2-3 are reduced by 2-$\emptyset$.

3) The 3 possible codes have the same effect 000. Therefore we get only one morphene and 3 relations. For each of them, we have 3 cases corresponding to the possible codes. We get thus 9 terms of which only 2 are segments.

4) We cannot go further than the effect 000. The computation is finished.

We have got 5 segments for M1 : 2-$\emptyset$, 1-3, 3-3, 1-12 and 3-12.

The determination graph of a code has a vertex i for each coding variable c_i and oriented branches from each vertex i to j, ..., k so that i-j...k is a segment. Figure 7 gives for M1 the determination graph and the usual coding equations. The graph visualizes the complexity of the coding equations.

4 - PRINCIPAL CODES

The relation p(h)' $\leqslant$ g(i) is the more easily verified the smaller p(h)' is, that is to say the greater p(h) is. We deduce from this a heuristic for selecting the most interesting coding variables.

We call principal coding variable any c_i of which the effect g(i) leads to a morphene g(i) p(i) so that p(i) is not null. We call principal grouping the effect of a principal coding variable. g(i) p(i) is then a principal morphene. Each principal grouping is formed by two blocks of states. We call left-hand block the one containing the state of minimal rank (0 or 1). The right-hand block is the other one.

The principal groupings computation is performed through 3 successive steps : the first step for the left-hand blocks, the second one for the right-hand blocks, the last one for the principal groupings. For machine M3 given by T3, we have :

1) We start with the block 1 and increase it in every possible way. We keep the 18 left-hand blocks for which the corresponding morphenes do not have a null second component.

2) We have, a priori, 18 possible right-hand blocks which are the complements of the 18 left-hand blocks. After computation of the morphenes, only 13 right-hand blocks remain.

3) We concentrate the morphenes associated to every pair of complementary blocks. At last we get 7 principal groupings.

Figure 8 shows the useful morphenes : suggested grouping as first members, minimal groupings for allowing these suggested groupings as second members.

We have 7 principal coding variables which lead to the assignments $\{c_1, c_2, c_3\}$, ... which we note in a simpler way 123, 124, 126, 127, 135, 136, 137, 145, 146, 147, 156, 157, 234, 235, 237, 245, 246, 256, 257, 267, 345, 346, 347, 356, 367, 467 and 567. Let us compute the principal segments.

- order 1 : 1-3, 2-7, 3-4, 4-1, 5-5, 6-2 and 7-6.
- order 2 : 1-14, 1-26, 1-57, 2-16, 2-24, 2-35, 3-13, 3-27, 3-56, 4-25, 4-34, 4-67, 5-12, 5-37, 5-46, 6-15, 6-36, 6-47, 7-17, 7-23 and 7-45.
- order 3 : 1-125, 1-127, 1-256.

With these principal segments we may part the set of the graphs corresponding to the 27 principal assignments into 10 classes. This points out to interesting properties of M3 : circular determination (code 267), decompositions (codes 135, 145 and 345), clocks (codes 157, 452, 356).

5 - CONCLUSION

We have shown that the n-valued logic algebra introduced for the processing of combinatory functions is also available in sequential algebra for the computation of homomorphisms and codes.

The representation of figure 8 points out to the fact that a morphene is comparable to a partition pair. Hartmanis' theory [6,7] may be considered as a particular case of our n-ary algebra, which appears as an interesting and powerful tool, unifies different logics and brings original results in these domains.

It is interesting to notice that Boolean algebra is the particular case for n = 1 of this n-ary algebra. This must not surprise, since Boole worked only one true propositions where the veracity is indeed a 1-valued object. The falsity of a true proposition is the trivial impossible case allowing without any constraint any logical speculation.

The considering of degrees of freedom is more normal than the considering of states since there is a correspondence between each hypothesis we may arbitrarily state on a system and the constraints this hypothese requires which may be easily deduced from the expression of the problem.

REFERENCES

1 - P. TISON - "Introduction à une algèbre des systèmes logiques" AFIRO Congress, Nancy, May 1967

2 - P. TISON - "Fondement d'une algèbre des systèmes logiques" Automatisme, vol. 12 n° 7 and 8, July-August 1967, pp. 298-303.

3 - P. TISON - "Generalization of consensus theory and application to the minimization of Boolean functions" - IEEE Transactions on electronic computers, vol. EC-16 n° 4, August 1967, pp. 446-456.

4 - P. TISON - "Une logique des machines séquentielles" Automatisme, vol. 12 n° 9, September 1967, pp. 345-351.

5 - P. TISON - "Logic functions and sequential machines" Master's thesis - Polytechnic Institute of Brooklyn, Research report n° 1344-66, June 1967.

6 - J. HARTMANIS and R.E. STEARNS - "On the state assignment problem for sequential machines" - IRE Transactions on electronic computers, vol. EC-10 n° 4, December 1961, pp. 593-603.

7 - J. HARTMANIS - "Pair algebra and its application to automata theory" Information and Control, n° 7, 1964, pp. 485-507.

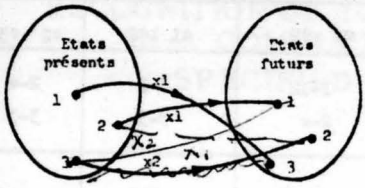


Fig. 1 : Transitions pour M1

M1		Entrées	
		x1	x2
Etats	1	3	
	2	1	2
	3	1	

Table T1

M2		x1	x2	x3
Etats	1	1	2	
	2	3	1	1
	3	3	4	
	4	1	2	4

Table T2

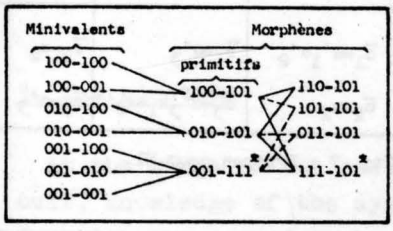


Fig. 2 : Calcul des morphènes pour M1

Fusions proposées	Partitions impossibles	Impliquants primitifs
1-2	1-2, 1-3	100000-001111
1-3	1-3, 2-4	010000-101101
1-4	Néant	001000-111111
2-3	1-4	000100-110111
2-4	1-2,1-3,2-4	000010-001101
3-4	1-3, 2-4	000001-101101

Fig. 3 : Calcul des impliquants primitifs pour M2

Terme initial	:	000000-111111
100000-001111		100000-001111
010000-101101		010000-101101 110000-001101
001000-111111		001000-111111 011000-101101 111000-001100
000100-110111		001100-110111 011100-100101 111100-000101
000010-001101		111010-000101 111110-000101
000001-101101		111011-000101 111111-000101
primitifs utilisés pour la concentration		formés à partir du terme final

Fig. 4 : Homomorphènes pour M1

000111

Etape	Code déterminant	Effet g(h)	Morphène g(h)p(h)	Relations $p(h^{\infty}) \leq g(1)$	Segments
1		111	111-101	010-g(2)	2-v
2	c_1	100	100-101	010-g(2)	~~2-1~~
	c_2	010	010-101	010-g(2)	~~2-2~~
	c_3	001	001-111	000 < g(1)	1-3
				000 < g(2)	~~2-3~~
				000 < g(3)	3-3
3	c_1, c_2	000	000-111	000 < g(1)	1-12 ~~1-13~~ ~~1-23~~
	c_1, c_3			000 < g(2)	~~2-12~~ ~~2-13~~ ~~2-23~~
	c_2, c_3			000 < g(3)	3-12 ~~3-13~~ ~~3-23~~

Fig. 5 : Calcul des segments pour M1

M2		Assignements		
		c_1,c_2	c_1,c_3	c_2,c_3
Etats	1	00	00	00
	2	01	01	11
	3	10	11	01

Fig. 6 : Codage des états de M2

M1

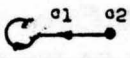
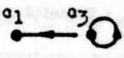
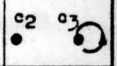
Codes	c_1 ,c_2	c_1 ,c_2	c_2 ,c_3
Segments utiles	1-12 2-v	1-3 3-3	2-v 3-3
graphes			
Equations de codage	$\bar{c}_1=c'_1c'_2$ $\bar{c}_2=x_2$	$\bar{c}_1=c'_3$ $\bar{c}_3=c'_3x_1+x_2$.	$\bar{c}_2=x_2$ $\bar{c}_3=c'_3$

Fig. 7 : Graphes pour M1

Morphènes pour blocs gauches	1,2,3,4,5,6-1,2,3,4,5,6 12,3,4,5,6-15,26,3,4 13,2,4,5,6-12,34,56 14,2,3,5,6-16,2,3,4,5 15,2,3,4,6-13,2,4,5,6 16,2,3,4,5-1,2,35,46 123,4,5,6-1256,34 124,3,5,6-1256,34 125,3,4,6-135,246 126,3,4,5-135,246 134,2,5,6-1256,34 135,2,4,6-1234,56 136,2,4,5-12,3456 145,2,3,6-136,2,4,5 146,2,3,5-146,235 156,2,3,4-135,246 1234,5,6-1256,34 1256,3,4-135,246
Morph.pour blocs droits	1,2,34,5,6-15,26,3,4 1,2,3,4,56-15,26,3,4 1,234,5,6-1256,34 1,235,4,6-146,235 1,236,4,5-1,245,3,6 1,245,3,6-12,3456 1,246,3,5-1234,56 1,256,3,4-135,246 1,2,345,6-145,236 1,2,346,5-145,246 1,2,356,4-145,246 1,2,3,456-145,236 1,3456,2-145,236
Morph.principaux	12,3456-145,236 135,246-1234,56 136,245-12,3456 145,236-136,245 146,235-146,235 1234,56-1256,34 1256,34-135,246

Fig. 8 : Groupements principaux pour M3

M3		Entrées			
		x_1	x_2	x_3	x_4
Etats	1	1	6		3
	2	5	2	4	
	3	2	5		4
	4	6	1	3	
	5	3		6	1
	6		4	2	5

Table T3

Effets

Codeuses principales

$c_1 \rightarrow$ 12,3456
$c_2 \rightarrow$ 135,246
$c_3 \rightarrow$ 136,245
$c_4 \rightarrow$ 145,236
$c_5 \rightarrow$ 146,235
$c_6 \rightarrow$ 1234,56
$c_7 \rightarrow$ 1256,34

Fig. 9 : Codeuses pour M3

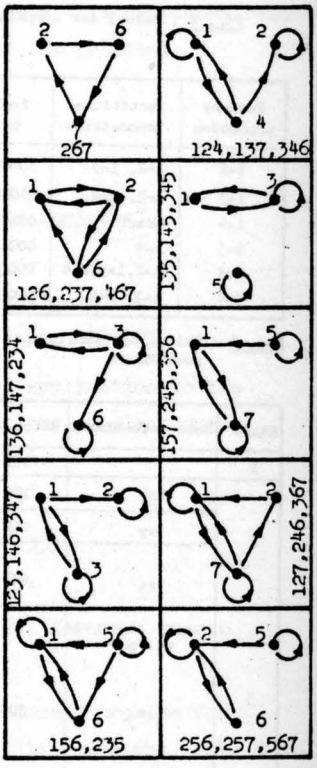


Fig. 10 : Graphes pour M3

RECOGNITION OF TOTAL OR PARTIAL SYMMETRY IN A COMPLETELY OR INCOMPLETELY SPECIFIED SWITCHING FUNCTION

by

Ryszard S. Michalski
Instytut Automatyki PAN, Warszawa, Poland

1. Introduction

At the stage of the structural synthesis of a switching circuit, knowledge of the symmetry properties of the switching function has proved to be very valuable. First, if we do not limit the structure of the circuit, the methods of designing the switching circuits that realize symmetric functions are very simple [1-3]; especially when a circuit is constructed on such elements as relays [1] or threshold elements [3]. If we are interested in obtaining a minimal-cost two-level switching circuit, constructed on conventional gate-type switching elements "or", "and", "not" the problem arises of determining the minimal normal sum-of-products form.

Resolving of this problem can be essentially simplified [4,5], once we have the information that the function is symmetric. This fact is particularly significant, because as a result of Kazakov's work [6], the class of switching functions possibly processing a maximal number of prime implicants, potentially the most difficult to minimize, is included in the class of symmetric functions *. Symmetry information is also useful for showing equivalence between two multiple output switching functions. And most of the gates used today to realize switching circuits produce symmetric functions.

A number of works have been devoted to the problem of recognizing symmetry in switching functions [7-12]. The majority

* Assuming that the class of symmetric functions includes the functions symmetric with respect to the literals, i. e. unprimed or primed variables (or functions $f(x) = x$ and $f(x) = \bar{x}$).

of these works deal with recognizing total symmetry in a switching function. Mukhopadhyay [10] describes a method for determining the total or partial symmetry sets of a completely specified switching function which uses n or $\binom{n}{2}$ respectively so-called "decomposition charts". Schneider and Dietmeyer [12] describe a computer-oriented method for recognizing symmetry in a completely or incompletely specified (multiple output) switching function which involves performing certain operations on the rows of an array which represents the given function. This method deals with recognizing symmetry with respect to (unprimed) variables.

The present paper describes a method for recognizing total or partial symmetry with respect to the literals in a (single or multiple output) switching function which may be completely or incompletely specified. The method represents a new approach which is based on the use a certain two-dimensional topological model of a function, the so-called function image T(f). The use of this model allows the method to be easily applied in hand (for $n \leqslant 7$-8) as well as machine calculations.

2. Notation and Definitions of the Basic Terms

Let $\Omega_j = (\omega_1, \ldots, \omega_n), j \in \{0, 1, \ldots, 2^n - 1\}$, $\omega_i \in \{0, 1\}$, $i = 1, \ldots, n$, denote a sequence of values of the input variables $x_1, \ldots, x_n$ and assume further that $j = \sum_{i}^{n} \omega_i 2^{n-i}$.

Let $\Omega = \{\Omega_j\}$ be the set of all such sequences Ω_j.

Let $Y_k = (\gamma_1, \ldots, \gamma_m)$, $k \in \{0, 1, \ldots, 3^n - 1\}$, $\gamma_i \in \{0, 1, *\}$, $i = 1, \ldots, n$, where $*$ represents an unspecified ("don't care") value, denote a sequence of values of the output variables $y_1, \ldots, y_n$ and assume further that $k = \sum_{i=1}^{m} \hat{\gamma} \cdot 3^{n-i}$, where $\hat{\gamma}_i = \begin{cases} \gamma_i, & \text{if } \gamma_i = 0, 1 \\ 2, & \text{if } \gamma_i = * \end{cases}$. Let $Y = \{Y_j\}$ be the set of all such sequences Y_j.

A multiple output switching function f is then defined as a mapping from Ω into Y $(f : \Omega \rightarrow Y)$. For our considera-

tions now we assume that $m = 1$, but later we will show how to extend the method for the case when $m > 1$.

Let Ω^0, Ω^1, Ω^* denote the sets of sequences Ω_j for which $f(\Omega_j) = 0, 1, *$ respectively. If $\Omega^* = \theta$, where θ is the empty set, then the function f is completely specified and if $\Omega^* \neq 0$ then f is incompletely specified. An incompletely specified function f determines a set $\Phi(f) = \left\{f_i^*\right\}$ of completely specified functions f_i^*, $i = 0, 1, \ldots, 2^{c(\Omega^*)} - 1$, where $c(\Omega^*)$ is the cardinality of the set Ω^*. These functions f_i^* are determined by all possible partitions of the set Ω^* into subset Ω^{*0} and Ω^{*1}, such that $f_i^*(\Omega^{*0}) = \left\{0\right\}$ and $f_i^*(\Omega^{*1}) = \left\{1\right\}$.

<u>Definition 1</u>: A function $f(x_1, \ldots, x_n)$ is said to be <u>symmetric with respect to the set of literals</u> $X = \left\{x_i^{\sigma_i}\right\}_{i \in I}$ where $I \subseteq \left\{1, \ldots, n\right\}$, $\sigma_i \in \left\{0, 1\right\}$ and $x_i^{\sigma_i} = \begin{cases} x_i, & \text{if } \sigma_i = 1; \\ \overline{x}_i, & \text{if } \sigma_i = 0 \end{cases}$ if:

(1) if $\Omega^* = \theta$, then the function is invariant with respect to any permutation of the literals of the set X;

(2) if $\Omega^* \neq \theta$, then in the set $\Phi(f) = \left\{f_i^*\right\}$ there is at least one function f_i^* which satisfies condition (1).

A set of literals X with respect to which the function f is symmetric we will call a <u>symmetry set of</u> f. If X is a symmetry set of f and X is maximal under inclusion among the symmetry sets of X, then we will say simply that X is a <u>maximal symmetry set</u>. Symmetry sets of f of largest cardinality among the symmetry sets of f will be denoted X_m. If $c(X_m) = n$, the function f is said to be totally symmetric, if $1 < c(X_m) < n$, f is said to be partially symmetric.

If $X = \left\{x_k^{\sigma_k}, x_l^{\sigma_l}\right\}$ is a symmetry set of the function f, then, according to definition 1, $x_k^{\sigma_k}$ can be permuted with $x_l^{\sigma_l}$ and $x_l^{\sigma_l}$ with $x_k^{\sigma_k}$ without changing the function f, if f is completely specified or if f is incompletely specified,

without changing some functions f_i^* in the set $\Phi(f)$. We will denote the set of all such functions f_i^* by $\Phi_{k,l}$. The relation saying that the literal $x_k^{\sigma_k}$ is permutable with $x_l^{\sigma_l}$ without changing the function f is written $(x_k^{\sigma_k} \sim x_l^{\sigma_l})_f$. If f is incompletely specified then $(x_k^{\sigma_k} \sim x_l^{\sigma_l})_f$ will be written to mean that the set $\Phi_{k,l}$ defined above is not empty. If functions of the set $\Phi_{k,l}$ are determined we write $(x_k^{\sigma_k} \sim x_l^{\sigma_l})_{\Phi_{k,l}}$. It is easy to prove that the relation $\sim$ is reflexive, symmetric and transitive, but in the case $\Omega^* \neq 0$ transitivity is understood as

$$\left(x_k^{\sigma_k} \sim x_l^{\sigma_l}\right)_{\Phi_{k,l}} \wedge \left(x_l^{\sigma_l} \sim x_m^{\sigma_m}\right)_{\Phi_{l,m}} \Rightarrow \left(x_k^{\sigma_k} \sim x_m^{\sigma_m}\right)_{\Phi_{k,l} \cap \Phi_{l,m}} \quad (1)$$

When $X = \left\{x_1^{\sigma_1}, \ldots, x_m^{\sigma_m}\right\}$ is a symmetry set, a relation $(x_1^{\sigma_1} \sim \ldots \sim x_m^{\sigma_m})_f$ can be defined analogously to the above.

The elementary symmetric functions are:

$$S_0\left(x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\right) = x_1^{1-\sigma_1} x_2^{1-\sigma_2} \ldots x^{1-\sigma_n}$$

$$S_1\left(x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\right) = x_1^{\sigma_1} x^{1-\sigma_2} \ldots x^{1-\sigma_n} \vee x^{1-\sigma_1} x^{\sigma_2} x^{1-\sigma_3} \ldots x^{1-\sigma_n} \vee \ldots$$

$$\vdots$$

$$S_n\left(x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\right) = x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_n^{\sigma_n}$$

Each totally symmetric function is a sum of a set of elementary symmetric functions and can therefore be written in the form $S_A(X)$, where $X = \left\{x_i^{\sigma_i}\right\}_{i=1,\ldots,n}$ is a symmetry set of the function, and $A = \left\{a_i\right\}_{i=1,2,\ldots}$ is the set of indices of the elementary symmetric functions whose sum is the given function (the so-called a-numbers set). It is also well known that:

$$S_A(X) \equiv S_{\hat{A}}(\hat{X}) \quad (2)$$

where $\hat{X} = \left\{ x_i^{\hat{\sigma}_i} \right\}$, $\hat{\sigma}_i = 1 - \sigma_i$ and $\hat{A} = \left\{ \hat{a}_i \right\}_{i=1,2,\ldots}$, $\hat{a}_i = n - a_i$. The function $S_A(x_1, \ldots, x_n)$ is written S_A^n.

Any switching function may be written in the form of the following matrix:

$$\hat{T}(f) = \left[t_{v,h} \right]_{\substack{v=0,1,\ldots,2^{\left[\frac{n}{2}\right]}-1 \\ h=0,1,\ldots,2^{n-\left[\frac{n}{2}\right]}-1}}$$

where $t_{v,h} = f(\Omega_j)$, $j = v \cdot 2^{\left[\frac{n}{2}\right]} + h$ and $\left[\frac{n}{2}\right]$ is the entire part of $\frac{n}{2}$. To the matrix $T(f)$ there correspond in a one-to-one fashion a certain topological model called the image $T(f)$ of the function. The image $T(f)$ is determined with the use of the n-variable logical diagram defined below.

Let us divide any rectangle into $2^{\left[\frac{n}{2}\right]}$ rows and $2^{n-\left[\frac{n}{2}\right]}$ columns according to the rules:

(1) In the first step we divide a rectangle into two rows with a horizontal line. In step m each row obtained in step m-1 we divide into two rows. We execute $\left[\frac{n}{2}\right]$ steps.

(2) The steps $\left[\frac{n}{2}\right] + 1, \ldots, n$ are executed similarly as it was done in rule (1), but by division of the rectangle into columns with vertical lines.

The lines which divide the rectangle in step i $(i=1,\ldots,n)$ we call the axes of the variable x_i. The intersection of any row with any column we call a cell of the logical diagram, assuming that the cell does not include the points belonging to any axis or the perimeter of the rectangle. To set of cells lying above x_1 axis we assign the literal $\bar{x}_1$ and to those lying below the axis the literal x_1. The axes $x_1, x_2, \ldots, x_i$, $i = 1, 2, \ldots, \left[\frac{n}{2}\right] - 1$, divide the rectangle into 2^i sets of cells. To the set consisting of the cells of all the top halves of these sets described above we assign the literal $\bar{x}_{i+1}$ and to set consisting of the cells of all the bottom halves we assign the literal x_{i+1}.

To the set of cells lying on the left of the axis $x_{\left[\frac{n}{2}\right]+1}$ we assign the literal $\bar{x}_{\left[\frac{n}{2}\right]+1}$ and to the set of those lying on the right the literal $x_{\left[\frac{n}{2}\right]+1}$. The axes $x_{\left[\frac{n}{2}\right]+1}, \ldots$ $\ldots, x_{\left[\frac{n}{2}\right]+i}$, $i = 1,2,\ldots, n - \left[\frac{n}{2}\right] - 1$, divide the rectangle into 2^i set of cells. To the set consisting of the cells of all the left halves of those sets described above we assign the literal $\bar{x}_{\left[\frac{n}{2}\right]+i+1}$ and to the set consisting of the cells of all the right halves we assign the literal x_{i+1}.

Definition 2: The figure described above we call an n-variable logical diagram (see Fig. 1 *).

The n-variable logical diagram is similar to the "tables"** of Venn [13], "charts" of Veitch [14], "maps" of Karnaugh [15] and other diagrams. To the logical operations on the literals there correspond set-theoretic operations on the sets of cells assigned to these literals: to the product of literals there corresponds the intersection of sets of cells, to the sum of literals the union of sets of cells. The set of cells corresponding to a product α of literals we will denote by $L(\alpha)$. To a product of n literals $\alpha = x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$ corresponds in a one-to-one fashion one cell e of the diagram. We have $\alpha = 1$, when the variables x_i, $i = 1, \ldots, n$, take values $\omega_i = \sigma_i$. Then the cell e corresponds uniquely to the sequence $\Omega_j = (\omega_1, \ldots, \omega_n)$, where $\omega_i = \sigma_i$ and $j = \sum_{i=1}^{n} \omega_i 2^{n-i}$.

Definition 3. The number of the cell e in an n-variable logical diagram is the number $\gamma(e) = j$, where j is the index of the sequence Ω_j to which corresponds the cell e.

The cell numbers $\gamma(e)$ are distributed in the logical di-

* In this figure a given variable x_i is written beside only one of the x_i variable axes.

** Not to be confused with the "diagrams" of Venn.

agram in an ordered manner.For instance in Fig. 2 is shown the distribution of cell numbers in a 6-variable logical diagram.

Definition 4: The weight of the cell e in an n-variable logical diagram is the number $\delta(e) = \sum_{i=1}^{n} \omega_i$, where ω_i, $i = 1, \ldots, n$, are elements of the sequence $\Omega_{\gamma(e)}$.

The weights of the cells are located in the logical diagram in a certain characteristic order. Fig. 3 presents the example of the distribution of weights in a 6-variable logical diagram.

To each cell e in an n-variable logical diagram let us assign the value $f(\Omega_{\gamma(e)})$ where $f(x_1, \ldots, x_n)$ is a given function.

Definition 5: The set of all cells of an n-variable logical diagram with values assigned as described above we call the image of the function f and denote by $T(f)$.

If e is a cell in $T(f)$ to which was assigned the value p, we will say simply that $e \in T(f)$ has value p. We define $F^p = \{e \in T(f) : e \text{ has value } p\}$, where $p = 0, 1, *$. Let

$$\alpha_1 = \bigwedge_{i \in I_1} x_i^{\sigma_i^1} \quad \text{and} \quad \alpha_2 = \bigwedge_{i \in I_2} x_i^{\sigma_i^2}$$

where $I_1 = I_2 \leqslant \{1, \ldots, n\}$.

Definition 6: A pair of cells (e_j^1, e_j^2), where $e_j^1 \in L(\alpha_j)$ and $e_j^2 \in L(\alpha_2)$ are called a pair of corresponding cells in $L(\alpha_1)$ and $L(\alpha_2)$ if $\gamma(e_j^1) - \gamma(e_j^2) = \sum_{i \in I} (\sigma_i^1 - \sigma_i^2) 2^{n-i}$ where $I = I_1 = I_2$.

It can beseen that corresponding cells in $L(\alpha_1)$ and $L(\alpha_2)$, if $I_1 = I_2$, we can make coincide by a parallel shift (see Fig. 4).

3. Recognition of Symmetry in Completely Specified Functions

3.1. Recognition of Total Symmetry

A given function is totally symmetric if $c(X_m) = n$. Detection of that fact is based on the following theorem:

Theorem 1: A necessary and sufficient condition that a func-

tion $f(x_1, \ldots, x_n)$ be totally symmetric is the existence of a sequence $\sigma = (\sigma_2, \sigma_3, \ldots, \sigma_n)$, $\sigma_i \in \{0, 1\}$, such that

$$\forall (i \in \{2, \ldots, n\}), (x_1 \sim x_i^{\sigma_i})_f$$

Proof: Sufficiency: This results immediately from the fact that the relation $\sim$ is transitive. Necessity: According to formula (2), if the set $\{\bar{x}_1, x_2^{\sigma_2}, \ldots, x_n^{\sigma_n}\}$ is a symmetry set then the set $\{x_1, x_2^{1-\sigma_2}, \ldots, x_n^{1-\sigma_n}\}$ is also a symmetry set. Therefore if the function is totally symmetric, there exists the symmetry set including the literal x_1. Suppose then f is totally symmetric and $\{x_1, x_2^{\sigma_2}, \ldots, x_n^{\sigma_n}\}$ is a symmetry set of f. Then clearly $\sigma_2, \ldots, \sigma_n$ is a sequence with the desired property. Q.E.D.

This theorem is also valid if we permute in any way the indices $i \in \{1, 2, \ldots, n\}$ of the literals $x_i^{\sigma_i}$.

Theorem 1 shows that testing whether the function f is to-symmetric consists of testing for symmetry with respect to certain pairs of literals. The minimal number of pairs to test is $n - 1$ and maximal $2(n - 1)$.

Let us expand the function $f(x_1, \ldots, x_n)$ with respect to the variables x_k and x_l, $k,l \in \{1, \ldots, n\}$, $k \neq n$:

$$f(x_1, \ldots, x_n) = \bar{x}_k\bar{x}_l f^0_{k,l} \vee \bar{x}_k x_l f^1_{k,l} \vee x_k\bar{x}_l f^2_{k,l} \vee x_k x_l f^3_{k,k} \quad (3)$$

where $f^i_{k,l} = f(x_1, \ldots, x_{k-1}, \omega^i_k, x_{k+1}, \ldots, x_{l-1}, \omega^i_l, x_{l+1}, \ldots, x_n)$, $i = 0, 1, 2, 3$, $i = 2\omega^i_k + \omega^i_l$.

Theorem 2: A. $(x_k^{\sigma_k} \sim x_l^{\sigma_l})_f$ where $\sigma_k \neq \sigma_l$; $\sigma_k, \sigma_l \in \{0,1\}$ if and only if $f^1_{k,l} \equiv f^2_{k,l}$ (a)

B. $(x_k^{\sigma_k} \sim x_l^{\sigma_l})_f$ where $\sigma_k \neq \sigma_l$; $\sigma_k, \sigma_l \in \{0,1\}$ if and only if $f^0_{k,l} \equiv f^0_{k,l}$ (b)

Proof:

A. If (a) is satisfied by transformation of the formula (3) we obtain:

$$f(x_1, \ldots, x_n) = \bar{x}_k\bar{x}_l f^0_{k,l} \vee (\bar{x}_k x_l \vee x_k\bar{x}_l) f^1_{k,l} \vee x_k x_l f^3_{k,l}$$

This equation shows that permutations $\begin{pmatrix} x_k, x_l \\ x_l, x_k \end{pmatrix}$ and $\begin{pmatrix} \bar{x}_k, \bar{x}_l \\ \bar{x}_l, \bar{x}_k \end{pmatrix}$ do not change the function f, that is $(x_k^{\sigma_k} \sim x_l^{\sigma_l})_f$ in case $\sigma_k = \sigma_l$. If (a) is not satisfied, then the above permutations change f, proving the necessity of (a).

B. This can be proved analogously. Q.E.D.

Conditions (a) and (b) can be easily checked using the image $T(f)$. The images $T(f^i_{k,l})$, $i = 0,1,2,3$, are determined by the subsets of the image $T(f)$ consisting of the cells of $L^i = L(x_k^{\sigma^i_k} x_l^{\sigma^i_l})$, where $i = 2\sigma^i_k + \sigma^i_l$. Such subsets of $T(f)$ we will denote by $T^i_{k,l}(f)$. Condition (a), (b) is then equivalent to the relation $T^1_{k,l}(f) \doteq T^2_{k,l}(f)$, $(T^0_{k,l}(f) \doteq T^3_{k,l}(f))$, where by $\doteq$ we mean that corresponding* cells in $T^1_{k,l}(f)$ and $T^2_{k,l}(f)$ ($T^0_{k,l}(f)$ and $T^3_{k,l}$) have the same value. When the condition (a) is satisfied we say that f has symmetry of the first kind and when condition (b) is satisfied that f has symmetry of the second kind with respect to the pair of variables $\{x_k, x_l\}$.

Fig. 5 presents algorithm S for testing the total symmetry of the function f based on theorems 1 and 2. The sign := is used as in Algol (it denotes that the variable on the left of the sign takes the new value resulting from the operation written on the right). As a result of algorithm S, we obtain the answer whether the function f is totally symmetric and in the case of a positive answer, a symmetry set X_m is determined. In general the function f may have a number of alternative symmetry sets X^i_m, $i = 1, 2, \ldots$, whose difference from one another is that contain different litterals of the

---

* By corresponding cells in $T^{p_1}_{k,l}(f)$ and $T^{p_2}_{k,l}$ ($p_1 = 1, 0$, $p_2 = 2,3$) we mean corresponding cells in L^{p_1} and L^{p_2}.

same variables. To determine all sets X_m^i, we modify algorithm S , so that both condition (a) and(b) are tested for each pair of variables (x_1, x_i) , $i = 2, 3, \ldots, n$.

In the case of hand realization of the algorithm it is more convenient to adopt a different order of testing for symmetry with respect to pairs of variables, namely to test for symmetry with respect to pairs: $(x_1, x_{[\frac{n}{2}]+1})$, $(x_1, x_{[\frac{n}{2}]+2})$, $\ldots$, (x_1, x_n), $(x_2, x_{[\frac{n}{2}]+1})$, $(x_3, x_{[\frac{n}{2}]+1})$, $\ldots$, $(x_{[\frac{n}{2}]}, x_{[\frac{n}{2}]+1})$. In this case the sets of cells whose values we compare to test for symmetry are characteristically located in the diagram and are easy to define (see Fig. 6).

3.2. Determination of the a-numbers Set

If we have determined that a function f is totally symmetric then we can represent it in the form $S_A(X)$, where A is the set of a-numbers and X is the symmetry set. Having determined that f is totally symmetric, we will also have found a symmetry set $X = X_m$. Thus we need only to determine the set A in order to find the form $S_A(X) \equiv f$. Determination of this set is helped by the following theorem:

Theorem 3: Let f be a function such that $f \equiv S_A(X)$, $X = \{x_1, x_2, \ldots, x_n\}$, $A \subseteq \{0, 1, \ldots, n\}$. The set F^1 is the set of all cells $e \in T(f)$ having weights $\delta(e) \in A$.

Proof: Recall that F^1 is the set of all cells in T(f) with value 1. Let $A = \{a_i\}_{i=1,2,\ldots}$. Because $S_A(X) \equiv f$ then $f = \bigvee_{a_i \in A} S_{a_i}(X)$. A function $S_{a_i}(X)$, $a_i \in A$ has $\binom{n}{a_i}$ components, each including a_i literals x_i and $n-a_i$ literals $\bar{x}_i$. To each component there corresponds the sequence Ω_j including a_i ones and $n - a_i$ zeros, so the weights of the cells corresponding to those sequences are $\delta(e) = a_k$. As the function f is a sum of S_{a_i} , $i = 1, 2, \ldots$, then F^1 consists of all cells, which have weights $\delta(e) \in A$. Q.E.D.

Theorem 3 is illustrated by Fig. 6, which shows the image of the function $S^6_{2,3}$ (the cells of F^1 are dark). Due to this theorem the determination of the a-numbers in case the symme -

try set is $\{x_i\}$, $i = 1, \ldots, n$, may be accomplished by the next operations:

1. Determination of a certain minimal set of cells $E = \{e_i\}$ such that the weights $\delta(e_i)$ exhaust the set of all possible values of the a-numbers i.e. the set $\{0, 1, \ldots, n\}$.

2. The test showing which cells e_i belong to F^1. If $e_i \in F^1$ then $a_i = \delta(e_i) \in A$.

It can be easily proved that $E = E_h \cup E_v$, where E_h consists of the cells e, which have numbers $\gamma(e) = 2^k - 1$, $k = 0,1, \ldots, n - \left[\frac{n}{2}\right]$ and E_v consists of the cells e, which have numbers $\gamma(e) = 2^{n-\left[\frac{n}{2}\right]+k} - 1$, $k = 1, 2, \ldots, \left[\frac{n}{2}\right]$. The set E_h is included in the set $H = L(\bar{x}_1, \ldots, \bar{x}_{\left[\frac{n}{2}\right]})$ consisting of the cellslying in the top row of the diagram. The set E_v is included in the set $V = L(x_{\left[\frac{n}{2}\right]+1}, \ldots, x_n)$ consisting of the cells lying in the right hand column of the diagram. Figure 7 shows the set $E = E_h \cup E_v$ in the 6-variable logical diagram.

Let us now assume that the symmetry set $X_m = \{x_i^{\sigma_i}\}$, $i = 1, 2, \ldots, n$, includes at least one literal $\bar{x}_i$. To reduce the present case to the previous one we might construct an image $T'(f)$ which consists of the cells of a logical diagram, whose variable axes are x_i', $i = 1, \ldots, n$, where $x_i' = x_i^{\sigma_i}$. The image $T'(f)$ can be constructed from $T(f)$ by successively replacing the sets of cells in $T(f)$ which were assigned the literal x_i by those which were assigned the literal $\bar{x}_i$ and vice versa, for each i such that $\sigma_i = 0$.

For determination of the set A we need only test the values of the cells of E, which are included in the set of cells of top row H and right hand column V of $T'(f)$. Therefore it is enough to perform replacements described above inside the set $L(x_1^{1-\sigma_1}, \ldots, x_{\left[\frac{n}{2}\right]}^{1-\sigma_{\left[\frac{n}{2}\right]}})$ for literals $\bar{x}_i \in X$, $\left[\frac{n}{2}\right]+1 \leqslant$

$\leqslant i \leqslant n$ and inside the set $L(x_{\left[\frac{n}{2}\right]+1}, \ldots, x_n^{\sigma_n})$ for literals $\bar{x}_i \in X$, $1 \leqslant i \leqslant \left[\frac{n}{2}\right]$. The sets so obtained will be the top row and right hand column respectively of the image $T'(f)$ which we seek in order to determine A.

3.3. Recognition of Partial Symmetry

A given function $f(x_1, \ldots, x_n)$ is partially symmetric if $1 < c(X_m) < n$. Determination of all the maximal symmetry sets X^i, $i = 1, 2, \ldots$, of function f may be easily accomplished when all symmetry pairs are known. The following implication is helpful here:

$$(\tilde{X}_a \sim x_p^{\sigma_p})_f \wedge (\tilde{X}_b \sim x_p^{\sigma_p})_f \Rightarrow (\tilde{X}_a \sim \tilde{X}_b \sim x_p^{\sigma_p})_f \qquad (4)$$

where $\tilde{X}_a$ represents a literal x_a or a sequence of literals connected by sign $\sim$: $x_{a_1}^{\sigma_{a_1}} \sim x_{a_2}^{\sigma_{a_2}} \sim \ldots$ and $\tilde{X}_b$ - analogously.
This implication results immediately from the transitivity of the relation $\sim$. Fig. 9 presents algorithm S^p for the determination of all the symmetry pairs of the function f. Because the relation $\sim$ is transitive, certain operations of this algorithm may be omitted: if $(x_a^{\sigma_a} \sim x_b^{\sigma_b})_f$ and $(x_a^{\sigma_a} \sim x_c^{\sigma_c})_f$ then also $(x_b^{\sigma_b} \sim x_c^{\sigma_c})_f$ and testing for symmetry with respect to the pair $\{x_b^{\sigma_b}, x_c^{\sigma_c}\}$ is superfluous. When f is totally symmetric, algorithm S^p is equivalent to algorithm S, in which for each pair of variables both conditions (a) and (b) are tested.

3.4. Determination of an Algebraic Form of a Partially Symmetric Function

Let $X = \{x_i^{\sigma_i}\}_{i \in I}$ where $I \subset \{1, \ldots, n\}$ be a symmetry set of the function $f(x_1, \ldots, x_n)$. Let us assume, without loss

of generality that $I = \{1, 2, \ldots, m\}$, $1 < m < n$. Generalizing Shannon's [16] formula to the case of symmetry with respect to literals, the function $f(x_1, \ldots, x_n)$ may be written:

$$f(x_1, \ldots, x_n) = \bigvee_{j=1}^{m} S_j(X) f_j(x_{m+1}, x_{m+2}, \ldots, x_n) \qquad (5)$$

where $f_j(x_{m+1}, x_{m+2}, \ldots, x_n) = f(\omega_1, \ldots, \omega_m, x_{m+1}, x_{m+2}, \ldots, x_n)$, $\omega_i = \begin{cases} 1, & \text{if } x_i \in X \\ 0, & \text{if } \bar{x}_i \in X \end{cases}$, for $i = 1, 2, \ldots, j$, $\omega_i = \begin{cases} 0, & \text{if } x_i \in X \\ 1, & \text{if } \bar{x}_i \in X \end{cases}$, for $i = j+1, j+2, \ldots, m$,

If we have found the maximal symmetry sets X^i of the function $f(x_1, \ldots, x_n)$ for X^i we can determine the form (5). Functions f_j may be determined from any algebraic form of the function f.

4. Recognition of Symmetry in Incompletely Specifies Functions

Let $f(x_1, \ldots, x_n)$ be an incompletely specified function. The function f is symmetric with respect to the pair $(x_k^{\sigma_k}, x_l^{\sigma_l})$. if there exists a non-empty set $\Phi_{k,l} \subseteq \Phi(f)$, such that for each $f_i^{*} \in \Phi_{k,l}$

(a) $T^1_{k,l}(f_i^{*}) \doteq T^2_{k,l}(f_i^{*})$, if $\sigma_k = \sigma_l$

or

(b) $T^0_{k,l}(f_i^{*}) \doteq T^3_{k,l}(f_i^{*})$, if $\sigma_k \neq \sigma_l$

Let us assume that $\sigma_k = \sigma_l$. Let $\{e_j^k, e_j^l\}$, $j = 1, 2, \ldots, 2^{n-2}$, $e_j^k \in T^1_{k,l}(f_i^{*})$, $e_j^l \in T^2_{k,l}(f_i^{*})$ be the sets corresponding cells in $T^1_{k,l}(f_i^{*})$ and $T^2_{k,l}(f_i^{*})$. The set $\Phi_{k,l}$ is determined (with a certain restriction - see row 3 in the table below) by an unspecified function $f_{k,l}$, whose image $T(f_{k,l})$ results from the image $T(f)$ by realization on each pair $\{e_j^k, e_j^l\}$ of the following operations which depend on the values of the cells of this pair:

	e_j^k	e_j^l	
1	0	0	values of e_j^k and e_j^l are not changed
2	1	1	
3	$\varkappa$	$\varkappa$	values are not changed, but for each $f_i^{\varkappa} \in \Phi_{k,l}$, e_j^k and e_j^l must have the same value 0 or 1
4	0	$\varkappa$	values $\varkappa$ are changed to 0 or 1 to obtain equality of values of e_j^k and e_j^l
5	$\varkappa$	0	
6	1	$\varkappa$	
7	$\varkappa$	1	
8	0	1	contradiction, $\Phi_{k,l} = \Theta$
9	1	0	

From this table we see, that a necessary condition for symmetry with respect to the pair $\{x_k^{\sigma_k}, x_l^{\sigma_l}\}$ is that the cells of no pair $\{e_j^k, e_j^l\}$ have values 0 and 1 or 1 and 0 respectively. In case $\sigma_k \neq \sigma_l$ the symmetry condition is analogous, but with respect to the pairs of corresponding cells in $T^0_{k,l}(f)$ and $T^3_{k,l}(f)$.

From the above considerations it follows that for a given image $T(f)$ and values of σ_k and σ_l the set $\Phi_{k,l}$ can be uniquely determined from the ordered triple $(C^0_{k,l}, C^1_{k,l}, R_{k,l})$ where $C^0_{k,l}$ is the set of cells whose values change from $\varkappa$ (in $T(f)$) to 0 (in $T(f_{k,l})$), $C^1_{k,l}$ is the set of cells whose values change from $\varkappa$ to 1 and $R_{k,l}$ is the family of sets $\{e_j^k, e_j^l\}$ such that cells e_j^k and e_j^l of each set have in $T(f)$ value $\varkappa$. The cells of every set $\{e_k^j, e_l^j\} \in R_{k,l}$ in the image of each $f_i^{\varkappa} \in \Phi_{k,l}$ must have the same value 0 or 1.

Let us assume that $(x_k^{\sigma_k} \sim x_l^{\sigma_l})_f$, $(x_l^{\sigma_l} \sim x_m^{\sigma_m})_f$ and that the triples $(C^0_{k,l}, C^1_{k,l}, R_{k,l})$ and $(C^0_{l,m}, C^1_{l,m}, R_{l,m})$ have been determined.

Theorem 4: $(x_k^{\sigma_k} \sim x_l^{\sigma_l})_f \wedge (x_l^{\sigma_l} \sim x_m^{\sigma_m})_f \Rightarrow (x_k^{\sigma_k} \sim x_l^{\sigma_l} \sim x_m^{\sigma_m})_f$ if and only if the following are true:

(a) $C^0_{k,l} \cap C^1_{l,m} = \Theta$

(b) $C^1_{k,l} \cap C^0_{l,m} = \Theta$

(c) $\nexists (c^0, c^1) \in (C^0_{k,l} \times C^1_{k,l} \cup C^0_{l,m} \times C^1_{l,m}) \exists$
$\exists (C^*_i \in R_{k,l} \cup R_{l,m}), \{c^0, c^1\} \subseteq C^*_i$

Proof:

Necessity: Let us assume that $(x_k^{\sigma_k} \sim x_l^{\sigma_l} \sim x_m^{\sigma_m})_f$. Then there exists a non-empty $\Phi_{k,l,m} \subseteq \Phi(f)$ such that for each $f^*_i \in$ $\in \Phi_{k,l,m}$ the set $\{x_k^{\sigma_k}, x_l^{\sigma_l}, x_m^{\sigma_m}\}$ is a symmetry set. According to (1) $\Phi_{k,l,m} = \Phi_{k,l} \cap \Phi_{l,m}$. If (a) or (b) is not true, there exists e, which in the image of every $f^*_i \in \Phi_{k,l}$ has the oposite value than in the image of every $f^*_i \in \Phi_{l,m}$. From this it follows that $\Phi_{k,l} \cap \Phi_{l,m} = \Theta$ and $\Phi_{k,l,m} =$ $= 0$, contrary to our assumption. If (c) is not true, there exists a pair of cells, which for each $f^*_i \in \Phi_{k,l}$ have opposite and for each $f^*_i \in \Phi_{l,m}$ have equal values or vice versa. From this also follows that $\Phi_{k,l,m} = 0$.

Sufficiency: The sufficiency of conditions (a), (b), (c) we will prove by the construction of $\Phi_{k,l,m}$.

Let $C^0_{k,l,m} = C^0_{k,l} \cup C^0_{l,m} \cup C^0_r$ and $C^1_{k,l,m} = C^1_{k,l} \cup C^1_{l,m} \cup C^1_r$ where C^0_r is the set of all cells e^0_1 such that

$$\forall e^0_1 \exists (e^0_2 \in C^0_{k,l} \cup C^0_{l,m}), \{e^0_1, e^0_2\} \in R_{k,l} \cup R_{l,m}$$

C^1_r is the set of all cells e^1_1 such that

$$\forall e^1_1 \exists (e^1_2 \in C^1_{k,l} \cup C^1_{l,m}), \{e^1_1, e^1_2\} \in R_{k,l} \cup R_{l,m}$$

Let $R^0_{k,l}, R^1_{k,l} \subseteq R_{k,l}$ denote the sets containing those members of $R_{k,l}$ which contain the cells e^0_1 and e^0_2 respectively. Let $R^0_{l,m}$ and $R^1_{l,m}$ denote analogous subsets of $R_{l,m}$.

Let $R_1 = R_{k,l} \setminus (R^0_{k,l} \cup R^1_{k,l})$ and $R_2 = R_{l,m} \setminus (R^0_{l,m} \cup R^1_{l,m})$. Let $R_{k,l,m} = R_1 \sqcup R_2$, where $\sqcup$ denotes the following two step operation:

1. The set $R_1 \cup R_2$ is determined.
2. Any pair of members of $R_1 \cup R_2$ whose intersection is non-empty we combine to form a new single member which is the

sum of these members. We then repeat this operation on the newly obtained set in successive such steps until we obtain a set each two of whose members have void intersection. (In the case we now consider this process ends after one step).

If (a) and (b) than $(C^0_{k,l} \cup C^0_{l,m}) \cap (C^1_{k,l} \cup C^1_{l,m}) = \Theta$ (i). If (c) than $C^0_r \cap (C^1_{k,l} \cup C^1_{l,m}) = 0$ (ii) and $C^1_r \cap (C^0_{k,l} \cup C^0_{l,m}) = \Theta$ (iii).

Because the members of $R_{k,l} \cup R_{l,m}$ including any cells of $C^0_{k,l} \cup C^1_{k,l} \cup C^0_{l,m} \cup C^1_{l,m}$ are disjoint then $C^0_r \cap C^1_r = \Theta$ (iv). From (i) - (iv) it follows that

$$C^0_{k,l,m} \cap C^1_{k,l,m} = 0$$

Let $T(f_{k,l,m})$ denote the image of $f_{k,l,m}$ we get after changing in $T(f)$ the values of cells $C^0_{k,l,m}$ from $*$ to 0 and values of cells $C^1_{k,l,m}$ from $*$ to 1. Let $\{f^*_i\}_{i \in I_{k,l,m}}$, $I_{k,l,m} \subseteq \{0, 1, 2, \ldots, 2^{c(\Omega^*)} - 1\}$ be a set of completely specified functions whose images $T(f^*_i)$ we get by changing the values $*$ of cells of $T(f_{k,l,m})$ in every possible way to 0 or 1, but assuming that to cells of every set of the family $R_{k,l,m}$ we assign the same value 0 or 1. For each function f^*_i, $i \in I_{k,l,m}$ we have the relations $(x^{\sigma_k}_k \sim x^{\sigma_l}_l)_{f^*_i}$ and $(x^{\sigma_l}_l \sim x^{\sigma_m}_m)_{f^*_i}$. From this it follows that $(x^{\sigma_k}_k \sim x^{\sigma_l}_l \sim x^{\sigma_m}_m)_{f^*_i}$. Then we have $\Phi_{k,l,m} = \{f^*_i\}_{i \in I_{k,l,m}}$. If there are no cells in $T(f_{k,l,m})$ having value $*$ than $\Phi_{k,l,m}$ consists of only one function $f_{k,l,m}$. Q.E.D.

From the above proof of sufficiency of theorem 4 it results that the set $\Phi_{k,l,m}$ is uniquely determined by the triple $(C^0_{k,l,m}, C^1_{k,l,m}, R_{k,l,m})$. Theorem 4 gives conditions for obtaining three element symmetry sets from two two element symmetry sets whose intersection is non-void. We now generalize this theorem as shown below.

Let $A = \{a_1, \ldots, a_p\}$, $B = \{b_1, \ldots, b_r\}$, $A, B \subseteq \{1, \ldots, n\}, p, r \geqslant$

≥ 2, and $A \cap B \supseteq \{k\}$. Let $\tilde{X}_A$ and $\tilde{X}_B$ be sequences of literals connected by : $x_{a_1}^{\sigma_{a_1}} \sim \ldots \sim x_{a_p}^{\sigma_{a_p}}$ and $x_{b_1}^{\sigma_{b_1}} \sim \ldots \sim x_{b_r}^{\sigma_{b_r}}$ respectively. Let us assume without loss of generality that $a_1 = b_1 = k$. Now assume that $(\tilde{X}_A)_f$ and $(\tilde{X}_B)_f$ and that we have determined the corresponding triples (C_A^0, C_A^1, R_A) and (C_B^0, C_B^1, R_B).

Theorem 5: $(\tilde{X}_A)_f \wedge (\tilde{X}_B)_f \Rightarrow (\tilde{X}_{A \cup B})_f$ if and only if

(a) $C_A^0 \cap C_B^1 = \theta$

(b) $C_A^1 \cap C_B^0 = \theta$

(c) $\nexists ((c^0, c^1) \in C_A^0 \times C_A^1 \cup C_B^0 \times C_B^1) \exists (C_i^{\ast} \in R_A \cup R_B)$, $\{c^0, c^1\} \subseteq C_i^{\ast}$.

Theorem 5 can be proved analogously to theorem 4. As in the proof of sufficiency in theorem 4 we can determine a triple $(C_{A \cup B}^0, C_{A \cup B}^1, R_{A \cup B})$. This triple describes the set $\Phi_{A \cup B} \subseteq \subseteq \Phi(f)$ of completely specified functions for which $(\tilde{X}_{A \cup B})_{\Phi_{A \cup B}}$.

As we have seen above it is easy to determine the symmetry pairs $(x_i^{\sigma_i}, x_j^{\sigma_j})$ and their corresponding triples $(C_{i,j}^0, C_{i,j}^1, R_{i,j})$ using the image T(f). Once knowing all such pairs and triples for f we can by virtue of Theorem 5 compute all maximal symmetry sets of f and their corresponding triples. It is also possible to compute the maximal symmetry sets of f using only the image T(f) (by making the proper assigments of the values $\ast$ in T(f) for each symmetry pair we test in using algorithm S^p). But in this case we must use one „copy" of the image T(f) for each maximal symmetry set so obtained.

Let us assume now that f is an m-output incompletely specified function. It is then equivalent to the set of one -output functions $\{y_j\} = \{f_j(x_1, \ldots, x_n)\}$, $j = 1, \ldots, m$. If we determine the symmetry sets $\{X_j^i\}$ then, as it is easy to prove [12], all symmetry sets of f are the intersections of symmetry sets picked in all possible ways from each $\{X_j^i\}$, $j = 1, \ldots, m$.

5. Conclusions

The method for recognition of symmetry described in this paper may be easily performed by hand calculations for completely specified functions up to 7-8 variables. In the case of incompletely specified functions because in addition we must assign values for α, the practically of this method depends also on the cardinality of Ω^{α}.

In machine calculations this method can be carried out by the use of matrices and submatrices corresponding to the images $T(f)$ and $T^{i}_{k,l}(f)$, $i = 0, 1, 2, 3$ respectively. Operations involving the comparison of matrices which are used in this method are suitable for digital computers, especially of the parallell type.

At the end it is worth noting that the notion of the image $T(f)$ described in this paper is useful also for analysis and synthesis of any type of switching circuits, in particular (as was partly shown in work [17]) for the synthesis of minimal forms of switching functions.

References

1. Lupanov O.V., K voprosu o realizacii simmetricheskikh funkcii algebry lokigi kontaktnymi schemami. Sb. Problemy kibernetiki. Vyp. 15. Moskva 1965.
2. Epstein G., Synthesis of electronic circuits for symmetric functions. IRE Trans. on Electronic Computer, vol. EC-7,pp. 60-67, 1958.
3. Sheng C.L., A graphical interpretation of realization of symmetric Boolean functions with threshold logic elements. IEEE Trans. on Electronic Computers vol. EC-14, pp. 8-18, 1965.
4. Kozakov V.A., Vid minimalnykh form simmetrichnykh buljevykh funkcii proizvolnogo chisla peremennykh. Sb. Avtomaticheskoe regulirovanie i upravlenie. Moscov 1962.
5. Michalski R.S., A paper to appear.
6. Kozakov V.A., Nakhożdenie maksimalnogo chisla prostykh implikantov proizvolnoj logicheskoj funkcii. Sb. Avtomaticheskoe upravlenie. Moskva 1960.
7. Caldwell S.H., The recognition and identification of symmetric switching functions. AIEE Trans. Pt. 2. Communication and Electronics vol. 73, pp. 142-146, 1954.
8. Povarov G.N., O metodike analiza simmetricheskikh kontakt - nykh schem. Avtomatika i Telemekhanika 1955 No. 4.
9. Mc Cluskey E.I., Detection of group invariance or total symmetry of a Boolean function. Bell Sys. Tech. J. vol.35, pp. 1445-1453, 1956.

10. Mukhopadhyay A., Detection of total or partial symmetry of a switching function with the use of decomposition charts. IEEE Trans. Electronic Computers (Correspondence) vol. EC--12, pp. 553-557, 1963.
11. Sheng C.L., Detection of totally symmetric Boolean functions. IEEE Trans. on Electronic Computers (Short notes), vol. EC-14, pp. 924-926, 1965.
12. Dietmeyer D.L., Schneider, Identification of symmetry, redundancy and equivalence of Boolean functions. IEEE Trans. on Electronic Computers, vol. EC-16, pp. 804-817, 1967.
13. Venn J., Symbolic logic. London 1894 pp. 138-140.
14. Veitch E.W., A chart method for simplifying truth functions. Proc. Assoc. for Comp. Machinery, pp. 127-133,Pittsburgh, Pen, Meeting May 2 and 3, 1952.
15. Karnaugh M., The map method for synthesis of combinational logic circuits. AIEE Trans. Pt. 1. Communications and Electronics, pp. 593-599, 1953.
16. Shannon C.E., The synthesis of twoterminal switching circuits. Bell Sys. Tech. J. vol. 28, pp. 69-98, 1949.
17. Michalski R.S., Graficzna minimalizacja w klasie altentatywnych normalnych wyrażeń funkcji logicznych na podstawie tablic typu tablic Veitcha-Karnaugha. Prace Instytutu Automatyki PAN z. 52, Warszawa 1967.

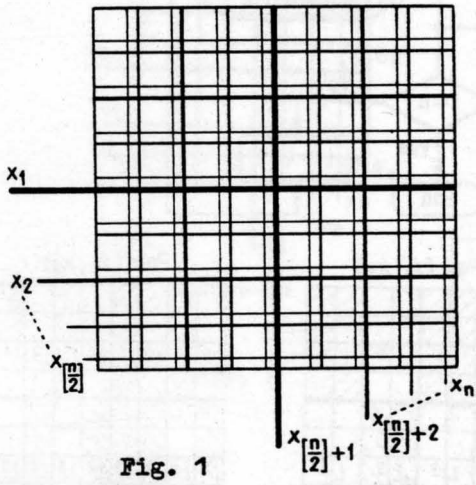


Fig. 1

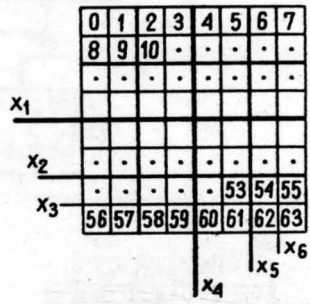


Fig. 2

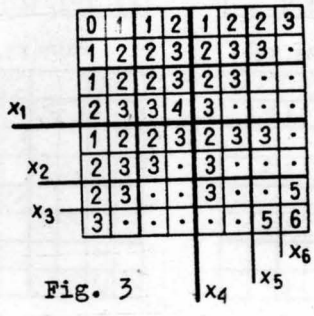


Fig. 3

$\alpha_1 = \bar{x}_1 x_5$

$\alpha_2 = x_1 \bar{x}_5$

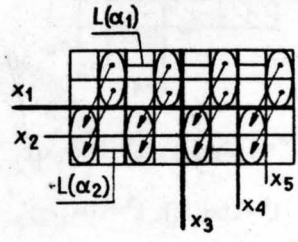


Fig. 4

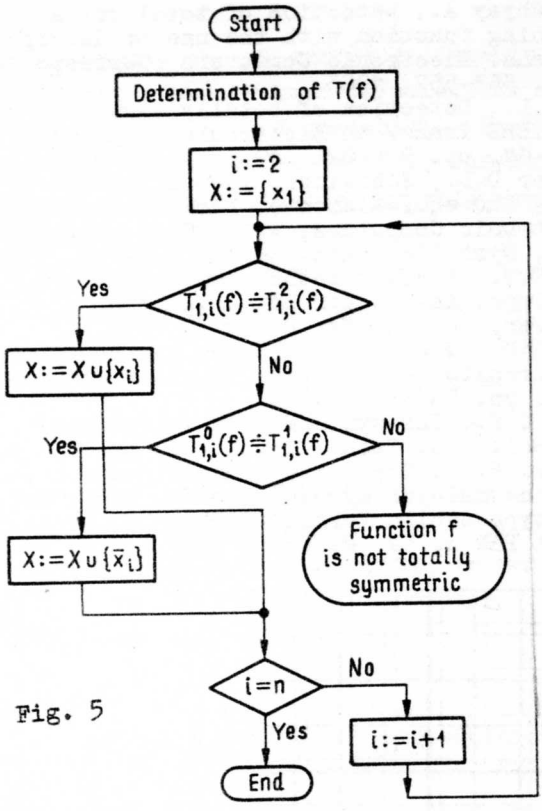
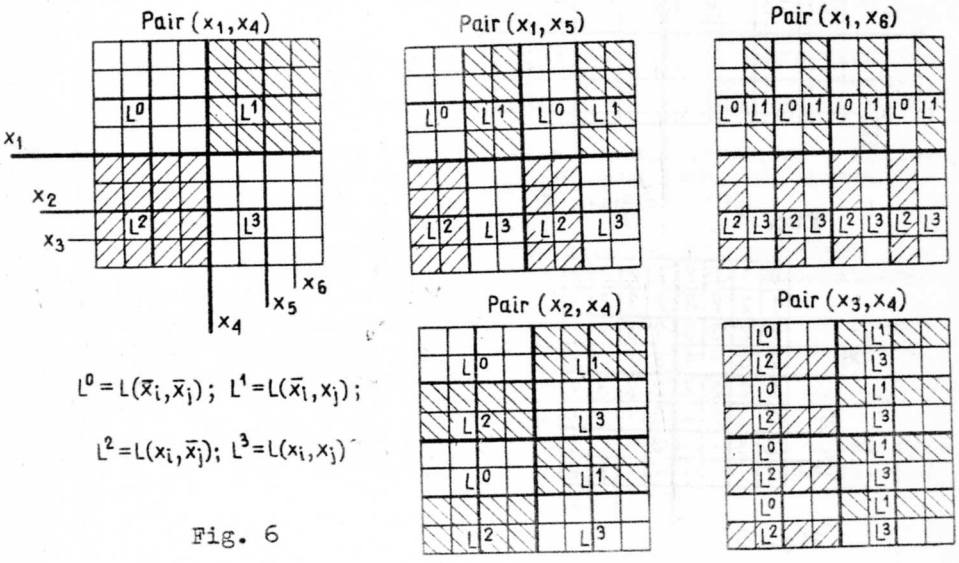


Fig. 5



$L^0 = L(\bar{x}_i, \bar{x}_j)$; $L^1 = L(\bar{x}_i, x_j)$;

$L^2 = L(x_i, \bar{x}_j)$; $L^3 = L(x_i, x_j)$

Fig. 6

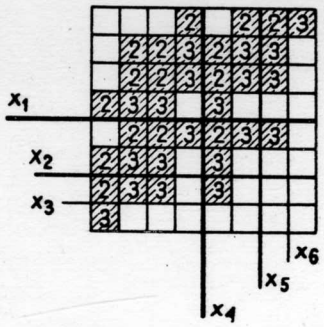
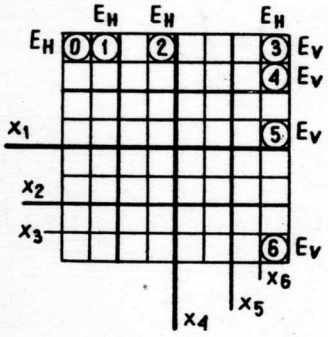


Fig. 7



$E = E_H \cup E_V$

Fig. 8

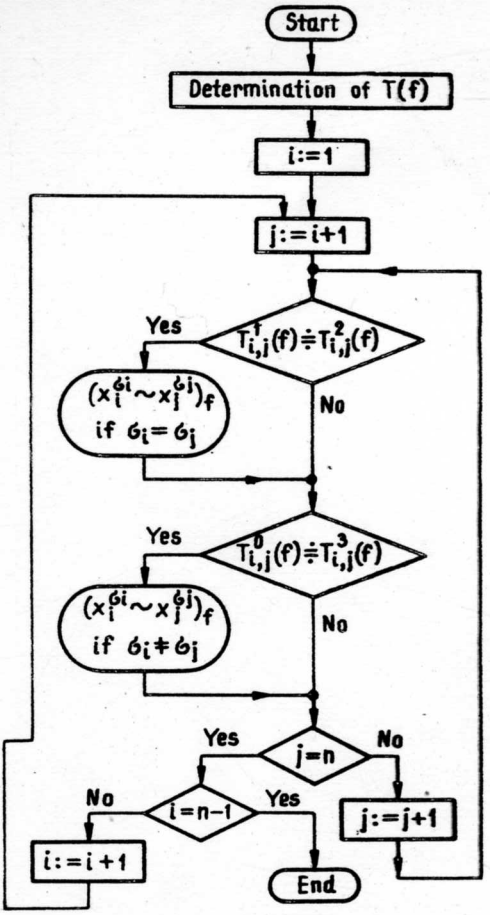


Fig. 9