4

Abstract Structure and Nets

4.1. THE CONCEPT OF SUBSTITUTION OF SEQUENTIAL MACHINES

Consider two sequential machines: a machine s_1 (Fig. 4.1,a), which transforms symbols ρ of an alphabet $\{\rho\}$ into symbols $\varkappa$ of an alphabet $\{\varkappa\}$, and a machine s_2 (Fig. 4.1,b), which transforms symbols η of an alphabet $\{\eta\}$ into symbols ζ of an alphabet $\{\zeta\}$. Let us also

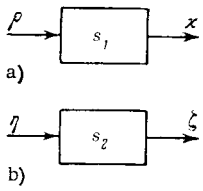


Fig. 4.1.

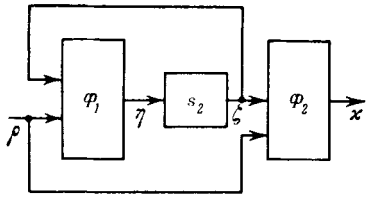


Fig. 4.2.

introduce the function converters Φ_1 and Φ_2, which perform

$$\eta = \Phi_1(\zeta, \rho),$$
$$\varkappa = \Phi_2(\zeta, \rho).$$

respectively. That is, the converter Φ_1 instantaneously and uniquely matches a symbol η of the alphabet $\{\eta\}$ with the symbol pair ζ and ρ from the alphabets $\{\zeta\}$ and $\{\rho\}$, respectively, whereas the converter Φ_2 matches a symbol $\varkappa$ of the alphabet $\{\varkappa\}$ with the above pair.

Let us couple the converters Φ_1 and Φ_2 with the sequential machine s_2 as illustrated in Fig. 4.2. The resulting system is a new s-machine that operates on the same alphabets as machine s_1.

If machines s_1 and s_2 are given, it may be possible to select converters Φ_1 and Φ_2 such that the resulting system of Fig. 4.2 will duplicate any result produced by the machine s_1. If that is possible,

we shall say that *the machine* s_1 *replaces the machine* s_2 or, what is the same thing, that *the machine* s_2 *substitutes for the machine* s_1.

To give a strict definition of these terms, we must first define what we mean by the statement that a system "duplicates any result produced by a given s-machine." We shall agree that a machine s_2 substitutes for a machine s_1 if for each initial state $\varkappa^0$ of s_1 there exists at least one initial state ζ^0 of s_2 such that, for any input sequence of symbols from the alphabet $\{\rho\}$, both the system produced by coupling s_2 to appropriate converters Φ_1 and Φ_2 in the manner of Fig. 4.2 and the machine s_1 will generate the same output sequence of symbols from the alphabet $\{\varkappa\}$, starting from ζ^0 and $\varkappa^0$, respectively. The fact that the machine s_2 can be substituted for the machine s_1 will be indicated by:

$$s_2 \Rightarrow s_1.$$

When we write $s_2 \Rightarrow s_1$, we mean that the machine s_2, appropriately coupled with appropriate converters Φ_1 and Φ_2, can operate in the same way as the machine s_1, thus replacing it. In this sense, the system of Fig. 4.2 is also a sequential machine.

The fact that $s_2 \Rightarrow s_1$ does not necessarily mean that $s_1 \Rightarrow s_2$. Our definition of substitution is based on the complete independence of the choice of converters and initial states from the sequence of input symbols $\{\rho\}$. Naturally, we could have given a broader definition, relating the choice of converters and initial states to the input sequence. However, we are not concerned with such a broad concept (although it may be useful in some problems). We can also introduce the concept of *relative substitution* for an S-machine, if the set L of admissible input sequences of the machine s_1 is restricted.

The idea of substitution immediately involves the following problem: Given two s-machines s_1 and s_2 determine whether s_2 can substitute for s_1, that is, whether there exist function converters Φ_1 and Φ_2 such that the diagram of Fig. 4.2 describes a machine that substitutes for the machine s_1; if the answer is affirmative, construct converters Φ_1 and Φ_2. This problem has a trivial solution — all that is necessary is to test all the (finitely many) pairs of converters Φ_1 and Φ_2. If any of these pairs proves "suitable," then s_2 will be a substitute for s_1. Obviously, this search method is cumbersome and cannot be used in practice. However, the present authors know of no better method.

We shall now leave the generalized system of Fig. 4.2 and shall consider those of its special cases which are shown in Fig. 4.3. Of these, the system of Fig. 4.3,b is extremely important. Here, each

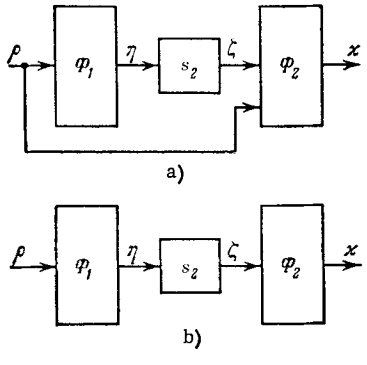


Fig. 4.3.

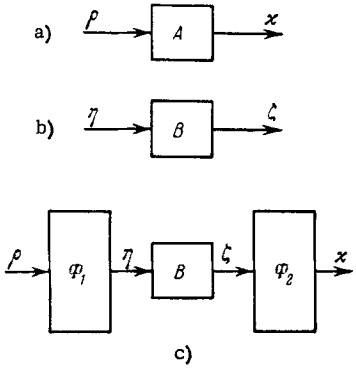


Fig. 4.4.

of the two converters performs functions of a single variable:

$$\eta = \Phi_1(\rho), \quad x = \Phi_2(\zeta).$$

In this special case, the problem formulated above has, in addition to a trivial solution, the following additional solution: the machine s_2 substitutes for the machine s_1 *if the state diagram of* s_1 *is superposable on the state diagram of* s_2 *(that is, is part of it) while preserving the uniqueness of functions* $\Phi_1(\rho)$ *and* $\Phi_2(\zeta)$. We shall illustrate this solution by an example.

Let us introduce the concept of *substitution for finite automata*, which is analogous to that of substitution for the s-machine: all the definitions are retained, except that instead of sequential machines s_1 and s_2, we are given two finite automata A and B (Fig. 4.4,a,b), and the substituting system is that of Fig. 4.4,c which is similar to that of Fig. 4.3,b.

(The above definitions obviously apply also to the special case of autonomous automata. However, the definition is simpler in this case since there are no input sequences and thus there is no need for an input symbol converter.)

As an example of the substitution of finite automata, let automata A and B have the state diagrams of Figs. 4.5 and 4.6, respectively. Then can automaton B, associated in the system of Fig. 4.4, substitute for automaton A? Converter Φ_2 relabels the states of automaton B (that is, it relabels the circles in its state diagram), while converter Φ_1 changes the labels above the arrows in the state diagram. (The usual condition of uniqueness must, of course, also be satisfied in the case of converters Φ_1 and Φ_2, that is, different values of the arguments should not result in the same value of the function.) If

after two such relabeling operations any part of the state diagram of automaton B still coincides with the state diagram of automaton A, then B substitutes for A.

In our example, the diagram of Fig. 4.5 is superposable on that part of Fig. 4.6 which consists of circles ζ_2, ζ_4 and ζ_5 with the associated arrows. Therefore converter $\varkappa = \Phi_2(\zeta)$ is a device for relabeling the states which operate in accordance with Table 4.1.

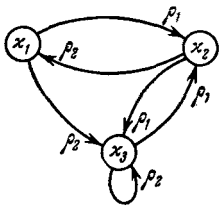


Fig. 4.5.

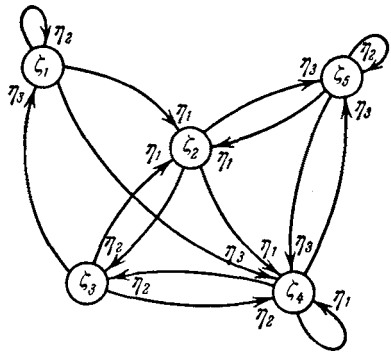


Fig. 4.6.

Table 4.1

ζ	ζ_1	ζ_2	ζ_3	ζ_4	ζ_5
$\varkappa = \Phi_2(\zeta)$		x_1		x_3	x_2

The operation of converter $\Phi_2(\zeta)$ is unspecified in states ζ_1 and ζ_3 of B, since these states do not occur during operation of the system. If desired, the operation of the converter in these states may be specified in an arbitrary manner, for instance, as shown in Table 4.2.

Table 4.2

ζ	ζ_1	ζ_2	ζ_3	ζ_4	ζ_5
$\varkappa = \Phi_2(\zeta)$	x_4	x_1	x_5	x_3	x_2

Converter $\Phi_2(\zeta)$ is now completely defined. It is seen from Table 4.2 that it satisfies the condition of uniqueness; that is, a given ζ uniquely determines $\varkappa$.

Let us now discuss converter $\Phi_1(\rho)$. The circles $\varkappa_1$ and $\varkappa_2$ in the diagram of A (Fig. 4.5) are connected by an arrow bearing the label ρ_1. Table 4.2 specifies that circle $\varkappa_1$ is matched by circle ζ_2 in the state diagram of automaton B (Fig. 4.6), and that circle $\varkappa_2$ of Fig. 4.5 is matched by circle ζ_5 in Fig. 4.6. The diagram of B shows that circles ζ_2 and ζ_5 are connected by an arrow labeled η_3 (that is, automaton B transforms from the state 1 into state 5 upon an input of η_3). Since automaton A transforms from state $\varkappa_1$ to $\varkappa_2$ upon an input of ρ_1, converter Φ_1 must place symbol η_3 into correspondence with symbol ρ_1.

Table 4.3

ρ	ρ_1	ρ_2
$\eta = \Phi_1(\rho)$	η_3	η_1

A similar reasoning may be applied to other portions of automatons A and B, and will finally result in Table 4.3 for converter Φ_1. This means that relationships

$$\Phi_1(\rho_1) = \eta_3, \quad \Phi_1(\rho_2) = \eta_1,$$

hold in every instance; that is, the condition of uniqueness is satisfied for converter Φ_1.

Now, if the labels above the branches of the state diagram of Fig. 4.6 were to be changed in accordance with Table 4.3 and the states were to be renumbered in accordance with Table 4.2, then the diagram of Fig. 4.6 would correspond exactly to a part of the diagram of Fig. 4.5.

To sum up, automaton A may be substituted by automaton B.

Still another variant of the same substitution is given by Tables 4.4 and 4.5 (instead of Tables 4.2 and 4.3).

Table 4.4

ζ	ζ_1	ζ_2	ζ_3	ζ_4	ζ_5
$\varkappa = \Phi_2(\zeta)$	$\varkappa_4$	$\varkappa_1$	$\varkappa_2$	$\varkappa_3$	$\varkappa_5$

Table 4.5

ρ	ρ_1	ρ_2
$\eta = \Phi_1(\rho)$	η_2	η_1

If the diagram of B were that of Fig. 4.7, the diagram of A remained the same (Fig. 4.5), then B could not substitute for A. Indeed, in this case there are only two possible tables for converter Φ_2: 4.6 or 4.7. However, the converter Φ_1 is not unique in these two cases since it is required that $\Phi_1(\rho_1) = \eta_3$ and at the same time that $\Phi_1(\rho_1) = \eta_2$. Substitution is therefore impossible. But this conclusion only holds for the substitution system of Fig. 4.4,c, because

Table 4.6

ζ	ζ_1	ζ_2	ζ_3	ζ_4	ζ_5
$\varkappa = \Phi_2(\zeta)$	$\varkappa_4$	$\varkappa_1$	$\varkappa_5$	$\varkappa_3$	$\varkappa_2$

Table 4.7

ζ	ζ_1	ζ_2	ζ_3	ζ_4	ζ_5
$\varkappa = \Phi_2(\zeta)$	$\varkappa_4$	$\varkappa_1$	$\varkappa_2$	$\varkappa_3$	$\varkappa_5$

such substitution becomes possible with the generalized system of Fig. 4.2. In that case, the converter Φ_2 is described by Table 4.6. The output $\varkappa$ depends on only one input ζ, that is, $\varkappa = \Phi_2(\zeta)$, but the output of converter Φ_1 is a function of two variables, that is, $\eta = \Phi_1(\zeta, \rho)$, and converter Φ_1 is described by Table 4.8.

Table 4.8

ρ \ ζ	ζ_2	ζ_4	ζ_5
ρ_1	η_{13}	η_{12}	η_{13}
ρ_2	η_{11}	η_{11}	η_{11}

Previously, we referred to "superposition of state diagrams." We meant by this not just the coincidence of the circles and arrows, and the labels above them, but also the coincidence of the positions of the labels above the arrows. In other words, we were concerned with substitution of automata and s-machines for automata and s-machines *of the same type.*

We showed in Chapter 3 that an s-machine of the P - Pr type could, as we expressed it, always be *transformed* into an s-machine of the P - P type. In using the term "transformation" we relied on the reader's intuitive grasp of this concept. Now, in the light of the definitions introduced in the present chapter, it is clear that in Chapter 3 we were in fact dealing with the substitution of s-machines of the P - P type for a machine of the P - Pr type.

4.2. THE ABSTRACT STRUCTURE OF THE AUTOMATON

In Chapter 3 the finite automaton was formally defined as an operator "processing" a sequence of symbols ρ into a sequence of symbols $\varkappa$ in such a way that the sequences do not contain contradictory triads. Put in these terms, the abstraction "finite automaton" is represented by the recurrent relationship

$$\left.\begin{aligned} \varkappa^{p+1} &= F(\varkappa^p, \rho^p) \quad \text{for type P - P automata,} \\ \varkappa^{p+1} &= F(\varkappa^p, \rho^{p+1}) \quad \text{for type P - N automata,} \end{aligned}\right\} \tag{4.1}$$

where F is any unique function defined on sets $\{\varkappa\}$ and $\{\rho\}$.

Now let us consider in detail how the description of a finite dynamic system such as an automaton of, for example, the P - P type, can be reduced to a relationship of the form (4.1).

Assume a finite dynamic system that has n generalized coordinates $x_1, x_2, \ldots, x_n$ and is subjected to s external (input) effects $u_1, u_2, \ldots, u_s$. We shall call each such input effect an *input fiber*. At the sampling instants* $0, 1, \ldots, p, \ldots$, each of the generalized coordinates may assume only one of a finite number of values.

Assume coordinate x_i can have only one of k_i values ($i = 1, 2, \ldots, n$). Similarly, let each input effect u_j assume only one of r_j values ($j = 1, 2, \ldots, s$) at these instants. The "motion" in the system is subject to the condition that the value of each coordinate at the instant $p + 1$ must be uniquely determined by the values of all the coordinates x_i ($i = 1, 2, \ldots, n$) and of all the inputs u_j ($j = 1, 2, \ldots, s$) at the instant p. If that is the case, then the motion is described by the system of recurrence relations

$$x_i^{p+1} = f_i\left[x_1^p, x_2^p, \ldots, x_n^p; u_1^p, u_2^p, \ldots, u_s^p\right], \quad i = 1, 2, \ldots, n. \tag{4.2}$$

By introducing an n-dimensional vector $\boldsymbol{x}$ with coordinates $x_1, x_2, \ldots, x_n$, an s-dimensional vector $\boldsymbol{u}$ with coordinates $u_1, u_2, \ldots, u_s$, and a vector-function $\boldsymbol{f}$ with coordinates $f_1, f_2, \ldots, f_n$, relations (4.2) may be represented in vector form:

$$\boldsymbol{x}^{p+1} = \boldsymbol{f}\left[\boldsymbol{x}^p, \boldsymbol{u}^p\right]. \tag{4.3}$$

In the sampling instants, vector $\boldsymbol{x}$ can assume one of the $k = \prod_{i=1}^{n} k_i$ values, and the vector $\boldsymbol{u}$, one of the $r = \prod_{j=1}^{s} r_j$ values. Therefore, by selecting alphabets $\{\varkappa\}$ and $\{\rho\}$ consisting of k and r symbols, respectively, and assigning various symbols $\varkappa$ to the various vectors $\boldsymbol{x}$ and symbols ρ to the vectors $\boldsymbol{u}$, we obtain, instead of relation (4.3), a relation of the form of (4.1), in which there is a specific function F on the right-hand side. This function F is derived from the vector-function $\boldsymbol{f}$ (4.3) and is based on the coding selected for vectors $\boldsymbol{x}$ and $\boldsymbol{u}$.

It is now clear that the recurrence relations (4.2) represent a finite automaton

$$\varkappa^{p+1} = F(\varkappa^p, \rho^p). \tag{4.4}$$

*I.e., discrete moments.

Relations (4.2) illustrate more clearly than (4.4) such important features of a dynamic system as the number of its degrees of freedom, n, as well as the values of each of its generalized coordinates as a function of the state of each input fiber.

Let us agree to call a system of relations such as (4.2) the *abstract structure of the finite automaton* (4.4).

Thus a given abstract structure uniquely defines the corresponding finite automaton; that is, relation (4.4) may be uniquely reproduced from (4.2). In this sense (4.2) defines the automaton just as completely as (4.4) does. Accordingly, the concept of substitution of automata applies fully to machines defined by relations of type (4.2).

We shall now show that given a finite automaton A described by relation of type (4.4), we can specify a great number of abstract structures which can substitute for this automaton.

Let automaton A be associated with alphabets $\{\varkappa\}$ and $\{\rho\}$, and let k and r be given. We shall now select numbers n, s, k_i $(i = 1, 2, \ldots, n)$ and r_j $(j = 1, 2, \ldots, s)$. The selection of these numbers is restricted by only one condition: satisfaction of the inequalities

$$k \leqslant \prod_{i=1}^{n} k_i, \quad r \leqslant \prod_{j=1}^{s} r_j. \tag{4.5}$$

We now introduce n coordinates $x_1, x_2, \ldots, x_n$ (which assume $k_1, k_2, \ldots, k_n$ values, respectively) and s input fibers $u_1, u_2, \ldots, u_s$ (which assume $r_1, r_2, \ldots, r_s$ values, respectively). We now complete a table (Table 4.9) in the following manner.

Table 4.9

x^p				u^p				$\varkappa^p$	ρ^p	$\varkappa^{p+1}$	x^{p+1}			
x_1	x_2	...	x_n	u_1	u_2	...	u_s				x_1	x_2	...	x_n

We enter all the possible combinations of values $x_1, x_2, \ldots, x_n$ and $u_1, u_2, \ldots, u_s$ into the left-hand columns x^p, and u^p of the table. The number of such combinations is $\prod_{i=1}^{n} k_i \prod_{j=1}^{s} r_j$, and therefore Table 4.9 shall contain this number of rows.

In order to fill in column $\varkappa^p$ let us concentrate exclusively on columns x^p. We enter in these columns $\prod_{i=1}^{n} k_i$ different combinations of $x_1, x_2, \ldots, x_n$. Let us select at random k of these combinations, which we shall call the basic combinations, assign to them symbols $\varkappa_1, \varkappa_2, \ldots, \varkappa_k$, and enter these symbols into the corresponding rows of the $\varkappa^p$ column of Table 4.9. By virtue of inequalities (4.5) it is possible that $\prod_{i=1}^{n} k_i > k$; therefore some of the rows in column $\varkappa^p$ may remain blank. If that is the case, we assign to those combinations of $x_1, x_2, \ldots, x_n$ which were not included in the k selected combinations the used symbols $\varkappa$ (the order of assignment is immaterial), and enter these symbols into the blank rows of column $\varkappa^p$. Now column $\varkappa^p$ is completely filled in. We fill column ρ^p in a similar manner, using combinations of $u_1, u_2, \ldots, u_s$ entered in columns u^p.

At the end of this procedure, columns $\varkappa^p$, ρ^p contain all the possible combinations of the symbols $\varkappa$, ρ, but since the total number of such combinations is only

$$kr \leqslant \prod_{i=1}^{n} k_i \prod_{j=1}^{s} r_j,$$

some combinations of $\varkappa$, ρ may recur.

We now return to our automaton A. Using one of its definitions, for example, its basic table, we fill in column $\varkappa^{p+1}$ of Table 4.9. But we have already associated one of the basic combinations of $x_1, x_2, \ldots, x_n$ with each symbol $\varkappa$. We therefore enter in columns x^{p+1} the combinations corresponding to $\varkappa^{p+1}$, thus completing the table. This table defines the values of all the x_m^{p+1} starting from the given x_i^p and u_j^p, that is, it defines n functions f_i in recurrence relations (4.2).

If inequalities (4.5) were to be replaced by equations

$$k = \prod_{i=1}^{n} k_i, \quad r = \prod_{j=1}^{s} r_j, \tag{4.6}$$

then each pairwise combination of symbols $\varkappa$, ρ would be encountered only once in columns $\varkappa^p$, ρ^p of Table 4.9. If, however,

$$k < \prod_{i=1}^{n} k_i, \quad r = \prod_{j=1}^{s} r_j, \tag{4.7}$$

then Table 4.9 would contain the same rows as in the case in which

(4.6) holds, and, in addition, some supplementary rows, corresponding to the nonbasic combinations.

If we now attempted to derive an automaton from the abstract structure of Table 4.9, we would obtain an automaton differing from the starting one. The state diagram of such a new automaton would contain all the circles and branches of the diagram of the starting machine (these would be defined by the rows containing the basic combinations of x and u), but it would also contain supplementary circles and branches (corresponding to the nonbasic combinations of x and u). Since the conditions requiring unique operation of symbol converters are satisfied (by the very construction), then, assuming case (4.7) holds, the abstract structure defines an automaton that substitutes for the given (starting) automaton.

When we set up the abstract structure, that is, constructed the system of relations (4.2) from relation (4.4) with the aid of Table 4.9, we had no restriction on the selection of numbers n, s, k_i and r_j, provided conditions (4.5) were satisfied. It is obvious now that not only the form of the functions f_i on the right-hand side of (4.2) but also the number of relations involved in that system depend on how these numbers have been selected. And it is because we have this freedom that we can construct a large number of abstract structures, all which substitute for a given finite automaton A.

An important special case is one in which all the k_i and r_j equal two, that is, all the x and u are logical variables. The abstract structure in this case is

Table 4.10

χ \ p	p_1	p_2	p_3
χ_1	χ_3	χ_4	χ_6
χ_2	χ_2	χ_5	χ_4
χ_3	χ_6	χ_2	χ_3
χ_4	χ_5	χ_1	χ_5
χ_5	χ_3	χ_4	χ_2
χ_6	χ_1	χ_3	χ_2

$$x_i^{p+1} = L_i\left[x_1^p, x_2^p, \ldots, x_n^p; u_1^p, u_2^p, \ldots, u_s^p\right], \quad i = 1, 2, \ldots, n, \tag{4.8}$$

where all the L_i are logical functions. We shall call such an abstract structure ***logical*** or ***binary***. In this case,

$$\prod_{i=1}^{n} k_i = 2^n \text{ and } \prod_{j=1}^{s} r_j = 2^s. \tag{4.9}$$

If k and r are not integral powers of 2 (that is, they are not among

numbers 2, 4, 8, 16, 32, 64, 128, 256, ...), then Eq. (4.6) cannot be satisfied. To satisfy inequalities (4.5), n and s must be selected in accordance with the conditions

$$k < 2^n \text{ and } r < 2^s.$$

Thus, for example, if the automation is defined (given) by its basic table (Table 4.10), then $k = 6$, $r = 3$, and we can select, for example, $n = 3$, $s = 2$. The completion of a table such as Table 4.9 for this case is illustrated in Table 4.11.

Table 4.11

x^{p-1}			u^{p-1}		$\varkappa^{p-1}$	ρ^{p-1}	$\varkappa^p$	x^p		
x_1	x_2	x_3	u_1	u_2				x_1	x_2	x_3
0	0	0	0	0	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0
0	0	1	0	0	$\varkappa_2$	ρ_1	$\varkappa_2$	0	0	1
0	1	0	0	0	$\varkappa_3$	ρ_1	$\varkappa_6$	1	0	1
0	1	1	0	0	$\varkappa_4$	ρ_1	$\varkappa_5$	1	0	0
1	0	0	0	0	$\varkappa_5$	ρ_1	$\varkappa_3$	0	1	0
1	0	1	0	0	$\varkappa_6$	ρ_1	$\varkappa_1$	0	0	0
1	1	0	0	0	$\varkappa_1$	ρ_2	$\varkappa_4$	0	1	1
1	1	1	0	0	$\varkappa_2$	ρ_2	$\varkappa_5$	1	0	0
0	0	0	0	1	$\varkappa_3$	ρ_2	$\varkappa_2$	0	0	1
0	0	1	0	1	$\varkappa_4$	ρ_2	$\varkappa_1$	0	0	0
0	1	0	0	1	$\varkappa_5$	ρ_2	$\varkappa_1$	0	0	0
0	1	1	0	1	$\varkappa_6$	ρ_2	$\varkappa_3$	0	1	0
1	0	0	0	1	$\varkappa_1$	ρ_3	$\varkappa_6$	1	0	1
1	0	1	0	1	$\varkappa_2$	ρ_3	$\varkappa_4$	0	1	1
1	1	0	0	1	$\varkappa_3$	ρ_3	$\varkappa_3$	0	1	0
1	1	1	0	1	$\varkappa_4$	ρ_3	$\varkappa_5$	1	0	0
0	0	0	1	0	$\varkappa_5$	ρ_3	$\varkappa_2$	0	0	1
0	0	1	1	0	$\varkappa_6$	ρ_3	$\varkappa_2$	0	0	1
0	1	0	1	0	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0
0	1	1	1	0	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0
1	0	0	1	0	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0
1	0	1	1	0	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0
1	1	0	1	0	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0
1	1	1	1	0	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0
0	0	0	1	1	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0
0	0	1	1	1	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0
0	1	0	1	1	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0
0	1	1	1	1	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0
1	0	0	1	1	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0
1	0	1	1	1	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0
1	1	0	1	1	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0
1	1	1	1	1	$\varkappa_1$	ρ_1	$\varkappa_3$	0	1	0

We can also develop other binary structures that substitute for the same automaton; thus, for instance, we can take $n = 4$, $s = 3$, or $n = 3$, $s = 3$, and so on.

Table 4.12

x^p		u^p	$\varkappa^p$	ρ^p	$\varkappa^{p+1}$	x^{p+1}	
x_1	x_2					x_1	x_2
0	0	0	$\varkappa_1$	ρ_1	$\varkappa_3$	0	2
0	1	0	$\varkappa_2$	ρ_1	$\varkappa_2$	0	1
0	2	0	$\varkappa_3$	ρ_1	$\varkappa_6$	1	2
1	0	0	$\varkappa_4$	ρ_1	$\varkappa_5$	1	1
1	1	0	$\varkappa_5$	ρ_1	$\varkappa_3$	0	2
1	2	0	$\varkappa_6$	ρ_1	$\varkappa_1$	0	0
0	0	1	$\varkappa_1$	ρ_2	$\varkappa_4$	1	0
0	1	1	$\varkappa_2$	ρ_2	$\varkappa_5$	1	1
0	2	1	$\varkappa_3$	ρ_2	$\varkappa_2$	0	1
1	0	1	$\varkappa_4$	ρ_2	$\varkappa_1$	0	0
1	1	1	$\varkappa_5$	ρ_2	$\varkappa_1$	0	0
1	2	1	$\varkappa_6$	ρ_2	$\varkappa_3$	0	2
0	0	2	$\varkappa_1$	ρ_3	$\varkappa_6$	1	2
0	1	2	$\varkappa_2$	ρ_3	$\varkappa_4$	1	0
0	2	2	$\varkappa_3$	ρ_3	$\varkappa_3$	0	2
1	0	2	$\varkappa_4$	ρ_3	$\varkappa_5$	1	1
1	1	2	$\varkappa_5$	ρ_3	$\varkappa_2$	0	1
1	2	2	$\varkappa_6$	ρ_3	$\varkappa_2$	0	1

We shall now show that the same automaton could be replaced by an abstract structure that is not binary. Suppose, for example, that $n = 2$, $k_1 = 2$, $k_2 = 3$, but that $s = 1$, $r_1 = 3$ as before. Then $k = k_1 k_2 = 6$, and $r = r_1 = 3$; that is, Eqs. (4.6) are satisfied.

Completing Table 4.9 for this case, we get an abstract structure of Table 4.12, but this structure is no longer binary.

4.3. NETS

Suppose we have the simplest finite automaton, for which Eq. (4.1) becomes

$$\varkappa^{p+1} = \rho^p. \tag{4.10}$$

In analyzing this automaton we assume that even if the alphabet $\{\varkappa\}$ contains some symbols that are not contained in $\{\rho\}$, these shall appear only at the initial instant. The symbol $\varkappa$ produced by such an automaton at the instant p is identical to the input symbol in the preceding instant $p - 1$. We shall call this simple automaton a *one-instant delay* (or simply *delay*).

Returning to our P - P automaton and introducing a new variable μ (alphabet $\{\mu\}$ coincides with alphabet $\{\varkappa\}$), and replace (4.1) by relations

$$\left.\begin{aligned} \mu^p &= F[\varkappa^p, \rho^p], \\ \varkappa^p &= \mu^{p-1}. \end{aligned}\right\} \tag{4.11}$$

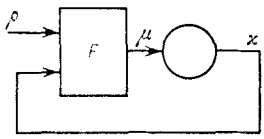


Fig. 4.8.

Such a relationship was already discussed in Section 3.3. We shall now treat the first of relations (4.11) as representing an instantaneous converter of the symbols $\varkappa$ and ρ into symbols μ, while the second relation defines a delay. Accordingly, relations (4.11) may be represented by the system of Fig. 4.8, where the delay is shown as a circle.

Let us now consider the abstract structure of some automaton, for instance, of a P - P automaton, that is, a system of relations of the form (4.2):

$$\begin{gathered} x_i^{p+1} = f_i[x_1^p, x_2^p, \ldots, x_n^p; u_1^p, u_2^p, \ldots, u_s^p], \\ i = 1, 2, \ldots, n. \end{gathered} \tag{4.12}$$

Since each relation of this system, for example,

$$x_1^{p+1} = f_1[x_1^p, x_2^p, \ldots, x_n^p; u_1^p, u_2^p, \ldots, u_s^p],$$

may in itself be treated as combining the delay

$$x_1^{p+1} = y_1^p$$

and the function converter

$$y_1 = f_1[x_1, x_2, \ldots, x_n; u_1, u_2, \ldots, u_s] \tag{4.13}$$

the entire abstract structure (4.12) may be represented by n delays and n function converters. The input of each delay is connected to the output of the corresponding converter. The outputs of all the delays are connected to the inputs of each of the converters. In addition, we feed external effects u to the inputs of all the converters (Fig. 4.9).*

If we have several abstract structures, we can derive new abstract structures by "interconnecting" the original ones.

*In Fig. 4.9 and henceforth, circles represent delays and rectangles denote converters.

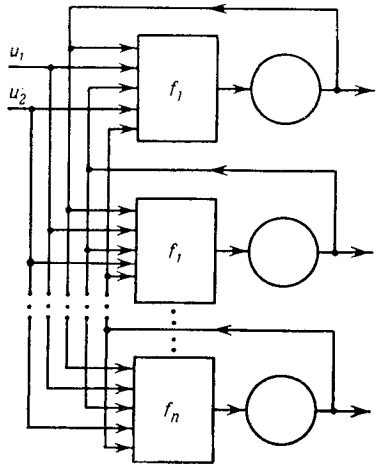


Fig. 4.9.

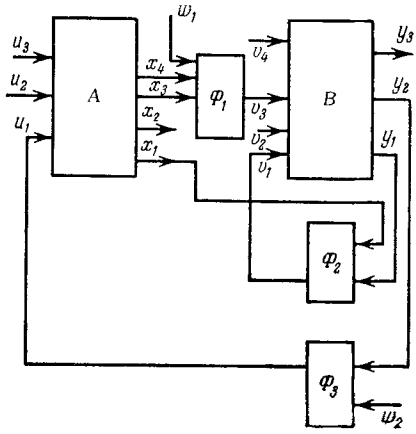


Fig. 4.10.

By "interconnection of automata" we mean the identifying of the output symbols of one automaton with the input symbols of another automaton. In this sense the output of one automaton can act upon the input of another one only if all the symbols of the output alphabet of the first automaton are contained in the input of the second. If this is not the case then "interconnection" of automata can be achieved only by means of auxiliary converters.

For example, suppose we have two abstract structures: structure A

$$x_i^{p+1} = f_i(x_1^p, x_2^p, x_3^p, x_4^p; u_1^p, u_2^p, u_3^p), \quad i = 1, 2, 3, 4 \tag{4.14}$$

and structure B

$$y_j^{p+1} = \varphi_j(y_1^p, y_2^p, y_3^p, v_1^p, v_2^p, v_3^p, v_4^p), \quad j = 1, 2, 3. \tag{4.15}$$

Let us supplement (4.14) and (4.15) with the equations of three converters:

$$\left.\begin{aligned} v_3 &= \Phi_1(x_3, x_4, w_1), \\ v_1 &= \Phi_2(y_1, x_1), \\ u_1 &= \Phi_3(y_2, w_2). \end{aligned}\right\} \tag{4.16}$$

Here w_1 and w_2 are auxiliary input fibers of the converters. Figure 4.10 shows schematically the coupling of structures A and B by means of these three converters.

Together, Eqs. (4.14), (4.15), and (4.16) again give an abstract structure. Indeed, substituting (4.16) into (4.14) and (4.15), we obtain a system of seven recurrence relations with coordinates (x_1, x_2, x_3, x_4, y_1, y_2 and y_3) and input fibers (u_2, u_3, v_2, w_1 and w_2), of the same type which we already know as an "abstract structure."

We shall designate by the term *net* a system of finite number of recurrence relations similar to Eq. (4.2) and supplemented by converter equations which express some of the inputs by means of coordinates.

The net itself is an abstract structure. Its coordinates are all the generalized coordinates of all its component abstract structures. The input fibers of the net can be both the input fibers of the component abstract structures that are not acted upon by converters, as well as the input fibers of the converters that are not acted upon by the coordinates of the abstract structures constituting the net.

To obtain a system of relations such as (4.2) for the net, one uses the converter equations to eliminate the input variables acted upon by the converters.

Since the net itself is an abstract structure, it substitutes for some finite automaton. Thus when one uses converters to combine abstract structures into nets, one generates new finite automata from other finite automata.

We shall say that a net is a *delay net* if it consists only of delays connected by means of function converters. It follows from previous discussion that any abstract structure and any net can be represented by a delay net. This was shown in Fig. 4.9. Such a representation is not unique in the sense that every delay net may substitute for some automaton, and, as was pointed out in the preceding section, one can generate many abstract structures which can substitute for each automaton. This means that one can construct many

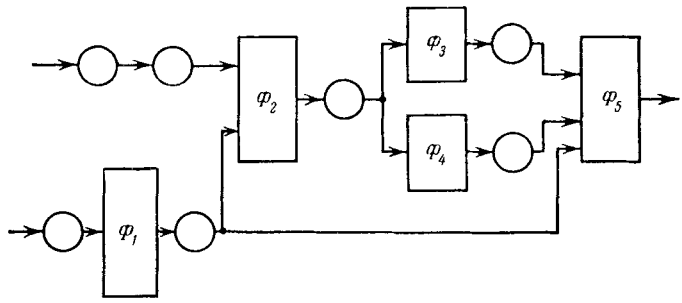


Fig. 4.11.

delay nets to represent the same automaton, such nets differing in the number of constituent delays and in the alphabets on which the delays are defined.

Among these delay nets we can distinguish the subclass of *loop-free nets*. A delay net is said to be a loopfree net if, starting from any delay, one moves along the net in the direction of its operation (along the arrows of the schematic such as Fig. 4.5) and never returns to the starting delay. Figure 4.11 shows a loopfree net, while Fig. 4.12 presents a net containing loops.

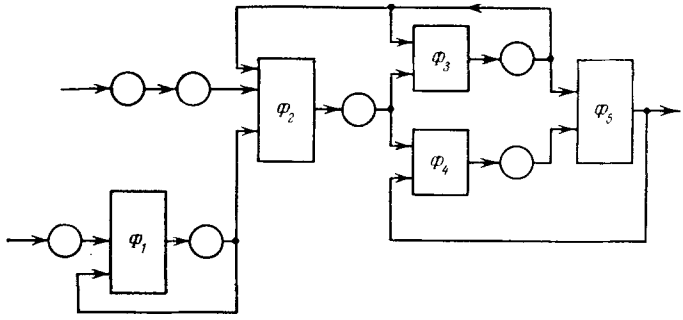


Fig. 4.12.

One important loop-free net is that consisting of q series-connected delays (Fig. 4.13). If the inputs and outputs of the delays are defined on different alphabets, then one must interpose converters between the delays (Fig. 4.14).

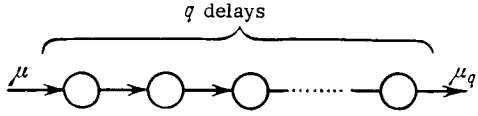


Fig. 4.13.

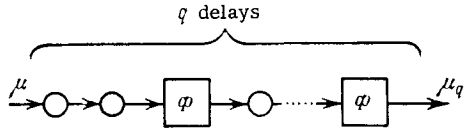


Fig. 4.14.

Let us call such a net a *delay line*. As any other net, a delay line is a finite automaton, but with an important difference. If the delay line has an input μ and an output μ_q, then the symbol μ_q will not be a function of the symbol μ generated during the preceding sampling instant, but will be defined by the input symbol μ appearing q instants before:

$$\mu_q^{p+q} = f(\mu^p).$$

If the input and output symbols of all the delays are defined on the same alphabet, then output μ_q will coincide with input μ, supplied q instants before:

$$\mu_q^{p+q} = \mu^p.$$

Let us now return to the diagram of Fig. 4.8, but let us substitute its delay by a delay line. We then obtain Fig. 4.15. Then, instead of relations (4.11), we have

$$\left.\begin{aligned} \mu^p &= F(x^p, \rho^p), \\ x^{p+q} &= \mu^p. \end{aligned}\right\} \tag{4.17}$$

Eliminating μ, we have

$$x^{p+q} = F(x^p, \rho^p), \tag{4.18}$$

whereas by eliminating x from (4.17) [rather than μ], we obtain

$$\mu^{p+q} = F(\mu^p, \rho^{p+q}). \tag{4.19}$$

A delay line can also be connected to the input of a converter. Then, instead of relations (4.17), we get

$$\left.\begin{aligned} \mu^p &= F(x^p, \psi^p), \\ \psi^{p+q} &= \rho^p, \\ x^{p+q} &= \mu^p, \end{aligned}\right\} \tag{4.20}$$

or, by eliminating ψ and x,

$$\mu^{p+q} = F(\mu^p, \rho^p). \tag{4.21}$$

Note that if we had designated by symbols x the state of the entire system, taking into account the outputs of all the delays, then

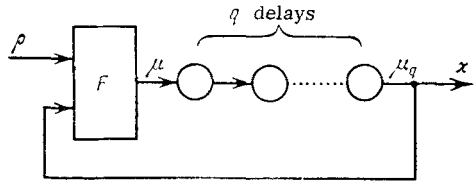


Fig. 4.15.

at the instant $p + 1$ the $\varkappa$ would depend only on the $\varkappa$ at the instant p. However, if we assign the symbol $\varkappa$ only to the output of the last delay, we can use this automaton to embody relation (4.19). This is precisely what we meant when, on introducing the concept of a finite automaton in Chapter 3, we said that an expression such as (3.5′):

$$\varkappa^{p+1} = F(\varkappa^p, \rho^p)$$

is so general that it includes a finite automaton in which the output $\varkappa^{p+q}$ depends on the inputs $\varkappa^p$ and ρ^p introduces a finite number (q) of instants previously.

Now let us return to the relationship between automata and sequential machines of the P - P and P - Pr types. Consider the schematic of Fig. 4.8, described by Eq. (4.11). As stated in Chapter 3, Eq. (4.11) describes automata of either the P - P or P - Pr type, depending on whether we eliminate variable μ or variable $\varkappa$. The diagram of Fig. 4.8 can represent either case, depending on whether the output variable is $\varkappa$ (P - P automaton) or μ (P - Pr automaton). Figures 4.16,a and b shows these two machines.

As pointed out in Chapter 3, a P - P s-machine differs from P - Pr machine only in that it contains a P - P rather than a P - Pr automaton. Thus such machines may be represented as shown in Fig. 4.17,a,b.

Let us recall that the P - P machine can always substitute for a P - Pr machine simply by changing converter Φ. However, the converse statement, that is, that merely by exchanging the output converter a P - Pr machine can be made to substitute for a P - P machine, is not true. Now that we can define automata by means of tables and have introduced diagrams such as Figs. 4.16 and 4.17, we can illustrate this statement by an example.

Assume an s-machine of the P - P type (Fig. 4.17,a) is given by the tables of converter F (Table 4.13) and converter Φ (Table 4.14).

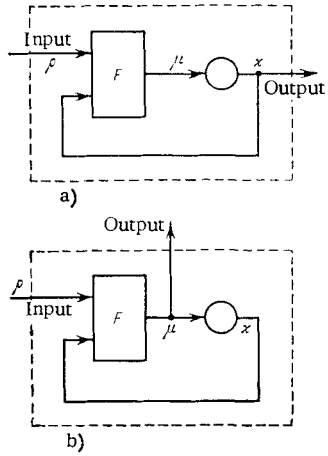


Fig. 4.16.

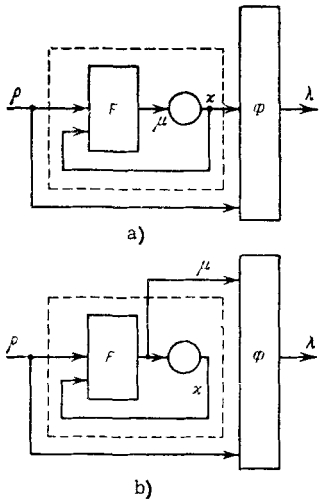


Fig. 4.17.

Table 4.13

Converter F

x \ ρ	ρ_1	ρ_2	ρ_3
x_1	x_3	x_1	x_2
x_2	x_3	x_3	x_4
x_3	x_2	x_1	x_2
x_4	x_2	x_4	x_3

Table 4.14

Converter Φ

x \ ρ	ρ_1	ρ_2	ρ_3
x_1	λ_2	λ_3	λ_2
x_2	λ_3	λ_1	λ_3
x_3	λ_2	λ_1	λ_2
x_4	λ_1	λ_2	λ_3

Retaining converter F as is, find a new table for a converter $\lambda = \Phi^*(\mu, \rho)$ such that the s-machine Fig. 4.17, b incorporating it will substitute for the initial s-machine.

Consider, for example, the intersection of the first row and the first column in the table for converter Φ (Table 4.14), where one finds symbol λ_2. The corresponding square of Table 4.13 for converter F contains the symbol x_3. We shall thus write the symbol λ_2 at the intersection of the first column and the third row in the table of the new converter Φ^* (Table 4.15): we can complete the remainder of this table in the same manner.

Now, however, let us try to repeat the procedure, this time starting with the square located in the first column and the second row of Table 4.14 of the old converter Φ. This square contains symbol λ_3, and the corresponding square of Table 4.13 contains the symbol $\varkappa_3$ as before.

Thus our new table would have to contain λ_3 in a square already occupied by symbol λ_2. But this means that we would obtain a nonunique converter. Our example confirms the statement that, in general, a P - Pr machine cannot substitute for a P - P machine if the only change introduced into the P - Pr machine is that of the output converter.

Table 4.15

Converter Φ^*

$\varkappa$ \ ρ	ρ_1	ρ_2	ρ_3
$\varkappa_1$			
$\varkappa_2$			
$\varkappa_3$	λ_2		
$\varkappa$			

Now let us return to the relationship between a finite automaton and a sequential machine. We have accumulated sufficient material to define this relationship more precisely, by means of the following theorem.*

Theorem. *For every sequential machine s there exists a system consisting of a finite automaton A and an output converter such that, given any initial state of S and any input sequence to it, there is an initial state of A such that, at all $p \geqslant 1$, the output sequence of A is a repetition of the output sequence of S with a delay of one sampling instant. Conversely, for any system consisting of a finite automaton A and an output converter, there exists a sequential machine s such that, given an initial state of A and any input sequence to it, there is an initial state of S such that, at all $p \geqslant 0$ the output sequence of S repeats the output sequence of A with a lead of one sampling instant.*

Proof of the first statement. Assume a sequential machine s described by equations

$$\varkappa^{p+1} = F(\varkappa^p, \rho^p), \tag{4.22}$$

$$\lambda^p = \Phi(\varkappa^p, \rho^p). \tag{4.23}$$

We shall now construct a net consisting of two finite automata and define it by equations

$$y^{p+1} = F_1(y^p, z^p), \tag{4.24}$$

$$z^{p+1} = \rho^p, \tag{4.25}$$

*Compare this with Theorem 1 of [16].

and we define the output converter of the net by

$$\chi^p = \Phi(y^p, z^p), \tag{4.26}$$

where the alphabets $\{y\}$ and $\{\chi\}$ coincide with the alphabets $\{\varkappa\}$ and $\{\lambda\}$, respectively; the alphabet $\{z\}$ differs from $\{\rho\}$ by one additional symbol z_0; the function Φ in (4.26) coincides with the corresponding function in (4.23) for all pairs of symbols from alphabets $\{\varkappa\}$ and $\{\rho\}$, but is indeterminate (or may be defined in any desired way) for $z = z_0$; the function F_1 in (4.24) coincides with F in (4.22) for all combinations of symbols that do not contain z_0, but for z_0-containing combinations, $F_1(y, z_0) = y$. Equations (4.24) and (4.25) specify the finite automaton A whose states are coded by the symbol pair (y, z), while Eq. (4.26) defines the output converter for this automaton. We shall now compare with each initial state $\varkappa^0$ of S that initial state of automaton A for which

$$y^0 = \varkappa^0, \quad z^0 = z_0. \tag{4.27}$$

When $p = 0$, it follows from (4.24) that $y^1 = y^0 = \varkappa^0$. When $p \geqslant 1$, the symbol z^0 cannot be generated, so that for the sampling instants the function F_1 in (4.24) can be replaced by F, and

$$y^{p+1} = F(y^p, z^p), \quad p \geqslant 1. \tag{4.28}$$

Introducing a new variable $Y^p \equiv y^{(p+1)}$ $(p \geqslant 0)$, and using (4.24), we can write

$$Y^{p+1} = F(Y^p, \rho^p), \quad p \geqslant 0, \tag{4.29}$$

whereby $Y^0 = y^1 = \varkappa^0$. Equations (4.22) and (4.29) and the initial conditions $\varkappa(0) = \varkappa^0$ coincide, so that for any $p \geqslant 1$

$$Y^p \equiv \varkappa^p.$$

But $y^p = Y^{p-1}$, so that

$$y^p = \varkappa^{p-1}, \quad p \geqslant 1. \tag{4.30}$$

Substituting (4.25) and (4.30) into (4.26) and comparing with (4.23), we get

$$\chi^p = \Phi(\varkappa^p, \rho^p) = \lambda^{p-1}, \quad p \geqslant 1,$$

which proves the first statement of the theorem.

We shall now prove the second statement. Let the finite automaton A be

$$z^{p+1} = F(z^p, \rho^p) \tag{4.31}$$

and its output converter

$$\chi^p = \Phi(z^p). \tag{4.32}$$

We shall now compare this automaton with the sequential machine s:

$$y^{p+1} = F(y^p, \rho^p), \tag{4.33}$$

$$\lambda^p = \Phi[F(y^p, \rho^p)], \tag{4.34}$$

where the alphabets $\{y\}$ and $\{z\}$ coincide. We shall match each initial state of A with an initial state of S. Since Eqs. (4.31) and (4.33) and the corresponding initial conditions coincide, we have

$$y^p = z^p \tag{4.35}$$

for all $p \geqslant 0$. Substituting (4.35) into (4.34) and using (4.31) and (4.32), we have, for all $p \geqslant 0$,

$$\lambda^p = \Phi[F(z^p, \rho^p)] = \Phi(z^{p+1}) = \chi^{p+1},$$

q. e. d. Thus the entire theorem is proved.

If one wants to determine what a sequential machine or a finite automaton "can do," then, by virtue of the above theorem, one need not examine these machines separately. However, the two abstract concepts are not equivalent, in the sense that the same "task" may be performed in a sequential machine with a smaller number of states than in the corresponding automaton. This is important in cases in which it is desired to minimize the number of such states (see Chapter 9).

4.4. ABSTRACT AGGREGATES OF AUTOMATA AND SEQUENTIAL MACHINES

We have proved that an automaton may be replaced by a variety of abstract structures, and that each abstract structure may itself be replaced by a great variety of nets. On the other hand, if each abstract structure is an automaton, then the very concept of a net

permits us to devise new automata from other automata. This reasoning leads to the following problem: is it possible to find such a set of automata and converters that, by employing any devised number of automata and converters of this set, one can construct nets which would substitute for a variety of automata and sequential machines?

The process of constructing nets by employing only automata and converters of a given set will be called the *aggregation of finite automata and sequential machines.*

Automata and converters contained in a set are said to be *elements of the set.* We shall say that a set is *complete* if its elements can be used to construct a net which can substitute for any *a priori* given finite automaton or sequential machine.

We have already shown that, given a set of any delay elements (that is, operating in any alphabets) and any converters, one can construct a net which will substitute for any given automaton.

Assume a delay element operating in alphabet $\{\mu\}$. If, in addition, we had some set of elementary converters from which we could construct any converter operating in alphabet $\{\mu\}$, then we would have a complete set.

Thus, for example, one important complete set is that consisting of a binary delay element (that is, a bistable delay element) plus elements performing the operations of disjunction, conjunction, and negation.

Indeed, any automaton may be replaced by an abstract logical structure, that is, an abstract structure in which all the $k_i = 2$ and all the $r_j = 2$ $(i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, s)$. But such an abstract structure may be represented by a net consisting exclusively of binary delay elements and logical converters. But since any logical function may be represented by a conjunction of disjunctive groups, any logical converter can be constructed of converters performing disjunction, conjunction, and negation (see Chapter 1 and 2). Therefore, the above set is complete.

Obviously, we would also have a complete set if the latter consisted of the binary delay element plus a converter performing any logical function (such as a converter performing the Sheffer stroke).

In a similar manner, one can develop complete sets operating in alphabets containing more than two symbols (for instance, m symbols); the problem then reduces to that of expressing any logical function of m-valued logic by means of several primitive functions. Such primitive functions, plus a delay element operating in the alphabet of m symbols, constitute a complete set.

Chapter 5 will describe the practical embodiments of various complete sets as well as the construction of finite automata and

sequential machines based on such sets. However, let us first briefly consider an important abstract model, which was developed in connection with certain problems in physiology.

4.5. ABSTRACT NEURONS AND MODELS OF NEURAL NETS

The behavior of nerve cells (neurons) and of nervous systems is assumed to be representable by abstract (idealized) neurons and abstract models of neural nets. The McCulloch - Pitts neural net is one of the best known abstractions of this kind.*

The *McCulloch-Pitts neuron* is an imaginary logical element which may exist in only one of two possible states:

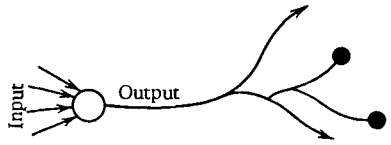


Fig. 4.18.

"stimulated" and "not stimulated." A neuron may have any finite number of inputs, but only one output which may, however, have any finite number of branches (Fig. 4.18). Each input may terminate in either of two endings: *"inhibitory"* (black dot in Fig. 4.18) or *"simple"* (arrow in Fig. 4.18). The branching output endings of a neuron may act upon the inputs of other neurons or on their own input. Some of the neuronal inputs may be externally stimulated. Again, an external stimulus may either stimulate or not stimulate the input. We can thus form *abstract neural nets* to simulate nerve tissues (see Fig. 4.19).

Let $f(t)$ be the number of simple input ("$\rightarrow$") endings which are stimulated at the instant t and act upon a given neuron and let $g(t)$ be the number of stimulated inhibitory endings ($\bullet$) which also act upon this neuron.

The functioning of a neuron (and, consequently, of the net) is determined by the following condition of stimulation: A neuron is stimulated at the instant $t+\tau$ if the following conditions are satisfied at the instant t:

$$g(t)=0, \quad f(t)\geqslant h, \tag{4.36}$$

*Developed in 1943 (see [62]), this abstraction has by now lost its value to physiology because of more recent studies in the properties of neurons.

where h is a given finite number, called the *threshold of stimulation.* A neuron is not stimulated unless these conditions are met. Thus the inhibitory ending has "veto power"; that is, even if inequality (4.36) is satisfied, the output is not stimulated if the input of the neuron in question is connected to even one inhibitory branch of a stimulated neuron.*

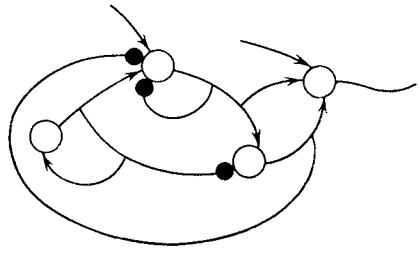


Fig. 4.19.

If w_i is the state of the simple input fibers ($i = 1, 2, \ldots, s$), v_j is the state of the inhibitory input fibers ($j = 1, 2, \ldots, q$), and x is the state of the neuron, then the behavior of a McCulloch-Pitts neuron, such that $s = 3$, $h = 2$, $q = 2$ is described by

$$x(t+\tau) = \{[w_1(t) \& w_2(t)] \vee [w_1(t) \& w_3(t)] \vee [w_2(t) \& w_3(t)]\} \& \overline{v_1(t)} \& \overline{v_2(t)}. \tag{4.37}$$

Let us mark off points $0, \tau, 2\tau, 3\tau, \ldots$ on the time axis and observe the neurons and neural nets only at such instants; that is, let us introduce sampling instants. Instead of $0, \tau, 2\tau, 3\tau, \ldots$ we introduce integers $0, 1, 2, \ldots, p \ldots$ respectively, to denote the occurrence of these sampling instants. Then expression (4.37) may be written as

$$x^{p+1} = [(w_1^p \& w_2^p) \vee (w_1^p \& w_3^p) \vee (w_2^p \& w_3^p)] \& \overline{v_1^p} \& \overline{v_2^p}. \tag{4.38}$$

Now consider any McCulloch-Pitts abstract neural net consisting of n neurons and having s "free" inputs, through which external effects can be introduced. We shall label the neurons contained in the net by $1, 2, \ldots, n$ and the free input fibers by $1, 2, \ldots, s$; we shall also denote by x_i the state of the ith neuron and by u_j the state of the jth input fiber of the net. Then we can write n equations of the form (4.38). Each input of a neuron of the net is acted upon either

*Frequently, other conditions for the functioning of neurons are specified (see, for example, [73]).

by the ending of one of the other neurons or by an external effect. Therefore each w or v can be identified with an x or a u; that is, we can introduce x^p or u^p into the right-hand sides of expressions (4.38), to replace w^p and v^p, respectively.

It follows from the foregoing that the McCulloch-Pitts abstract neural net can be described by the system of relations

$$x_i^{p+1} = L_i(x_1^p, x_2^p, \ldots, x_n^p; u_1^p, u_2^p, \ldots, u_s^p), \quad i = 1, 2, \ldots, n, \tag{4.39}$$

where L_i are logical functions such as (4.38).

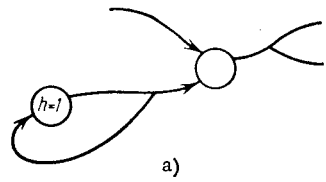
Thus, the McCulloch-Pitts abstract neural net is in effect a net according to our definition of this term, and is therefore a finite automaton. But since the right-hand side of (4.39) contains not just any logical functions but functions of a special type [the (4.38) type], there arises a question: Can we construct a neural net to correspond to any given finite automaton operating in the 0, 1, 2, ... (that is, 0, τ, 2τ, ...) timing sequence?

To answer this question, it will be pointed out, first of all, that a self-simulated neuron with $h = 1$ and no inhibiting inputs (Fig. 4.20) is a blocked, or permanently simulated, neuron. Therefore neurons may have permanently stimulated inputs (Fig. 4.21,a), denoted as shown in Fig. 4.21,b. We shall say in this case that the stimulated input fiber is *fixed.*

We shall now consider a neuron with $h = 1$, $s = 1$, and $q = 0$ (Fig. 4.22). Such a neuron is described by

$$x^{p+1} = w^p,$$

that is, one neuron will produce a delay



a)

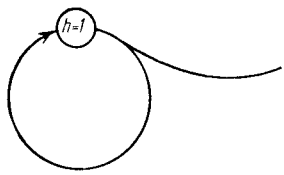
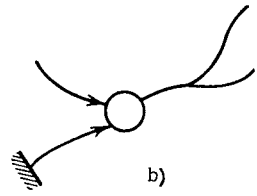


Fig. 4.20.



b)

Fig. 4.21.

of one sampling instant. By connecting q such neurons in series (Fig. 4.23), we can produce a delay of q sampling instants.

Fig. 4.22.

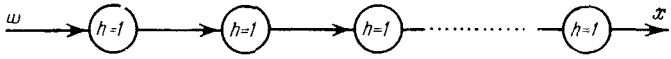


Fig. 4.23.

Consider now a neuron for which $h = 1$, $s = 1$, $q = 1$, and fix the input stimulus (Fig. 4.24). Then the neuron embodies the relationship

$$x^{p+1} = \overline{v^p},$$

that is, the single neuron performs the operation of negation with a delay of one instant.

A neuron for which $h = 1$, $q = 0$, and s is any number (Fig. 4.25) is an embodiment of a disjunction of s variables with a delay of one instant:

$$x^{p+1} = w_1^p \vee w_2^p \vee \ldots \vee w_s^p,$$

while a neuron with $h = s$ (for any s) and $q = 0$ (Fig. 4.26) is an embodiment of a conjunction of s variables with a delay of one instant.

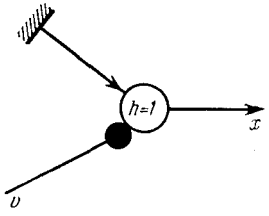


Fig. 4.24.

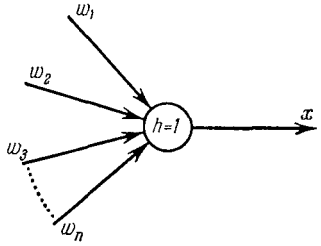


Fig. 4.25.

If we desire to perform a conjunction of s variables, some of which are negated, then the negated variables must be introduced at the inhibitory inputs while h must equal the number of negated variables. Thus, for example, the conjunction

$$x^{p+1} = w_1^p \,\&\, w_2^p \,\&\, \overline{w_3^p} \,\&\, \overline{w_4^p}$$

can be embodied by a single neuron, as shown in Fig. 4.27.

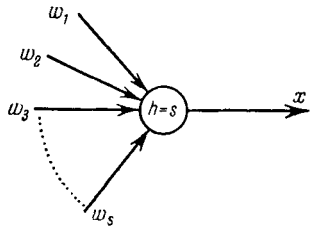


Fig. 4.26.

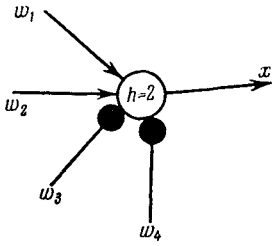


Fig. 4.27.

Consider now an arbitrary disjunction of conjunctive groups. Let such a disjunction contain m groups. Then each conjunctive group may be performed by a single neuron in accordance with Fig. 4.27. The outputs of these neurons are connected to a neuron which performs the disjunction (Fig. 4.25). Since all the neurons performing conjunctions "fire" during one instant, and since just one additional instant is required for performing a disjunction, the entire disjunctive form may be performed in two instants, that is, in time 2τ. Thus,

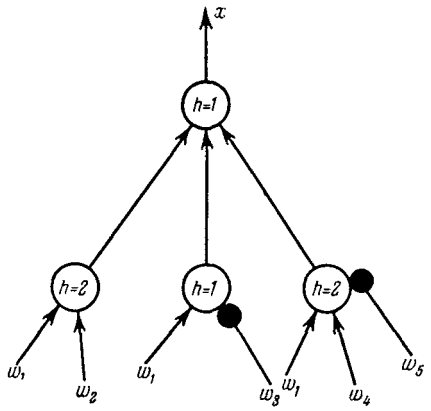


Fig. 4.28.

Fig. 4.28 shows a net consisting of McCulloch-Pitts neurons and embodying the form

$$x^{p+2} = (w_1^p \,\&\, w_2^p) \vee (w_1^p \,\&\, \overline{w_3^p}) \vee (w_4^p \,\&\, \overline{w_5^p} \,\&\, w_1^p).$$

Since any logical function may be presented as a disjunction of conjunctive groups, it can be performed over two sampling instants by an abstract net consisting of McCulloch-Pitts neurons. Thus any

logical converter L may be constructed from McCulloch-Pitts neurons, but, in contrast to our usual assumptions, such a converter will not be instantaneous, because it will require two sampling instants to finish its operation.

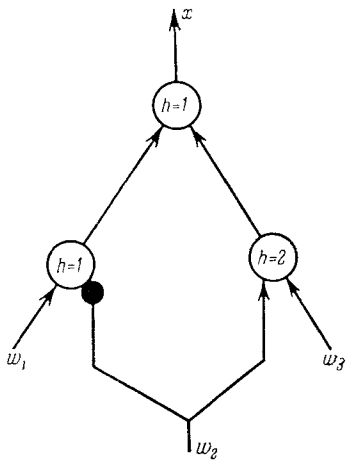


Fig. 4.29.

Assume now that the inputs vary and that the state of the net is observed at instants 0, 2τ, 4τ, ... With such a timing, one can construct a net of McCulloch-Pitts neurons that performs any logical conversion over a single sampling instant. Also, neurons may be employed to form a delay element for such a "doubled" timing; to do this (see Fig. 4.23), two delay elements are connected in series. Now, having a logical converter performing any desired conversion and having a delay element, we can construct a net embodying any desired automaton operating with such a timing.

In Chapter 10 we shall consider methods for synthesizing automata operating with any desired "slow" timing, starting from elements operating with "fast" timing, provided the synchronizing signals for the occurrence of the "slow" timing are supplied from an outside source. We shall show that to synthesize such systems we need elements capable of producing a delay for any such externally supplied timing.

To finish the discussion of neural nets, we shall show how such a delay element may be synthesized from McCulloch-Pitts neurons. Thus let us construct a net (Fig. 4.29) that performs, over time 2τ, the function

$$x^{p+2} = (w_1^p \,\&\, \overline{w_2^p}) \vee (w_3^p \,\&\, w_2^p).$$

We combine two such nets into one net with two inputs (u and u_t), as shown in Fig. 4.30. The input u is the basic input of the net, while u_t is used for introduction of a synchronizing signal (it is assumed that the next sampling instant occurs when u_t changes from 1 to 0).

Figure 4.31 shows the variation of x and x_1 for some variations of u and u_t. The value of x coincides with the value of u, but with a delay of one sampling instant. The net will function correctly provided signals u_t follow each other at intervals not shorter than 4τ. The arrows in Fig. 4.31 bracket intervals 2τ.

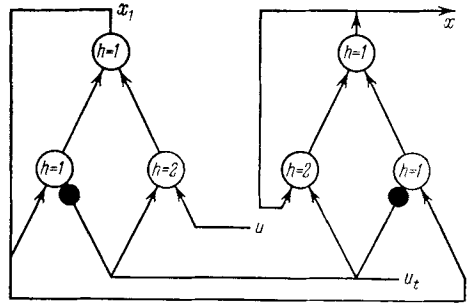


Fig. 4.30.

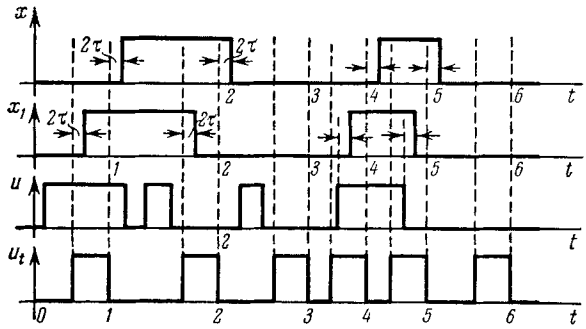


Fig. 4.31.

Thus, having a delay element for any external synchronizing source, as well as a converter "firing" in the time 2τ, one can, with the aid of the methods of Chapter 10, use McCulloch-Pitts neural nets to embody any automaton (or sequential machine) with any desired timing, provided a single condition is satisfied: the interval between sampling instants cannot be shorter than 4τ.