Conclusion

We now return to the two fundamental problems posed in the Introduction, namely: (1) Finding out what a finite automaton or a sequential machine can and cannot "do," and (2) the development of techniques for syntheses of devices which are dynamical systems of this class and perform specific tasks. The answers to these problems have been gradually accumulating in the course of our presentation of the theory. We shall now endeavor to combine the solutions scattered through previous chapters into one coherent system.

1. WHAT CAN A FINITE AUTOMATON OR A SEQUENTIAL MACHINE "DO"?

That depends whether the machine in question is autonomous or not.

If the finite automaton is autonomous, then beginning with some cycle it will only generate a periodically recurring sequence of states (the corresponding s-machine can only generate a fixed sequence of outputs). If this is a one-symbol sequence, then the machine will achieve a state of equilibrium within a finite number of cycles. If this is a multisymbol sequence, then the automaton will assume, one after another, all the states corresponding to these sequence, and will continue doing so *ad infinitum*. That is all an autonomous machine can "do."

However, regardless of what this finite periodic sequence of states is, one can always synthesize an autonomous finite automaton which will start to generate this sequence as early as its second cycle. Because of that, and because a fixed cycle of successive operations is characteristic of much of modern technology, dynamical systems which within allowable idealizations may be regarded as autonomous automata are widely used. A very old example of such automata are the animated figurines which go through complex sequences of motions, for instance, writing down a text on a piece of paper, playing predetermined music on an instrument, and so on. Modern examples range from washing machines to automatic lathes, assembly lines and control systems for cyclic operations.

If the automaton is nonautonomous, that is, its input state varies from cycle to cycle, then the answer to the question of what it can or cannot "do" can be formulated in a variety of terms, for example, in the language of representation of events. Indeed, a nonautonomous finite automaton (or s-machine) merely transforms sequences of input symbols into sequences of states (or outputs). Therefore, if we ask what such a machine can or cannot do, we are merely specifying which sequences can be transformed in a given machine and which cannot. But since the number of states (or of outputs, in the case of the s-machine) is finite, our question is equivalent to the following: which specific inputs produce each of the possible states of the automaton (or, each of the possible outputs of the s-machine)? In the terminology of the theory of finite automata this question is formulated as follows: Which events can (and which cannot) be represented by each of the possible states of the automaton (or by each of the outputs of the s-machine)? The exact answer, in terms of the necessary and sufficient conditions for representability of events in the machine, is given by the theorems of Kleene. Kleene's theorems state that only regular events can be represented in a finite automaton, a regular event being the input of sequence belonging to the class of regular sets. Thus in the language of representation of events our question receives an unambiguous answer: A finite automaton can only represent regular events.

Many important input sets encountered in practice are known to be regular. For instance, the following are regular: (a) the set consisting of any finite number of input sequences of finite length; (b) the set of any periodically repeating input sequences; (c) the set of infinitely long sequences which always terminate in specified finite sequences; and so on. However, in general, if we are faced with an infinitely long set of input sequences, we do not know *a priori* whether this set is regular or not. This is because we only have techniques for *generating* regular sets (by induction), but lack effective solution for the inverse problem of finding out whether a given set is regular or not. Thus, even though the theorems of Kleene do answer the question as to what a finite automaton can do, the answer is not an effective one. Present research attempts to construct other languages in which the answer could be given more effectively. This language problem is also of cardinal importance in the initial steps of the synthesis of automata, that is, it also figures in the second of the two problems formulated above.

Our class of dynamical systems consisting of "finite automaton" and "sequential machine," can be extended by providing the machines with an infinite memory (this can be done by letting the machine

have an infinite number of states, or providing it with an infinite tape, and so on). This gives a broader class of abstract systems—the Turing machines. The answer to the question, "what can they do?" is much simpler: they can realize any *a priori* specified algorithm. Now in modern mathematics the algorithm itself is defined as a computation of values of some recursive function. And since we know so precisely and unambiguously what a Turing machine can do, we can use this machine to define the concept of the algorithm. We thus close our chain of reasoning with the statement: an algorithm is any process which can be realized in a finite automaton supplemented by an infinite memory, that is, in a Turing machine.

2. THE SYNTHESIS OF A PRACTICAL DEVICE REALIZING A FINITE AUTOMATON OR SEQUENTIAL MACHINE

If we wish to sample the input and the output of a system only at some specified discrete times (where these instants can either be specified *a priori*, or may be the result of the very operation of the system), then we have every reason to suspect that the device embodying our requirements will be a finite automaton or an s-machine. Since the object of the design is to ensure the generation of desired outputs in response to specified inputs, we could specify our device by enumerating all the possible input-output relationships. If this enumeration results in a finite list of noncontradictory sequences of a finite length,* then we can be sure that our specification can be embodied by a finite automaton or an s-machine. Furthermore given these input-output relationships, we can derive from them the basic table of the finite automaton and the table of the output converter; together, these form one of the possible s-machines realizing the specification (the algorithm for the synthesis of such tables is described in Section 8.2**).

It is much more difficult to ensure the generation of specific outputs in response to infinitely long inputs. Such cases are frequently encountered in practice, when the duration of the operation

*We assume *a priori* that either there can be no other inputs, or that if there are such inputs, then the output may be arbitrary or, finally, that any other input will be signaled by the appearance of some symbol indicating that input.

**This type of definition is frequently encountered in the design of relay circuits, where the required input-output relationships may be enumerated, for example, by means of the so-called switching tables.

of the device being designed cannot be predetermined, and when the outputs must indicate some general properties of the input sequences, as from the first beat of the operation of the machine. In such cases a basic question arises: In what way can the relationship between the infinite input and output sequences be defined, since a direct enumeration of the sequences is impracticable in this instance. No matter how this relationship has been defined, it boils down in the end to the definition of an algorithm enabling it to be established for each beat what the input and the output symbols are in that beat.

If definition of the relationship between the infinite sequences is not restricted in some way, then from the very start we again come up against the same difficulty as that discussed at the end of the preceding section, i.e., there is no effective method of establishing whether the event which is to be represented by the automaton is regular. This means that if the language in which the definition is formulated is not restricted in some way, then there is no way of even establishing whether some finite automaton or an s-machine is capable of realizing the definition. Therefore there is no point in talking of a method for the synthesis of an automaton or an s-machine realizing the definition. Once more as a central problem arises that of finding a language sufficiently broad for the definitions of an automaton or an s-machine to be expressed in it, of great importance in technology; this language is to be such that there exist recognition algorithms as to whether there is an automaton or an s-machine capable of realizing the definition, and, when the answer is in the affirmative, the algorithms are to enable the required s-machine to be constructed.

Accordingly, in the formulation of definitions of an automaton (in the case of infinite sequences) special methods are employed (or, in other words, special languages) to avoid this difficulty. One of such methods is to write down the definition directly using the description of the regular events which the automaton is to represent, rather than the description of the correspondence between the input and the output sequences. This method is described in Section 8.4, where an effective method is indicated for the construction of the basic table of the automaton and the table of the converter, which together form the s-machine representing the given events.

Another, considerably less economical (as regards the number of the states of the automaton required) method has been described in Section 7.4, in the course of the proof of Kleene's theorems.

Other languages are also known, characterized by the fact that every definition which is expressible in the language is known to be

realized by an s-machine, and the corresponding s-machine (i.e., its tables) is effectively constructible from that expression. An example is Trakhtenbrot's predicate language, which has been briefly mentioned in the present book. The use of these languages is, in essence, based on the assumption that man is capable of nonalgorithmically (creatively) solving the problem indicated above, translating the definitions from the ordinary general language in which he thinks, into a special language in which the problem of recognition of representability of events does not arise. If one was unsuccessful in expressing a definition in such a language, the question remains open as to whether this has been caused by the fact that the definition cannot be translated into that language, and therefore realized by an s-machine, or because one failed to do so "creatively."

It follows from the foregoing that the first stage of the synthesis is in some cases carried out according to standard rules, and in some other cases it, in principle, requires creative action; but, in any case, provided the definition is realizable, the result of the first stage of the synthesis is the table of the automaton and the converter table, which form one of the s-machines realizing the definition. An s-machine so constructed is not unique; generally speaking, there is a set of other s-machines fulfilling the same definition, i.e., those equivalent to the one constructed by us, or representing it. Such s-machines may differ in the number n of the symbols in the state alphabet $\{\varkappa\}$, i.e., in the number of rows in the basic table of the automaton. The smaller the number n, the simpler is subsequent construction or the scheme of the real machine. Accordingly, the next, the second stage of the synthesis is the minimization of the machine obtained, i.e., the construction of an s-machine equivalent to the one evolved in the first stage of the synthesis and, at the same time, having the least possible number of states n.

The solution of the minimization problem depends essentially on the set of sequences which may appear at the input of the automaton during its operation. The set is of course, indicated in the original definition.

The simplest case is one where the set of the input sequences is not restricted in any way, i.e., when any sequence may appear at the input of the automaton. In this case the problem of the construction of a minimal s-machine, in the sense indicated, has been fully solved, i.e., the necessary and sufficient conditions for the minimization have been found. A method realizing the construction of a minimal S-machine involves breaking down the connection matrix into certain submatrices; it has been described in Section 9.6.

Matters are rather more complicated when the set of possible input sequences is restricted in some way. Assuming that the

constraints are arbitrary, i.e., that some arbitrary algorithm is given enabling it to be established whether a sequence satisfies the given constraints, the minimization problem turns out to be unsolvable (see Section 9.2). Accordingly, there is no minimization method suitable for any constraints and one can only attempt to find the necessary and sufficient conditions of minimization for some given particular form of constraints. However, excluding a complete sorting, even in the case of the most frequently encountered forms of constraints (e.g., when the constraint consists in only sequences of a given length appearing at the input, or sequences of any length but containing no identical symbols in succession, etc.), such necessary and sufficient conditions have not so far been found. Some observations on minimization in such cases were produced in Sections 9.4 and 9.7.

We know of only two problems with constraints imposed on the input sequences which have a full solution. These are the problem of construction of a minimal s-machine in the case when it is to operate as a finite automaton and the input sequences contain no identical symbols in succession, and the problem of construction of a minimal s-machine in the case of Aufenkamp-type constraints (see Section 9.8).

So, as a result of the second stage of the synthesis, provided it proved to be realizable, a basic table of an automaton and a converter table are constructed, which together determine an s-machine fulfilling the given definition and, at the same time have the least possible number of states. In the general case this completes the formation of the basic table of the automaton and the converter table and it is possible to pass on to the third stage of the synthesis, which consists of the construction of the abstract structure of the s-machine being designed. However, there is a particular case, frequently encountered in practice, where the input sequences are restricted, and some further work is required to construct a minimal s-machine. We are referring to the case where the rhythm of the operation of the machine being designed is determined by the change of the states at the input and there are, therefore, no input sequences containing identical symbols in succession.

In this case further work in constructing the tables of the s-machine is dependent on the technical procedures used to construct the tables. More precisely, it is essential to lay down beforehand which of two possible ways is to be followed. The first way is that of applying delay elements to a beat signal fed from outside, with special devices signaling the occurrence of a beat (i.e., a change in the input state). The second way does not require the application

of any special delay elements, but utilizes the fact that real elements have a certain inherent delay in operation and permit the construction of a machine by making use of steady states.

If the first way is used, the second stage of the synthesis described above, as far as it can be carried out to the end bearing in mind the constraints imposed on the input sequences, completes the construction of the tables of the s-machine and is immediately followed by the third stage of the synthesis: the transition to an abstract structure (see further on).

If the second way is used, further processing of the tables of the automaton and the converter is necessary. This means constructing the tables of another s-machine, which operates at a faster rhythm (as determined by the delay time in the elements employed in the construction of the s-machine), and which reproduces in its steady state the s-machine being designed, operating at a 'slow' rhythm which is determined by the moments when there is a change of state at the input.

To do this a "fast" machine satisfying this condition is first constructed, and this machine is then minimized, i.e., the second stage of the synthesis is repeated (for further details see Sections 10.2 and 10.3). In the end, by this second way we also obtain the tables of a minimal s-machine and can once more pass on to the third stage of the synthesis.

At the third stage of the synthesis an abstract structure is constructed, i.e., from the tables of the s-machine obtained in the preceding stage, the logical equations of an abstract structure representing this s-machine are set up, i.e., logical functions F_i and Φ_j in equations of the form*

$$x_i^p = F_i\left(x_1^{p-1}, x_2^{p-1}, \ldots, x_n^{p-1}; u_1^{p-1}, u_2^{p-1}, \ldots, u_s^{p-1}\right), \quad i = 1, 2, \ldots, n,$$

$$z_j^p = \Phi_j\left(x_1^p, x_2^p, \ldots, x_n^p; u_1^p, u_2^p, \ldots, u_s^p\right), \quad j = 1, 2, \ldots, l.$$

Depending on the number of states in the elements at our disposal for the construction of the machine, the functions will be those of two-, three-, and generally of m-valued logic. The method of coding and the construction of these functions is given in Section 4.2.

In the case of construction of an automaton based on steady states, the coding and construction of the functions F_i and Φ_j are given in Section 5.4.

As a result of the third stage of the synthesis the problem is reduced to one which is much more familiar to the project engineer,

*The equations are written out for a machine of the P-P type.

that of realizing a system of logical relations with the technical means at his disposal. At this point, broadly speaking, the problems of abstract synthesis are no longer relevant. Consequently, there are no problems of the general theory of finite automata and sequential machines which are applicable. From this moment on, the problem belongs to the realm of technical realizations of the abstract structure which has been obtained.

The problems arising in this connection are studied in the theory of switching circuits and the theory of logical systems, in the narrow sense of these terms. The problems solvable by these theories have hardly been considered in this book, or a mere mention of them has been made in passing.

If the subsequent construction of the scheme is based on delay elements, then the number of such elements is predetermined by the number of the equations in the abstract structure, which is known to be minimal if the second stage of the synthesis has been carried out to the end. It was just this obtaining of a scheme with a minimal number of elements of delay that constituted the second stage of the synthesis. The problem of technical realization then reduces to the construction of logical converters realizing the functions F_i and Φ_j contained in the right-hand sides of the equations of the abstract structure. From one and the same set of logical elements the converters may be constructed in various ways. This too has its own minimization problems, but these in fact concern converters and not sequential machines, i.e., they relate to statics and not to dynamics, and, therefore, only a brief mention of them has been made in Section 2.6.

If the available set of logical elements does not contain a ready-made delay element, this does not exclude the possibility of constructing schemes, since the delay element itself, being the simplest automaton, can be constructed from the elements of the set, for example, using the steady states of equilibrium.

If the entire machine is constructed using steady states, i.e., without special delay elements for the beat signal fed from without, this means that a fast machine is to be constructed according to the abstract structure obtained at the end of the third stage; elements which have an inherent delay (e.g., repeaters) serve as delayers. In particular, in the common case when the schemes are assembled from relays, the delayer for the "fast" s-machine will be the intermediate relays, while the converters F_i and Φ_j are formed from chains of contacts of the input and intermediate relays.

With such a circuit construction (using steady states), there arise additional technical difficulties, in connection with the fact

that in a nonsynchronized systems the delay time of the elements is not strictly the same. This leads to the danger of relay "competition" arising, which may sometimes result in an incorrect operation of the circuit. In such cases the danger of competition is obviated by special circuits, called realizations; in these circuits either not more than one relay operates in each beat, or the feedback circuits are artificially cut off at the switching moments.

There are various methods for the construction of realizations, only one of which has been briefly described in Section 5.4, since realization problems do not relate to the general theory of finite automata and sequential machines. In the construction of circuits using delayers, with the beat signal to them from outside, there is no likelihood of competition arising and, therefore, the realization problem does not arise.