3

Finite Automata and Sequential Machines: Basic Concepts

3.1. DISCRETE TIME AND DISCRETE TIME MOMENTS

Let $\{x\}_i$ $(i = 1, 2, \ldots, n)$ and $\{y\}$ be alphabets containing a finite number of symbols. Then the functional relationship

$$y = f(x_1, x_2, \ldots, x_n) \tag{3.1}$$

matches any set of symbols, taken one at a time from alphabets $\{x\}_1, \{x\}_2, \ldots, \{x\}_n$, with one symbol of alphabet $\{y\}$.

Now consider an ideal device embodying relationship (3.1). This device has n inputs and a single output. Inputs $x_1, x_2, \ldots, x_n$ are fed symbols from alphabets $\{x\}_1, \{x\}_2, \ldots, \{x\}_n$, respectively, all these inputs being made in a block, that is, at exactly the same time. This instantaneously generates a symbol from alphabet $\{y\}$ at the output, as specified by (3.1). We shall call such an instantaneously operating ideal device a *function converter.* In the special case when each of the alphabets $\{x\}_1, \{x\}_2, \ldots, \{x\}_n$ and $\{y\}$ consists of two symbols only, that is, when $x_1, x_2, \ldots, x_n$ and y are logical variables and f is a logical function, such a device is a *logical converter.* Instantaneously responding combinational relay switching circuits and similar devices for performing the operations of propositional calculus would be practical embodiments of the abstract concept of a "logical converter."

So far, our functional relationships have neglected the time factor and we have also assumed that the function converter acts instantaneously. Now, however, we shall introduce the concept of time.

We usually assume that time varies only in one direction ("into the future"), that it varies continuously, and that it thus passes through all possible values on the positive real axis. In other words,

when time appears as an argument of a function, it is usually defined on a continuum, namely, the positive real axis (the time axis).

In contrast to this, it is convenient to study discrete-action devices in terms of a hypothetical discrete time. Let us imagine that the continuous time axis can be divided into an infinite number of finite intervals, not necessarily of equal length (Fig. 3.1). Moving along the axis from $t = 0$ toward $t = \infty$, we mark the points separating these intervals by characters $t_0, t_1, t_2, \ldots$ These points then constitute a countable set.

Let us further agree to represent the characters $t_0, t_1, t_2, \ldots$ by a series of positive integers 0, 1, 2, ... and call that imaginary time which consecutively assumes only these integral values the *discrete time* t.

The time instants $t_0, t_1, t_2, \ldots$, now denoted by numbers 0, 1, 2, ..., shall be called *discrete moments*, and the numbers 0, 1, 2, ... shall be treated as symbols constituting an alphabet $\{t\}$.

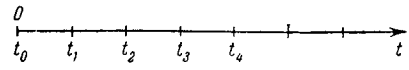

Fig. 3.1.

The current discrete moment (the one corresponding to the present instant) shall be denoted by p (present). Thus p divides all t's into those preceding p ($p-1, p-2, \ldots$) and those following ($p+1$, $p+2, \ldots$) [Fig. 3.2].

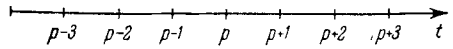

Fig. 3.2.

Thus far, we have treated the variables of Eq. (3.1) as time-variant. Assume now that $x_1, x_2, \ldots, x_n$ vary in the discrete time. That is, the variables assume definite values for each t, so that we have functions $x_i(t)$, where $t = 0, 1, 2, \ldots$, and where $\{x\}_i$ assumes values from alphabets $\{x\}_i$ ($i = 1, 2, \ldots n$). Then, by virtue of (3.1), we can set up a correspondence between these functions and the function

$$y(t) = f[x_1(t), x_2(t), \ldots, x_n(t)], \tag{3.2}$$

where y varies with the same t as x_i and assumes values from alphabet $\{y\}$. Such a system operates in discrete time but "has no memory" in the sense that the "output" value y at any instant $t = p$ depends solely on the values of the "inputs" x_i at that instant.

One can, however, imagine systems which also operate in discrete time and whose inputs and outputs are also symbols drawn from infinite alphabets, but in which the relationship between the

input and the output is not so simple. For example,

$$y(p) = f[x_1(p-1),\ x_2(p-1),\ \ldots,\ x_n(p-1);\ x_1(p),\ x_2(p),\ \ldots,\ x_n(p)]. \tag{3.3}$$

In other words, the value of y at any $t = p$ depends not only on the values of all the x_i at p but also on their values of the preceding discrete moment $p-1$. Further, the value of y at $t = p$ may even be a function of the entire history of values y. Consider the case in which y is a logical variable whose value at any p is specified as a negation of the value of y at the preceding moment $p-1$:

$$y(p) = \overline{y(p-1)}. \tag{3.4}$$

Although the relation

$$y = \bar{y}$$

is contradictory, Eq. (3.4) does not lead to contradictions; it specifies a function $y(t)$ which consecutively assumes the values 1 and 0 even though there are no input (external) signals.

These special dynamical systems differ from the common ones (such as the pendulum or the four-terminal network) in that they operate in discrete time and their coordinates (inputs and outputs) are defined on finite sets.

Henceforth we shall be dealing with dynamical systems that are distinguished by these two properties.

3.2. ON DYNAMICAL SYSTEMS

A *dynamical system* is one involving time-varying processes. The state of a natural or man-made dynamical system at any instant is given by some number (finite or infinite) of *generalized coordinates*. Dynamical systems may be divided into several classes, depending on:

a) whether they are time-continuous or time-discrete, that is, whether the time is assumed to vary in a continuum or a countable set;

b) whether the system has a finite or infinite number of generalized coordinates; and finally

c) the cardinality of the set of all possible values of each generalized coordinate, that is, whether these sets are finite, countably infinite, or continua.

The concept "dynamical system" is usually associated with systems described by ordinary or partial differential equations. In systems of this type the number of generalized coordinates may be finite (in which case they are described by ordinary differential equations) or infinite (described by partial differential equations), but in either case both the coordinates and the time vary in continua.

In those cases where the time is discrete, that is, varies in a countable set, while each of the finite or infinite number of generalized coordinates may assume values from continuum sets, the behavior of the system is described by difference equations.

In a special class of dynamical systems the time is again discrete but the generalized coordinates (whose number may be finite or infinite) assume values from finite sets.

Every dynamical system may be affected by externally generated input signals. Such input signals may also be defined on a continuum, a countable set, or a finite set. Dynamical systems described by differential or difference equations are usually capable of handling only a finite number of input signals; the latter, however, may take on any values from some continuum. Dynamical systems whose generalized coordinates are defined on finite sets are usually analyzed in terms of a finite number of input signals, and each of these signals is also defined on a finite set.

Dynamical systems in which time is defined on a countable set, the coordinates and (externally generated) input signals are defined on finite sets, and the number of input signals and coordinates is finite will be called *finite dynamical systems*. Particular cases of this class of systems are finite automata and sequential machines.

Systems that differ from the finite only in that they have an infinite number of generalized coordinates constitute a more general class of dynamical systems. These include Turing machines* and similar idealized devices.

The reader must be reminded at this point that an equation describes only an idealized model and not a real system. In this sense any dynamical system is an abstraction. But although finite dynamical systems and Turing machines are no more than abstractions, they are very important abstractions because many technical devices and important natural processes lend themselves to representation by such abstractions.

---

*See Chapter 8.

3.3. FINITE AUTOMATA

Consider a finite dynamical system whose state at any instant is characterized by a finite number of generalized coordinates $\varkappa_1, \varkappa_2, \ldots, \varkappa_n$. This system is subject to a finite number of externally generated input signals $\rho_1(t), \rho_2(t), \ldots\ldots, \rho_m(t)$. We are given either a time scale divided into discrete moments or conditions that enable us to determine the instant at which the next discrete moment will occur. In our definition, the signals and the states of the system are meaningful *only* at such discrete moments (and are neglected at all other times).

At these moments, each of the generalized coordinates $\varkappa_i$ may take on values only from a finite set, while each input ρ_j also assumes values only from its finite set (of input signals).

Let us introduce an n-dimensional vector $\varkappa$ with coordinates $\varkappa_1, \varkappa_2, \ldots, \varkappa_n$ and an m-dimensional vector ρ with coordinates $\rho_1, \rho_2, \ldots, \rho_m$. Because all the coordinates of the vector $\varkappa$, that is, $\varkappa_1, \varkappa_2, \ldots, \varkappa_n$ are defined on finite sets, the vector $\varkappa$ is also defined on a finite set. If the coordinate $\varkappa_i$ may take on ξ_i values, then the vector $\varkappa$ may assume one of $k = \prod_{i=1}^{n} \xi_i$ values. Accordingly, the set on which $\varkappa$ is defined consists of k elements.

By exact analogy, the vector ρ with coordinates $\rho_1, \rho_2, \ldots, \rho_m$ is given on a finite set containing r elements, where $r = \prod_{j=1}^{m} \eta_j$, and η_j is the number of elements in the set on which ρ_j is defined.

Let us consider an alphabet $\{\varkappa\} = \{\varkappa_1, \varkappa_2, \ldots, \varkappa_k\}$ consisting of k symbols, and let us match the various possible values of the vector $\varkappa$ to the various symbols from this alphabet. We shall call the vector $\varkappa$ the *state* of our finite dynamical system.

Similarly, let us introduce an alphabet

$$\{\rho\} = \{\rho_1, \rho_2, \ldots, \rho_r\},$$

consisting of r symbols, and match the various values of the vector ρ to the various symbols from this alphabet. We shall call the vector ρ the *input* to the system.

Now we shall define "motion" in our system, that is, we shall specify the method by which the state of the system is defined at each discrete moment of time. One very important definition leads to the concept a finite automation.

Definition. A finite dynamical system is said to be a finite automaton if its state at each discrete moment is uniquely defined (a) by

its state in the preceding moment and (b) by its input at the preceding or the current moment.

A finite automaton whose current state is defined by its state and input at the preceding moment shall be called a *finite automaton of the P - P* (past — past) *type.* An automaton whose current state is defined by its state at the preceding moment and its current input shall be called a *finite automaton of the P - Pr* (past-present) *type.*

The term "finite automaton" also includes finite systems whose states are defined by their states and inputs during any desired finite number of preceding moments. The term does not, however, pertain to finite systems whose states are defined by random factors or by their entire history (that is, systems whose states and inputs cannot be specified unless one knows their value at all the preceding discrete moments).

To put the above definition of the finite automaton in other terms, the symbol $\varkappa$ at any discrete moment is uniquely defined by the $\varkappa$ of the preceding moment and ρ at the preceding or in the current moment. That is, for a P - P automaton:

$$\varkappa^{p+1} = F(\varkappa^p, \rho^p), \quad p = 0, 1, 2, \ldots, \tag{3.5'}$$

and for a finite automation of the P - Pr type

$$\varkappa^{p+1} = F(\varkappa^p, \rho^{p+1}), \quad p = 0, 1, 2, \ldots, \tag{3.5''}$$

where F is a function in the sense of Chapter 1 (it matches a symbol from the alphabet $\{\varkappa\}$ with symbols from the alphabets $\{\varkappa\}$ and $\{\rho\}$. However, in contrast to the sense of Chapter 1, the symbol-arguments and the symbol-function may now pertain to differing time instants. For this reason formulas (3.5) do not specify a converter but a dynamical system.

The discrete moments corresponding to given $\varkappa$ and ρ are identified by superscripts, where p stands for the present, $p + 1$ the next, and $p - 1$ the immediately preceding moment.

If a new symbol μ is defined in the same alphabet $\{\varkappa\} = \{\varkappa_1, \varkappa_2, \ldots, \varkappa_k\}$ as $\varkappa$, then relations (3.5′) and (3.5″) can be treated as derived from

$$\left.\begin{aligned} \mu &= F(\varkappa, \rho), \\ \varkappa^{p+1} &= \mu^p. \end{aligned}\right\} \tag{3.6}$$

In the first of these relations, all the symbols pertain to the same time. If that time is p, that is,

$$\mu^p = F(\varkappa^p, \rho^p),$$

when we add the second relations (3.6) and eliminate μ^p we get

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

that is, relation (3.5′).

If, however, the time corresponding to the first of relations (3.6) is $p + 1$, that is

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

we can add the second of relations (3.6), eliminate $\varkappa^{p+1}$ and get

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

that is, relation (3.5″).

Let us consider (3.5′):

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

Knowing ρ^0 and $\varkappa^0$ for the moment zero, we can, by putting $p = 1$, find $\varkappa^1$. Then, knowing $\varkappa^1$ and ρ^1, we can find $\varkappa^2$, and so on. The values of $\varkappa^1, \varkappa^2, \ldots$ can be determined in a similar fashion from Eq. (3.5″), starting from a given $\varkappa^0$ and a given input sequence $\rho^1, \rho^2, \ldots$ In this respect formulas (3.5) determine recurrence relations, which permit us to find consecutively all the $\varkappa^1, \varkappa^2, \ldots$, provided the initial state $\varkappa^0$ and the input sequence $\rho^0, \rho^1, \rho^2, \ldots$ are known.

We have already stated that $\varkappa$ and ρ as well as the behavior of the system, in general, are significant only during discrete time moments. Thus, in dealing with a real dynamical system (or process) we imagine a device that records (samples) the values of $\varkappa$ and ρ at such moments*, and that the decision as to whether or not the system is a finite automaton is made only on the strength of such a sampling record. In this sense the abstract concept of a "finite automaton" may also be employed to describe continuous devices (devices exhibiting a continuum of states varying in

*Or that there exists a stroboscopic device illuminating the observed process only at these instants.

continuous time), provided only that during the discrete sampling moments, when the system is observed, the set of its possible states is finite and that one of relations (3.5) is satisfied. Thus, for example, a continuous system having several equilibrium states may be treated as a finite automaton. This is possible provided the data sampling moments are made to coincide with the instants at which complete equilibrium becomes established and provided the state of equilibrium is always uniquely determined by the system's preceding equilibrium and by the input signals to which it is subjected at the instant when that equilibrium is disturbed (or established).

Since all real systems operate in continuous time, the use of discrete time in our discussion calls for a special device, a synchronizing source, which signals the advent of the next discrete time moment (that is, data sampling moment). We shall call such a source a *discrete clock* (or simply a clock).

The clock is not an integral part of the finite automaton. The signals it generates are external to the automaton in the same sense as are the input signals ρ. But the clock signals—the time input—do differ from the externally generated input signals, since they are not coded in symbols from the alphabet $\{\rho\}$ and they do not constitute arguments of function F in (3.5). If the finite automaton is a process, the clock signals can be used only in some device that records ρ and $\varkappa$ at the various time instants. In technical embodiments of a finite automaton, the clock signals are used only to determine the advent of the next discrete time moment.

Let us now examine some examples of the division of a continuous time scale into discrete moments.

a) The continuous time is divided into equal intervals so that an ordinary clock with a suitable regulated movement may serve as a synchronizing source. This is *uniform* time division.

b) The next discrete moment occurs wherever the symbol ρ is changed, that is, whenever there is a change in the input. Here, the continuous time is divided into a sequence of intervals that are not, in general, of the same length. The clock may then be any device that responds to a change in input.

c) The next moment occurs whenever a symbol ρ_i or ρ_j appears at the input.

d) The next moment occurs whenever a symbol ρ with an odd superscript is replaced by a ρ with an even superscript; and so on.

Returning to formulas (3.5), let us now assume that the input ρ does not vary. Then we have

$$\varkappa^{p+1} = F[\varkappa^p, \rho_*], \tag{3.7}$$

where ρ_* is a constant value of ρ. We shall call such a finite automaton *self-contained* or *autonomous*. It is independent of the externally generated input signals, but it still employs clock signals to indicate the next discrete time moment.

The symbol $\rho = \rho_*$ may be regarded as a parameter because it may be assigned any symbol from the alphabet $\{\rho\}$. By so doing we obtain r autonomous automata. In this sense each finite automaton may be transformed into r autonomous automata (some of which may be identical).

There is still another, formal definition of a finite automaton. This definition is unrelated to the concepts of a finite dynamical system, a state or an input. It merely says that *given two finite alphabets of symbols* $\{\varkappa\}$ *and* $\{\rho\}$, *as well as the variables* $\varkappa$ *and* ρ *which assume values from these alphabets, a finite automaton consists of the recurrence relations (3.5) coupling these variables.*

This is a very broad and a very abstract definition, but its value lies precisely in its generality. It applies to a great variety of seemingly unrelated devices, processes and phenomena. By using it, one can introduce order where there seems to be none, and discover general laws governing all these systems, starting from the most general assumptions. This is the object of the theory of finite automata.

3.4. SEQUENTIAL MACHINES

Consider a system (Fig. 3.3) consisting of (a) a finite automaton A, which converts symbols ρ of the alphabet $\{\rho\}$ into symbols $\varkappa$ of the alphabet $\{\varkappa\}$ as per Eq. (3.5′) or (3.5″), where the function F is given, and (b) a converter Φ which instantaneously and uniquely matches each symbol $\varkappa$ with a symbol λ from an alphabet $\{\lambda\}$:

$$\lambda^p = \Phi(\varkappa^p). \tag{3.8}$$

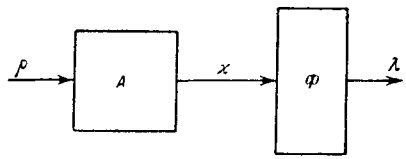

Fig. 3.3.

The instants at which the symbols ρ and λ appear coincide with the discrete time scale specific to the given automaton A. If one can select a single-valued function F^* (which may differ from F) in such a way as to make symbols λ satisfy

a relation of the form (3.5′) or (3.5″), that is, if

$$\lambda^{p+1} = F^*[\lambda^p, \rho^p] \quad \text{or} \quad \lambda^{p+1} = F^*[\lambda^p, \rho^{p+1}], \tag{3.9}$$

then the system consisting of automaton A and converter Φ will also be a finite automaton. Naturally, such a function F^* is not always available if for no other reason than that the alphabet $\{\lambda\}$ may differ from the alphabet $\{\varkappa\}$ in the number of symbols it contains; that is, the same symbol λ may be used to code several symbols $\varkappa$.

For example, let the alphabets $\{\varkappa\}$ and $\{\lambda\}$ consist of eight and two symbols, respectively, and let the converter Φ generate the symbol λ_1 in response to an input of symbols $\varkappa_1$ to $\varkappa_4$ or the symbol λ_2 when the input consists of any of the symbols $\varkappa_5$ to $\varkappa_8$. We shall assume that Eq. (3.5) holds and that F is such that after $\rho^p = \rho_1$, $\varkappa^p = \varkappa_1$ there follows the symbol $\varkappa^{p+1} = \varkappa_3$, and after $\rho^p = \rho_1$, $\varkappa^p = \varkappa_4$ there follows the symbol $\varkappa^{p+1} = \varkappa_6$. In the first instance the counting device will register

$$\rho^p = \rho_1, \quad \lambda^p = \lambda_1, \quad \lambda^{p+1} = \lambda_1,$$

and in the second case

$$\rho^p = \rho_1, \quad \lambda^p = \lambda_1, \quad \lambda^{p+1} = \lambda_2.$$

Thus, identical λ^p and ρ^p may be followed by different λ^{p+1}. This means that our automaton-converter system is not in itself an automaton, since it does not preserve a relation of the form (3.5′) between symbols λ and ρ.

The system shown in Fig. 3.3 is, however, a finite dynamical system. We shall call it a *finite automaton with an output converter*, or simply a *finite automaton with output*. In this case, the symbols λ are called the *output symbols* (as distinct from $\varkappa$, the *state symbols)*, the alphabet $\{\lambda\}$ is called the *output alphabet*, and the converter Φ is called the *output converter*.

In a more general case, the converter may have two inputs. The symbols fed to one of them are again from the alphabet $\{\varkappa\}$, while the signals to the other are symbols ρ. The converter then instantaneously matches a symbol λ with each $(\varkappa, \rho)$ pair (Fig. 3.4). A finite dynamical system obtained by coupling a finite automaton to an output converter which admits the symbol ρ (Fig. 3.4) is called a *sequential machine* (or, briefly, an *s-machine*). Of course, an s-machine may also be a finite automaton. Whether it is or not depends on the form of the function F in Eqs. (3.5) for the automaton

A, as well as the type of converter Φ used. In all cases, however, the system of Fig. 3.4 is a finite dynamical system.

A sequential machine becomes a finite automaton (that is, operates as a finite automaton) if the values of λ (the output) are uniquely defined by the value λ at the preceding discrete moment and the value of ρ at the current moment, that is, if the relationship

$$\lambda^{p+1} = F^*[\lambda^p, \rho^{p+1}]$$

holds. This, for example, will be the case when one uses an identity converter whose alphabet $\{\lambda\}$ coincides with the alphabet $\{\varkappa\}$, that is, a converter that generates a symbol λ^p coinciding with the input symbol $\varkappa^p$ regardless of ρ^p. In this sense, the concept of a "finite automaton" is a special case of the abstraction "sequential machine."

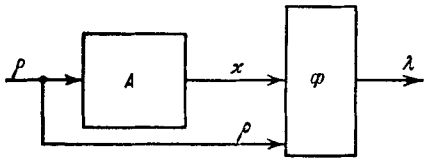

Fig. 3.4.

A finite automaton with an output converter may be treated as a special case of an s-machine in which the function Φ is independent of ρ.

At a first glance, the concept of a sequential machine appears broader than that of a finite automaton with output. However, this is not the case. This will be proven in Section 4.3, after we have formulated the concept of a "net."

A sequential machine is of the P - P or P - Pr type, depending on the automaton it contains. We shall now consider an arbitrary s-machine of the P - Pr type:

$$\varkappa^{p+1} = F[\varkappa^p, \rho^{p+1}], \quad \lambda^p = \Phi[\varkappa^p, \rho^p]. \tag{3.10}$$

Eliminating the symbol $\varkappa^p$ from the converter equation, we get

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

Let us now introduce the symbol $\tilde{\varkappa}$, which is defined in the alphabet

$\{\varkappa\}$ by the relation

$$\tilde{\varkappa}^{p+1} = \varkappa^p.$$

After substitution, we obtain

$$\lambda^p = \Phi[F(\tilde{\varkappa}^p, \rho^p), \rho^p] = \Phi^*(\tilde{\varkappa}^p, \rho^p).$$

If we now employ $\tilde{\varkappa}$ in Eq. (3.10) for the automaton, we obtain

$$\tilde{\varkappa}^{p+1} = F[\tilde{\varkappa}^p, \rho^p];$$

that is, all these transformations give an s-machine of the P - P type:*

$$\tilde{\varkappa}^{p+1} = F(\tilde{\varkappa}^p, \rho^p), \quad \lambda^p = \Phi^*(\tilde{\varkappa}^p, \rho^p). \tag{3.11}$$

Thus, any P- Pr type s-machine may be transformed into P - P type s-machine merely by replacing its output converter Φ by a Φ^* converter. However, the reverse is not generally true. We shall return to this problem in Section 5.4.

3.5. TECHNIQUES FOR DEFINING FINITE AUTOMATA AND SEQUENTIAL MACHINES

Any function

$$z = F(x, y),$$

where x and y assume values from finite sets, may be given by a table such as 3.1 showing the corresponding values of z.

The equation of finite automaton of the P - P type

$$\varkappa^{p+1} = F[\varkappa^p, \rho^p]$$

or the P - Pr type

$$\varkappa^{p+1} = F[\varkappa^p, \rho^{p+1}]$$

may be represented by an analogous Table 3.2, in which the $\varkappa^{p+1}$ symbol for a P - P automaton is represented by the intersection of row $\varkappa^p$ and column ρ^p, while the intersection of row $\varkappa^p$ and column ρ^{p+1} defines the symbol $\varkappa^{p+1}$ for a P - Pr automaton. We shall call this the ***basic table*** of the finite automaton.

*A finite automaton with an output converter described by Eqs. (3.5') and (3.8) is frequently referred to as a Moore machine (see [73]), and an s-machine given by Eq. (3.11) is called a Mealy machine (see [190]).

Table 3.1

x \ y	y_1	y_2	...	y_m
x_1	z_2	z_1	...	z_6
x_2	z_5	z_1	...	z_2
...	...	...	...	...
x_n	z_4	z_5	...	z_3

Table 3.2

$\varkappa^p$ \ ρ	ρ_1	ρ_2	...	ρ_r
$\varkappa_1$	$\varkappa_3$	$\varkappa_2$	...	$\varkappa_1$
$\varkappa_2$	$\varkappa_1$	$\varkappa_6$	...	$\varkappa_k$
...	...	...	...	...
$\varkappa_k$	$\varkappa_5$	$\varkappa_4$	...	$\varkappa_5$

Each column of such a basic table is, in turn, the basic table for an autonomous automaton, which is obtained from the finite one by making the ρ value in the heading of that column a constant.

Consider one such autonomous automaton (for example, Table 3.3). If we draw k circles on a plane, assign to each circle a symbol $\varkappa$, and draw arrows which show the transitions occurring in the autonomous automaton in accordance with its basic table, we obtain its *graph*. Only one arrow can start at each circle, but any number of arrows (not exceeding k, however) may terminate at it. The graph of Fig. 3.5 corresponds to Table 3.3. Since each finite automaton yields r autonomous ones, the basic table of one finite automaton yields r graphs of autonomous automata. Figure 3.6 shows the graphs derived from Table 3.4.

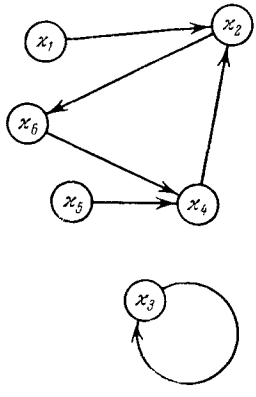

Fig. 3.5.

Since all these r graphs consist of the same k circles, they can be combined into one graph, in which each arrow is labeled with the ρ value at which that arrow can be performed. These labels are placed at the origin of the arrow in the case of the P - P automaton, and at its tip in the case of the P - Pr machine. Figure 3.7 shows the combined graph for Table 3.4 and Fig. 3.6, that is, for a P - Pr automaton.

Such a graph is called the *state diagram* of the automaton. In this case, each circle is the origin of r arrows. If several of these terminate at the same circle, they may be combined into one, the labels indicating their ρ values being joined by means of disjunction signs. The *state diagram* is fully equivalent and interconvertible with the basic table.

Table 3.3

χ \ ρ	$\rho_j = \rho^*$
χ_1	χ_2
χ_2	χ_6
χ_3	χ_3
χ_4	χ_2
χ_5	χ_4
χ_6	χ_4

Table 3.4

χ \ ρ	ρ_1	ρ_2	ρ_3
χ_1	χ_5	χ_6	χ_6
χ_2	χ_4	χ_4	χ_4
χ_3	χ_3	χ_3	χ_2
χ_4	χ_3	χ_2	χ_5
χ_5	χ_6	χ_5	χ_1
χ_6	χ_2	χ_5	χ_3

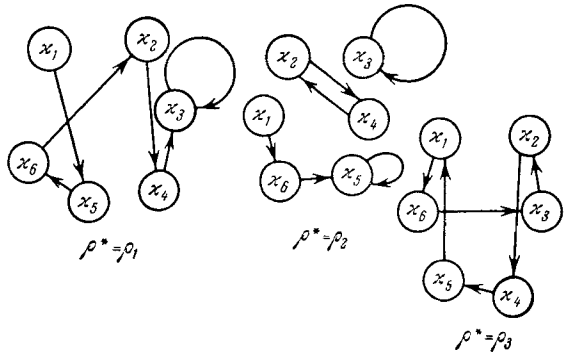

Fig. 3.6.

Let us construct a square $k \times k$ matrix C whose rows (from top to bottom) and columns (from left to right) are headed by symbols $\chi_1, \chi_2, \ldots, \chi_k$. The matrix element at the intersection of the χ_q-th row and the χ_s-th column is the label of the state diagram arrow that connects the circle χ_q with the circle χ_s. This element may consist of one symbol ρ or a disjunction of several ρ's. If there is no arrow between a circle χ_l and a circle χ_m, then the corresponding square of matrix C contains a 0. Thus we obtain the matrix of Table 3.5 for the state diagram of Fig. 3.7.

This matrix is called an *interconnection* (or transition) *matrix* and is still another way of defining a finite automaton. Here, just as in the case of a basic table, one must specify beforehand whether the

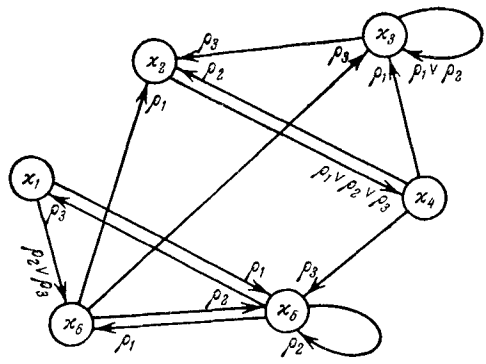

Fig. 3.7.

Table 3.5

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0	0	0	ρ_1	$\rho_2 \vee \rho_3$
x_2	0	0	0	$\rho_1 \vee \rho_2 \vee \rho_3$	0	0
x_3	0	ρ_3	$\rho_1 \vee \rho_2$	0	0	0
x_4	0	ρ_2	ρ_1	0	ρ_3	0
x_5	ρ_3	0	0	0	ρ_2	ρ_1
x_6	0	ρ_1	ρ_3	0	ρ_2	0

automaton is of the P - P or the P - Pr type. Each row of the matrix must contain every ρ_i once, and only once. The matrix may be derived directly from the basic table, dispensing with the intermediate state diagram.

Assume we have a basic table for a P - P automaton, that is, for the relation

$$x^{p+1} = F[x^p, \rho^p].$$

Then each cell of the basic table defines three symbols:

$$x^p, \rho^p, x^{p+1},$$

that is, the row heading, the column heading, and the character contained in the cell. Let us call such a symbol triplet a triad. Since the whole table has rk squares, and each square defines a triad, the table defines a set of rk triads. We shall say that a set of triads is ordered if the first two symbols (x^p and ρ^p) of any two triads of that

set do not coincide. Any basic table of a finite automaton defines an ordered finite set of triads. Conversely, any ordered set of rk triads defines a basic table, that is, a finite automaton.

The above also applies to P - Pr automata, that is, those defined by

$$\varkappa^{p+1} = F[\varkappa^p, \rho^{p+1}].$$

However, in this case each triad consists of the symbol triplet

$$\varkappa^p, \rho^{p+1}, \varkappa^{p+1}$$

so that an ordered set is said to be one in which no two triads have identical first two symbols $\varkappa^p$ and ρ^{p+1}.

Now, if we wish to define a sequential machine in a table form, we must consider simultaneously the equations of its constituent automaton [(3.5′) or (3.5″)] and converter (3.8). To do this, we draw up the basic table for the finite automaton and add to each square the symbol λ^{p+1} resulting from the converter equation. This combined table is the *basic table of the s-machine*.

If the automaton is of the P - P type [Eq. (3.5′)], then we add to its table symbol λ^p. For example, the automaton of Table 3.4 plus the converter of Table 3.6 define the s-machine basic table 3.7. If, however, the automaton is of the P - Pr type [Eq. (3.5″)], then we add to each square the symbol λ^{p+1}, obtained from the converter table at the intersection of $\varkappa^{p+1}$ (the symbol already present in the square of the basic table) and ρ^{p+1} (the heading of the basic table column in which the square is situated). Thus, if the P - Pr automaton is given by Table 3.4 and the associated converter by Table 3.6, then the basic table of the s-machine incorporating this P - Pr automaton is Table 3.8.

Let us note in passing that the converter table for the P - Pr case may contain blank spaces, because some $\varkappa$ values may be missing from the corresponding column of the automaton table. Thus we could leave the square $(\rho_1, \varkappa_1)$ of Table 3.6 blank, because column ρ_1 of Table 3.4. contains no $\varkappa_1$ entries.

To obtain a state diagram of an s-machine, we modify the diagram of the corresponding automaton by including the appropriate λ symbol at each circle. However, in an s-machine λ is defined not only by $\varkappa$, but also by ρ. For this reason the state diagram for an s-machine differs from that of a finite automaton with an associated output converter in that the symbol λ is not written inside a circle, but *side by side with the symbol* ρ *(above the arrow)*. The arrow connects state $\varkappa^p$ with state $\varkappa^{p+1}$. For a P - P automaton [Eq. (3.5′)],

Table 3.6

x \ ρ	ρ_1	ρ_2	ρ_3
x_1	λ_1	λ_2	λ_3
x_2	λ_2	λ_2	λ_3
x_3	λ_1	λ_1	λ_3
x_4	λ_3	λ_1	λ_2
x_5	λ_1	λ_1	λ_2
x_6	λ_3	λ_3	λ_1

Table 3.7

x^p \ ρ^p	ρ_1	ρ_2	ρ_3
x_1	x_5, λ_1	x_6, λ_2	x_6, λ_3
x_2	x_4, λ_2	x_4, λ_2	x_4, λ_3
x_3	x_3, λ_1	x_3, λ_1	x_2, λ_3
x_4	x_3, λ_3	x_2, λ_1	x_5, λ_2
x_5	x_6, λ_1	x_5, λ_1	x_1, λ_2
x_6	x_2, λ_3	x_5, λ_3	x_3, λ_1

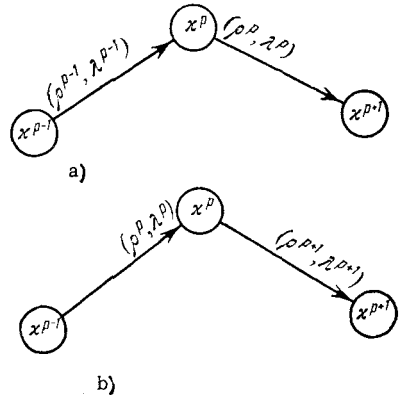

Fig. 3.8.

the symbol pair (ρ, λ) is written at the origin of the arrow (Fig. 3.8,a), whereas in the case of P - Pr automaton [Eq. (3.5'')], the label is at its tip (Fig. 3.8,b). In the first case, the output λ is defined in the converter table at the intersection of the symbol ρ, written above the arrow, and the symbol x, written inside the circle from which the arrow emerges; in the second case, the coordinates are the ρ above the arrow and the x in the circle at the tip of the arrow. Thus, Fig. 3.9 shows a state diagram based on Table 3.8.

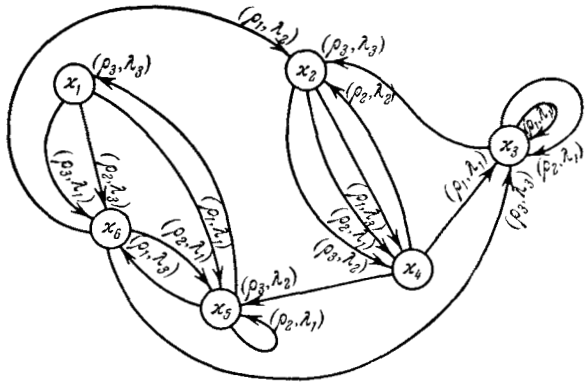

Fig. 3.9.

The state diagram of a sequential machine again gives an interconnection matrix. This matrix differs from that describing a finite automaton in that its elements consist of labels at the tips of the arrows of the state diagram, that is, the symbol pairs (ρ, λ). If the basic table is 3.8, then the interconnection matrix is that of Table 3.9.

Table 3.8

x^p \ ρ^{p+1}	ρ_1	ρ_2	ρ_3
x_1	x_5, λ_1	x_6, λ_3	x_6, λ_1
x_2	x_4, λ_3	x_4, λ_1	x_4, λ_2
x_3	x_3, λ_1	x_3, λ_1	x_2, λ_3
x_4	x_3, λ_1	x_2, λ_2	x_5, λ_2
x_5	x_6, λ_3	x_5, λ_1	x_1, λ_3
x_6	x_2, λ_2	x_6, λ_1	x_3, λ_3

3.6. RECORDING THE OPERATION OF AN AUTOMATON

So far, we have established that symbols ρ are sequentially "communicated" to the automaton (or s-machine) from the outside, and that they are independent of its operation. The machine then processes the input ρ into symbols $\varkappa$ (or λ).

Now we shall call an *input sequence* any finite (but as large as desired) series of ρ symbols, and we shall call the analogous set of $\varkappa$ or λ symbols a *state sequence* (or an *output sequence*). The number of symbols contained in such a set will be called the *length of the sequence*.

Table 3.9

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0	0	0	(ρ_1, λ_1)	$(\rho_2, \lambda_3) \vee (\rho_3, \lambda_1)$
x_2	0	0	0	$(\rho_1, \lambda_3) \vee (\rho_2, \lambda_1) \vee (\rho_3, \lambda_2)$	0	0
x_3	0	(ρ_3, λ_3)	$(\rho_2, \lambda_1) \vee (\rho_1, \lambda_1)$	0	0	0
x_4	0	(ρ_2, λ_2)	(ρ_1, λ_1)	0	(ρ_3, λ_2)	0
x_5	(ρ_3, λ_3)	0	0	0	(ρ_2, λ_1)	(ρ_1, λ_3)
x_6	0	(ρ_1, λ_2)	(ρ_3, λ_3)	0	(ρ_2, λ_1)	0

Both the automaton and the sequential machine are operators that process sequences of input symbols of one alphabet into sequences of output symbols of another alphabet. The basic table, the set of triads, the graphs, the state diagram, the interconnection matrix, and the transition table are various methods of defining such an operator, since any one of these is sufficient to recreate the corresponding sequence $\varkappa$ (or λ) if the sequence ρ and the initial state $\varkappa^0$ are known.

Table 3.10

Discrete moment	0	1	2	3	4	5	6	7	8	...
ρ	ρ_3	ρ_8	ρ_1	ρ_3	ρ_4	ρ_6	ρ_2	ρ_8	ρ_{12}	...
$\varkappa$	$\varkappa_1$	$\varkappa_2$	$\varkappa_7$	$\varkappa_1$	$\varkappa_2$	$\varkappa_{12}$	$\varkappa_6$	$\varkappa_3$	$\varkappa_8$	...

Consider now a three-row table 3.10, whose row 1 contains the ordinal number of the discrete time moment, and rows 2 and 3 contain the corresponding ρ and $\varkappa$. The table may have as many columns as desired. This is the *tape* of the finite automaton. Since a tape may be compiled for each input sequence $\rho(t)$, each automaton may have an infinite number of tapes.

Any initial piece of tape (from zero to any kth moment) contains an input sequence and the corresponding state sequence of length $k + 1$. Any three tape symbols, such as those delineated by the heavy line in Table 3.10, constitute a triad defining one cell of the basic table of the automaton (we shall use a heavy line to isolate a P - P triad, and a dotted line to delineate a P - Pr triad). Now, if we had a scanner with a cutout matching either the heavy or the dotted outline of the Table 3.10, then, sliding this scanner along the tape, we would consecutively see all the various triads contained in it (obviously, each tape has a finite number of such triads). If we could scan all the tapes of a given automaton (an infinite number), we could then read the set of all the triads contained in all the tapes. But since all these tapes are generated by a single automaton, the set of triads must be finite. In fact, it is the finite ordered set of triads containing rk elements.

In the preceding section we showed that since an automaton is an operator, it can be defined by a finite, ordered set of triads. We see now that all the tapes of the automaton consist of triads of this set.

Let us now select an alphabet of rk symbols, for instance, $\{\tau\}$, and assign symbols τ to all the triads of our automaton. Then the tape

shall be reduced to only two rows: the ordinal number of the discrete moment and, the symbol τ (see Table 3.11). However, this reduces the available degrees of freedom, for the sequence of triads in such a tape cannot be arbitrary. Indeed, let the first triad τ_5 consist of

$$\varkappa_7,\ \rho_3,\ \varkappa_{12}.$$

This immediately fixes the first symbol in the following triad, so that only the two remaining symbols can vary; that is, the next triad can only be some triad of the same set which starts with $\varkappa_{12}$, for instance, $\varkappa_{12}, \rho_8, \varkappa_6$ or $\varkappa_{12}, \rho_2, \varkappa_{12}$, and so on. The triads corresponding to a given triad τ_j are those triads from our ordered set whose first symbol coincides with the last symbol of the given triad τ_j. All the tapes of a finite automaton consist of triads arranged in such a way that each triad is followed by any one of its corresponding triads.

Table 3.11

Discrete moment	0	1	2	3	4	5	6	...
τ	τ_5	τ_1	τ_3	τ_2	τ_3	τ_1	τ_8	...

The concept of a tape may be extended to the sequential machine by supplementing the finite automaton tape with a row of output symbols λ, after which the $\varkappa$ row is deleted (Table 3.12). This tape may also be split into triads such as

$$\lambda^p,\ \rho^p,\ \lambda^{p+1} \text{ or } \lambda^p,\ \rho^{p+1},\ \lambda^{p+1}.$$

This set of triads may contain, however, some ''contradictory'' elements, in which the first two symbols λ^p, ρ^p (or λ^p, ρ^{p+1}) are identical but are followed by differing third symbols λ^{p+1}, that is, the set is not an ordered one. It becomes ordered if, and only if, the sequential machine as a whole is a finite automaton.

Table 3.12

Discrete moment	0	1	2	3	4	5	6	7	8	...
ρ	ρ_3	ρ_8	ρ_1	ρ_3	ρ_9	ρ_6	ρ_7	ρ_8	ρ_{12}	...
λ	λ_2	λ_3	λ_3	λ_2	λ_1	λ_1	λ_1	λ_2	λ_3	...

We shall now discuss still another way of describing the operation of a finite automaton. Let us draw its state diagram (Fig. 3.10)

and consider the two circles incorporating symbols $\varkappa_i$ and $\varkappa_j$, respectively. The transition from state $\varkappa_i$ to state $\varkappa_j$ may be accomplished over one discrete moment, provided the input is ρ_1. The transition over two moments may be accomplished via the following alternative routes: in the first moment, the input is ρ_2 or ρ_3, and in the second moment it is ρ_2; or it is ρ_1 in both moments. If three discrete moments are available, then one can accomplish the transition via nine different alternative routes (Table 3.13). Similarly, we can derive all possible sequences of ρ that would transform the state $\varkappa_i$ into the state $\varkappa_j$ over q discrete moments. Each such sequence is a *path of length* q leading from $\varkappa_i$ to $\varkappa_j$, and we shall represent it as a sequence of q symbols ρ; the aggregate of all the possible paths of length q shall be represented as a disjunction of such sequences. Thus, for example, Table 3.13 may be written in the form

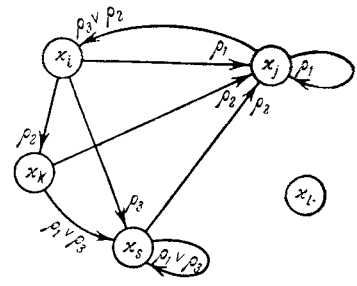

Fig. 3.10.

$$\rho_1\rho_1\rho_1 \vee \rho_1\rho_3\rho_1 \vee \rho_2\rho_1\rho_2 \vee \rho_2\rho_3\rho_2 \vee \rho_3\rho_1\rho_2 \vee \rho_3\rho_3\rho_2 \vee \rho_1\rho_2\rho_1 \vee \rho_2\rho_2\rho_1 \vee \rho_3\rho_2\rho_1 .$$

Let us denote by D^q_{ij} a disjunction describing all the possible $\varkappa_i$ to $\varkappa_j$ paths of length q. Then each disjunction $\{\rho\}$, shall comprise one or more sequences consisting of exactly q elements ρ_i, some of which may coincide.

Table 3.13

Path	Discrete moment		
	0	1	2
First	ρ_1	ρ_1	ρ_1
Second	ρ_1	ρ_3	ρ_1
Third	ρ_2	ρ_1	ρ_2
Fourth . . .	ρ_2	ρ_3	ρ_2
Fifth	ρ_3	ρ_1	ρ_2
Sixth	ρ_3	ρ_3	ρ_2
Seventh . . .	ρ_1	ρ_2	ρ_1
Eighth	ρ_2	ρ_2	ρ_1
Ninth	ρ_3	ρ_2	ρ_1

Let us now construct a matrix $C^{(q)}$. This matrix will contain D^q_{ij} (at the intersection of the ith row and the jth column) if there is at least one $\varkappa_i$ to $\varkappa_j$ path of length q, and 0 if there is no such path. For example,

$$C^{(q)} = \begin{array}{c} \\ \varkappa_1 \\ \varkappa_2 \\ \varkappa_3 \\ \varkappa_4 \end{array} \begin{array}{c} \begin{array}{cccc} \varkappa_1 & \varkappa_2 & \varkappa_3 & \varkappa_4 \end{array} \\ \begin{bmatrix} D^{(q)}_{11} & 0 & 0 & D^{(q)}_{14} \\ 0 & 0 & D^{(q)}_{23} & 0 \\ 0 & D^{(q)}_{32} & D^{(q)}_{33} & D^{(q)}_{34} \\ 0 & D^{(q)}_{42} & 0 & 0 \end{bmatrix} \end{array}.$$

This is the *matrix of the path of length* q. Just as in the interconnection matrix, the rows (top to bottom) and the columns (left to right) correspond to symbols $\varkappa_1, \varkappa_2, \ldots, \varkappa_k$ and are so denoted.

The matrix $C^{(q)}$ contains all the paths leading from any initial state to any final state (which may coincide with the initial state) over q discrete moments. Because any sequence consisting of q symbols ρ will transform the automaton from any state into the same or another state, each row of the matrix $C^{(q)}$ contains, but only once, all the possible sequences that may be formed from the alphabet $\{\rho\}$ by selecting q symbols at a time. Thus, for example, each row of the matrix $C^{(2)}$ must contain the sequences

$$\rho_1\rho_1;\ \rho_2\rho_2;\ \rho_3\rho_3;\ \rho_1\rho_2;\ \rho_2\rho_1;\ \rho_1\rho_3;\ \rho_3\rho_1;\ \rho_3\rho_2;\ \rho_2\rho_3.$$

Each row of the $C^{(q)}$ matrix may contain these groups of q symbols in different disjunctive arrangements and they may be distributed over different columns, in accordance with the basic table for a given automaton.

A complete set of matrices $C^{(1)}, C^{(2)}, C^{(3)}, \ldots$ completely specifies the operation of the automaton over any desired time period. However, such a set is not very useful and not really needed because we can always rederive the entire set of matrices from the starting interconnection matrix. We do this as follows.

To begin with, $C^{(1)} = C$, that is, the matrix of path length 1 coincides with the interconnection matrix since, by definition, its elements are those values ρ which transform $\varkappa_i$ to $\varkappa_j$ over one discrete moment.

Let us now square the interconnection matrix, in accordance with the following rules.

1. An element $C^2{}_{ij}$ of the product matrix (located at the intersection of the ith row and jth column) is specified, in accordance with general rules of matrix multiplication, as the sum of the

products of the elements of the ith row of the first factor by the elements of the jth column of the second factor. These products are not commutative; that is, in multiplying the elements, the positions of the factors cannot be interchanged.

2. Addition signs are replaced by disjunction signs throughout.
3. Multiplication signs define the operation of assigning symbols ρ.

For example, consider the matrix C for the state diagram of Fig. 3.7:

$$C = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & \rho_1 & \boxed{\rho_2 \vee \rho_3} \\ 0 & 0 & 0 & \rho_1 \vee \rho_2 \vee \rho_3 & 0 & 0 \\ 0 & \rho_3 & \rho_1 \vee \rho_2 & 0 & 0 & 0 \\ 0 & \rho_2 & \rho_1 & 0 & \rho_3 & 0 \\ \rho_3 & 0 & 0 & 0 & \rho_2 & \rho_1 \\ 0 & \boxed{\rho_1} & \rho_3 & 0 & \rho_2 & 0 \end{bmatrix} \end{array}$$

and square it in accordance with the above rules:

$$C^2 = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{bmatrix} \rho_1\rho_3 & \boxed{\rho_2\rho_1 \vee \rho_3\rho_1} & \rho_2\rho_3 \vee \rho_3\rho_3 & 0 & \rho_1\rho_2 \vee \rho_2\rho_2 \vee \vee \rho_3\rho_2 & \rho_1\rho_1 \\ 0 & \rho_1\rho_2 \vee \rho_2\rho_2 \vee \vee \rho_3\rho_2 & \rho_1\rho_1 \vee \rho_2\rho_1 \vee \vee \rho_3\rho_1 & 0 & \rho_1\rho_3 \vee \rho_2\rho_3 \vee \vee \rho_3\rho_3 & 0 \\ 0 & \rho_1\rho_3 \vee \rho_2\rho_3 & \rho_1\rho_1 \vee \rho_2\rho_1 \vee \vee \rho_1\rho_2 \vee \rho_2\rho_2 & \rho_3\rho_1 \vee \rho_3\rho_2 \vee \vee \rho_3\rho_3 & 0 & 0 \\ \rho_3\rho_3 & \rho_1\rho_3 & \rho_1\rho_1 \vee \rho_1\rho_2 & \rho_2\rho_1 \vee \rho_2\rho_2 \vee \vee \rho_2\rho_3 & \rho_3\rho_2 & \rho_3\rho_1 \\ \rho_2\rho_3 & \rho_1\rho_1 & \rho_1\rho_3 & 0 & \rho_3\rho_1 \vee \rho_2\rho_2 \vee \vee \rho_1\rho_2 & \rho_3\rho_2 \vee \rho_3\rho_3 \vee \vee \rho_2\rho_1 \\ \rho_2\rho_3 & \rho_3\rho_3 & \rho_3\rho_1 \vee \rho_3\rho_2 & \rho_1\rho_1 \vee \rho_1\rho_2 \vee \vee \rho_1\rho_3 & \rho_2\rho_2 & \rho_2\rho_1 \end{bmatrix} \end{array}.$$

We see that $C^2 = C^{(2)}$, that is, matrix C^2 is composed of all the possible paths of Fig. 3.7 which lead from one circle to another over two discrete moments. For example, element $C_{12}{}^2$ consists of $\rho_2\rho_1 \vee \rho_3\rho_1$; thus, paths $\rho_2\rho_1$ and $\rho_3\rho_1$ are the only paths over which x_1 can be transformed into x_2 in two moments.

The coincidence of C^2 and $C^{(2)}$ in this case is no accident. We see that $C_{12}{}^2$ is the product of nonzero elements C_{16} and C_{62}. However, such nonzero elements in C indicate that each of the transformations x_1 to x_6 and x_6 to x_2 requires one discrete moment, that is, that there

exist x_1 to x_2 paths which can be traversed in two moments. Therefore the coincidence found in this example is actually a general rule which states that *the square of the interconnection matrix is the matrix of all the two-moment paths*: $C^2 = C^{(2)}$. By analogy $C^3 = C^{(3)}$, that is, the cube of the interconnection matrix is the matrix of all the three-moment paths and, generally, *a matrix of all the q-moment paths is obtained by raising the interconnection matrix to the power of q*:

$$C^{(q)} = C^q.$$

We now see why we do not need the set $C^{(1)}, C^{(2)}, C^{(3)}\ldots$: all the possible paths over any number of moments q can be derived via multiplication of the interconnection matrix by itself.

All of the above also pertains to sequential machines. One must only remember that each arrow of the state diagram for the s-machine carries the two symbols ρ and λ, so that a transition in the s-machine is characterized by a pair (ρ_k, λ_s).

The "operation of symbol assignment" is then performed in accordance with the following rules:

$$1.\ (\rho_i, \lambda_j)(\rho_k, \lambda_s) = (\rho_i\rho_k, \lambda_j\lambda_s),$$
$$2.\ (\rho_k, \lambda_s)\,0 = 0\,(\rho_k, \lambda_s) = 0.$$

If one wants to multiply disjunctions of pairs, one utilizes the distributive property of the operation of multiplication of disjunctions. For example,

$$[(\rho_1, \lambda_3) \vee (\rho_2, \lambda_5)]\,[(\rho_8, \lambda_1) \vee (\rho_5, \lambda_3)] =$$
$$= (\rho_1\rho_8, \lambda_3\lambda_1) \vee (\rho_2\rho_8, \lambda_5\lambda_1) \vee (\rho_1\rho_5, \lambda_3\lambda_3) \vee (\rho_2\rho_5, \lambda_5\lambda_3).$$

As an example, consider the interconnection matrix of the tri-state s-machine whose state diagram is shown in Fig. 3.11.

$$C = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{ccc} x_1 & x_2 & x_3 \end{array} \\ \begin{bmatrix} 0 & (\rho_1, \lambda_3) \vee (\rho_2, \lambda_1) & 0 \\ 0 & (\rho_2, \lambda_2) & (\rho_1, \lambda_1) \\ (\rho_2, \lambda_3) & (\rho_1, \lambda_2) & 0 \end{bmatrix} \end{array}.$$

On squaring this matrix, we obtain

$$C^2 = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{ccc} x_1 & x_2 & x_3 \end{array} \\ \begin{bmatrix} 0 & (\rho_1\rho_2, \lambda_3\lambda_2) \vee (\rho_2\rho_2, \lambda_1\lambda_2) & (\rho_1\rho_1, \lambda_3\lambda_1) \vee \\ & & \vee (\rho_2\rho_1, \lambda_1\lambda_1) \\ (\rho_1\rho_2, \lambda_1\lambda_3) & (\rho_2\rho_2, \lambda_2\lambda_2) \vee (\rho_1\rho_1, \lambda_1\lambda_2) & (\rho_2\rho_1, \lambda_1\lambda_1) \\ 0 & (\rho_2\rho_1, \lambda_3\lambda_3) \vee (\rho_2\rho_2, \lambda_3\lambda_1) \vee & (\rho_1\rho_1, \lambda_2\lambda_1) \\ & \vee (\rho_1\rho_2, \lambda_2\lambda_2) & \end{bmatrix} \end{array}.$$

The very construction of matrix C^2 shows that the element C^2_{ij} represents the list of all the input (and the corresponding output) sequences of length 2 that will transform the ith state of the s-machine into its jth state over two discrete moments. Thus, in the case of the s-machine shown in Fig. 3.11, sequences $\rho_2\rho_1$, $\rho_2\rho_2$ and $\rho_1\rho_2$ transform the third state into the second, a transformation accompanied by the appearance of sequences $\lambda_3\lambda_3$, $\lambda_3\lambda_1$ or $\lambda_2\lambda_2$ at the output.

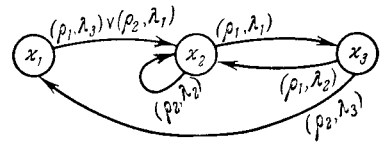

Fig. 3.11.

The matrix C^2 can again be multiplied by C, using analogous rules. The only difference is that the multiplication now involves elements that may contain not only symbol pairs (ρ_s, λ_t), but also pairs of symbol sequences $(\rho_i\rho_j, \lambda_k\lambda_l)$. Multiplication then means the operation of assigning symbols, for example:

$$(\rho_3\rho_5, \lambda_6\lambda_7)(\rho_4, \lambda_2) = (\rho_3\rho_5\rho_4, \lambda_6\lambda_7\lambda_2).$$

In multiplying elements containing disjunction one also uses the distributive law.

In our example the matrix C^3 has the form

$$C^3 = \begin{array}{c|ccc} & x_1 & x_2 & x_3 \\ x_1 & \begin{array}{l}[(\rho_1\rho_1\rho_2, \lambda_3\lambda_1\lambda_3)\vee \\ \vee(\rho_2\rho_1\rho_2, \lambda_1\lambda_1\lambda_3)]\end{array} & \begin{array}{l}[(\rho_1\rho_2\rho_2, \lambda_3\lambda_2\lambda_2)\vee(\rho_2\rho_2\rho_2, \lambda_1\lambda_2\lambda_2)\vee \\ \vee(\rho_1\rho_1\rho_1, \lambda_3\lambda_1\lambda_2)\vee(\rho_2\rho_1\rho_1, \lambda_1\lambda_1\lambda_2)]\end{array} & \begin{array}{l}[(\rho_1\rho_2\rho_1, \lambda_3\lambda_2\lambda_1)\vee \\ \vee(\rho_2\rho_2\rho_1, \lambda_1\lambda_2\lambda_1)]\end{array} \\ x_2 & (\rho_2\rho_1\rho_2, \lambda_2\lambda_1\lambda_3) & \begin{array}{l}[(\rho_1\rho_2\rho_1, \lambda_1\lambda_3\lambda_3)\vee(\rho_1\rho_2\rho_2, \lambda_1\lambda_3\lambda_1)\vee \\ \vee(\rho_2\rho_2\rho_2, \lambda_2\lambda_2\lambda_2)\vee(\rho_1\rho_1\rho_2, \lambda_1\lambda_2\lambda_2)\vee \\ \vee(\rho_2\rho_1\rho_1, \lambda_2\lambda_1\lambda_2)]\end{array} & \begin{array}{l}[(\rho_2\rho_2\rho_1, \lambda_2\lambda_2\lambda_1)\vee \\ \vee(\rho_1\rho_1\rho_1, \lambda_1\lambda_2\lambda_1)]\end{array} \\ x_3 & (\rho_1\rho_1\rho_2, \lambda_2\lambda_1\lambda_3) & \begin{array}{l}[(\rho_2\rho_1\rho_2, \lambda_3\lambda_3\lambda_2)\vee(\rho_2\rho_2\rho_2, \lambda_3\lambda_1\lambda_2)\vee \\ \vee(\rho_1\rho_2\rho_2, \lambda_2\lambda_2\lambda_2)\vee(\rho_1\rho_1\rho_1, \lambda_2\lambda_1\lambda_2)]\end{array} & \begin{array}{l}[(\rho_2\rho_1\rho_1, \lambda_3\lambda_3\lambda_1)\vee \\ \vee(\rho_2\rho_2\rho_1, \lambda_3\lambda_1\lambda_1)\vee \\ \vee(\rho_1\rho_2\rho_1, \lambda_2\lambda_2\lambda_1)]\end{array} \end{array}.$$

The elements of the matrix C^3 thus indicate all the input sequences which can transform one state into another over three discrete moments, as well as the corresponding output sequences. Continuing the process of multiplying matrix C by itself, we finally obtain matrix C^q and thus find all the input sequences which transform state x_i into state x_j over q moments.*

*There are yet other methods of specifying the operation of an automaton. For example, Kobrinskiy and Trakhtenbrot [43] employ the "tree" of an automaton for this purpose, but we shall not use this concept.

3.7. ON THE RESTRICTION OF INPUT SEQUENCES

So far, we have assumed that the input symbol sequences may be random, provided that each of these symbols was contained in the alphabet $\{\rho\}$. Thus, if the alphabet $\{\rho\}$ consists of r symbols, we have at our disposal r^k different sequences of length k.

However, one frequently deals with problems in which one needs to examine only those symbol sequences which satisfy some special conditions. Sequences that satisfy such additional special restrictions are termed *admissible* (or allowable, or legitimate). For example, we could impose any of the following restrictions:

1. The only admissible sequences are those in which even subscripts of ρ alternate with odd ones. Under this restriction, the sequence $\rho_7\rho_2\rho_1\rho_4\rho_7\rho_6\rho_3 \ldots$ would be admissible, whereas the sequence $\rho_7\rho_2\rho_4\rho_1\rho_7 \ldots$ would not be.
2. The only admissible sequences are those in which no two identical symbols ρ are consecutive. In this case, the sequence $\rho_2\rho_7\rho_5\rho_3\rho_8\rho_1 \ldots$ would be admissible, whereas the sequence $\rho_2\rho_7\rho_5\rho_5\rho_3\rho_8\rho_8 \ldots$ would not be.
3. The only admissible sequences are those in which ρ_i is not immediately followed by ρ_j.

The restrictions imposed on sequences are often due to the manner in which the continuous time is divided into discrete intervals. Thus, if the next discrete moment occurs whenever the input is changed, then the restriction on the sequences $\rho(t)$ is that no two identical symbols may be consecutive. Similar restrictions always occur in the other cases in which the timing of the system is synchronized with some input "event."* Thus the restrictions on the input sequences can be of two kinds:

1. They may be imposed by some characteristic of timing of the system, in which case only admissible sequences will appear at the output.
2. They may have no connection with timing, that is, generally speaking, the s-machine can respond to any input sequences $\rho(t)$, but under the operating conditions, only admissible sequences do appear at its input. This distinction is immaterial to us at this point. There are, however, instances where the input sequences cannot be arbitrary but must satisfy some supplementary conditions. We shall discuss this subject later.

*We are relying on the reader's intuition in using the term "event" at this point, but we shall define it at a later stage.

In such cases the basic table alone is not sufficient for the definition of an automaton. Just as it must be supplemented with conditions defining the clock of the system, so in these instances it must be supplemented by the specification of a legitimate input sequence.