2

Engineering Applications of Propositional Calculus

2.1. COMBINATIONAL RELAY SWITCHING CIRCUITS

We have already said that the promise of mathematical logic in engineering design first became apparent during the analysis of electrical relay switching circuits. In time, it became progressively more evident that this logic is not only applicable to *the analysis of relay switching* circuits but that the operation of such circuits mirrors the postulates of the logic. The result of this discovery was the relay switching theory. Then, when contactless devices that perform the same functions as relay switches came into existence, the special *theory of relay circuits* was extended into a general theory of switching systems.

We shall now examine this most conspicuous example of application of logic to engineering, concentrating on the so-called combinational relay switching circuits.

Every electric relay switching circuit contains two types of converters: electrical-to-mechanical and *mechanical-to-electrical*. The electromechanical relay converts electrical input signals into a mechanical displacement of its contacts. On the other hand, the mechanical-to-electrical converter is an electrical network comprising contacts and relay coils: it converts the mechanical displacement of its (input) contacts into electrical output signals (currents flowing in the coils of the relays). Connection of outputs of converters of one type to the inputs of converters of the other type gives a variety of relay switching networks.

The simplest electromechanical relay consists of a coil *1*, a core *2*, an armature *3*, and two groups of contacts: normally closed *4'*, and normally open *4''* (Fig. 2.1,a). If a current larger than the actuating current i_2 (Fig. 2.1,b) flows in the coil, the armature is

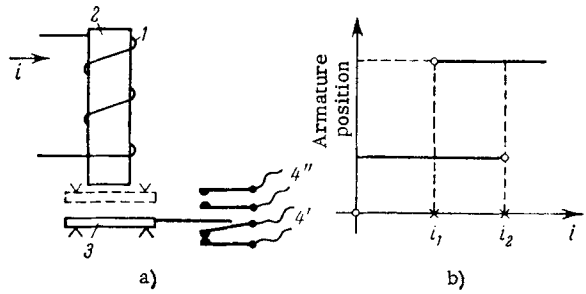

Fig. 2.1.

attracted to the core, and this causes all the normally open contacts to close and all the normally closed contacts to open. On the other hand, if a current smaller than i_1 flows in the coil (in particular, when the coil is de-energized, the armature recedes from the core, causing the normally closed contacts to close and the normally open contacts to open. To avoid the complications arising in transient states, we shall consider only the stable states of the relay, that is, states when the coil current i has either of the following values: $i < i_1$ or $i > i_2$. Thus the relay has two stable states or, to say it in another way, there are two states which are characteristic for all the elements of the relay. We are, however, interested only in the coil and in the contacts. Each contact has two states: closed and open. We shall designate these states by 1 and 0, respectively; that is, we shall treat the state of a contact as a binary logical variable assuming these values.

The coil also has two states. The first of these occurs at $i > i_2$, and we denote it by 1. In the second state, denoted by 0, the coil is de-energized ($i < i_1$). Thus the coil states can also be represented by a logical variable that can be either 0 or 1. The physical significance of these values is shown in Table 2.1.

Table 2.1

Component	Symbol	
	1	0
Contact	closed	open
Coil	energized	de-energized

The state of a relay contact is goverened by the state of the coil in the following manner:

For a normally open contact $x = X$,
For a normally closed contact $x' = \overline{X} = \bar{x}$.

where X, x, x' are the logical variables specifying the states of the coil, of the normally open contact and of the normally closed contact, respectively.

If a relay has more than one coil, it becomes convenient to an "equivalent coil." Thus, let the relay have two coils (X_1 and X_2) and let these be so connected that the relay shall operate only if both are energized, that is, if $X_1 \& X_2 = 1$. Let us now imagine a coil X_e, such that it will cause the relay to operate only if $X_1 \& X_2 = 1$. Obviously the action of our equivalent coil, which is related to that of the actual coils by

$$X_e = X_1 \& X_2$$

and which governs the state of the relay contacts in accordance with

$$x = X_e \quad \text{(for normally open contacts)}$$
$$x' = \overline{X}_e = \bar{x} \quad \text{(for normally closed contacts)},$$

is completely equivalent to the action of the two actual coils.

So far, we have discussed a relay with two coils. By the same reasoning, we can imagine a relay with m coils connected so that the relay operates only at certain combinations of the 1 and 0 states of the constituent coils.

A relay circuit incorporating m coils in a specific arrangement is described by a logical function specific to this circuit:

$$X_e = M(X_1, X_2, \ldots, X_m).$$

However, this specificity does not change the general relationship between the contacts of the relay and its equivalent coil.

We shall now turn to the representation of the mechanical-to-electrical converter, that is, of contacts connected to relay coils. Let us start with the case when the circuit (Figs. 2.2, a and 2.2,a′) consists of a contact *1* of an input relay and a coil *2* of another relay (the output relay), the coil being either in series (Fig. 2.2,a) or in parallel (Fig. 2.2,a′) with the contact.

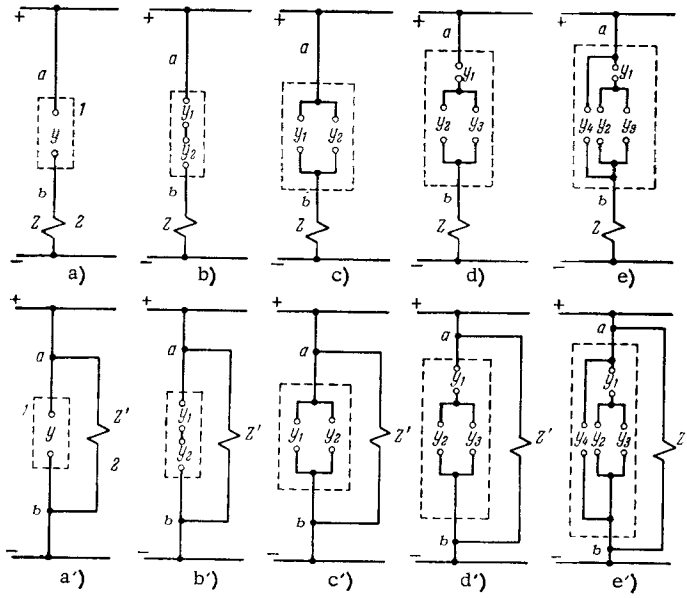

Fig. 2.2.

Retaining the symbols 0 and 1 for the states of the contacts and the coils, we obtain $Z = y$ for a coil connected in series and $Z' = \overline{y} = \overline{Z}$ for a coil connected in parallel; here, y is a logical variable specifying the state of the contact, and Z and Z' specify the states of the coils connected with it in series and in parallel, respectively.

In the usual practical case, we do not deal with a single contact I but with a group of contacts $y_1, y_2, \ldots, y_n$ belonging to several relays, all combined into an electrical network. We shall call an electrical network incorporating contacts a *switching network*. The dashed lines in Figs. 2.2,b-e enclose examples of such networks.

Again, it will be convenient to use an equivalent variable—the equivalent contact y_e—whose properties are analogous to those of the equivalent coil in a multiple-coil relay. For example, with the two contacts y_1 and y_2 of Fig. 2.2,b connected in series, the circuit in section ab will be closed only if $y_1 \& y_2 = 1$. By introducing the equivalent contact

$$y_e = y_1 \& y_2$$

and setting up the relationships

$$Z = y_e \qquad \text{for series connection Fig. 2.2,b)}$$

$$Z' = \bar{y}_e = \bar{Z} \text{ for parallel connection (Fig. 2.2,b')},$$

we preserve all the characteristics of the circuit.

Figure 2.2 also shows some other ways of arranging contacts into circuits; we thus obtain functions

$$y_e = y_1 \vee y_2 \qquad \text{(Fig. 2.2,c,c')},$$
$$y_e = y_1 \,\&\, (y_2 \vee y_3) \qquad \text{(Fig. 2.2,d,d')},$$
$$y_e = [y_1 \,\&\, (y_2 \vee y_3)] \vee y_4 \qquad \text{(Fig. 2.2,e,e')}.$$

In the general case,

$$y_e = N(y_1, y_2, \ldots, y_n);$$

In this generalized function, the relationship between the coils and the equivalent contact remains the same as that specified above (the matching condition N must, of course, reflect the actual arrangement of the contacts in the switching network when the function is used to represent a specific circuit). The physical meaning of y_e is that of the conductivity of a two-terminal network containing the given switching circuit.

Each of our converters has the ability to detect; that is, it exhibits a directional effect. In the electromechanical converter—the "relay with contacts"—the contact state is governed by the coil state, but the contacts have no effect on the coil state. In the mechanical-to-electrical converter—the "contact network with coils"—the coil state is governed by the states of the contacts, on which the coil has no effect. This property of converters allows us to treat them as devices with variable inputs and outputs. The input variables $X_1, \ldots, X_m$ of the electromechanical converter are the states of the relay coils (energized or de-energized); the two output variables x and x' of this device are the states of the two different contacts (normally open or closed). As already stated, this device may be treated as consisting of two series-connected subunits: The first performs the logical function

$$X_e = M(X_1, \ldots, X_m),$$

while the second realizes the functions

$$x = X_e, \quad x' = \bar{X}_e = \bar{x}.$$

In the mechanical-to-electrical converter the states of the contacts of the input relay act as input variables $y_1, \ldots, y_n$, and the coil states of the two output relays act as output variables Z and Z'. This device may again be treated as consisting of two series-connected subunits, the first of which realizes the logical function

$$y_e = N(y_1, \ldots, y_n),$$

while the second performs the functions

$$Z = y_e, \quad Z' = \bar{y}_e = \bar{Z}.$$

We see now that the two types of converters have identical properties. Any relay switching circuit may be broken down into units having the above-described properties.

We shall assume that our combinational relay switching circuits obey the following conditions: they consist of instantaneously actuated, ideal relays; and they have no feedback loops; that is, they consist only of subunits exhibiting a direction effect.

Figure 2.3,a shows the schematic of such a combinational relay switching circuit and Fig. 2.3,b shows the corresponding block diagram; it can be seen that the latter has no feedback loops. Contrast this with the schematic diagram shown in Fig. 2.4,a: its block diagram (Fig. 2.4,b) does show a feedback loop. The operation of such a circuit cannot be analyzed without taking into account the relay-actuating time.

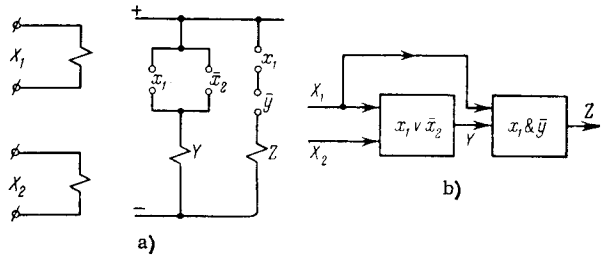

Fig. 2.3.

In drawing relay switching circuits we usually identify each coil by means of the corresponding logical variable; contacts are usually identified by an expression that specifies only the contact state (in terms of the state of the coil governing it).

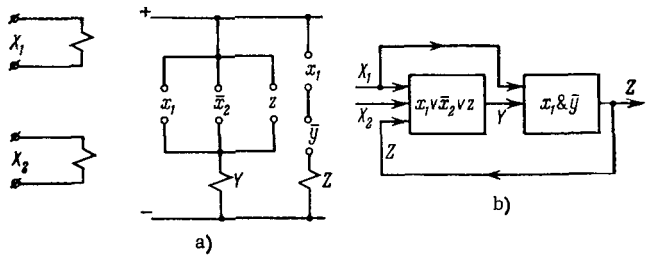

Fig. 2.4.

2.2. ANALYSIS OF COMBINATIONAL RELAY SWITCHING CIRCUITS

Consider the following problem: Given a combinational relay switching circuit, it is required to find its mathematical description, that is, to determine the logical function performed by this circuit.

We shall analyze only circuits with single-coil relays (these are the most common switching circuits). We have already seen (Section 2.1) that for these relays

$$x = X \text{ for in the case of normally open contacts}$$

$$x' = \bar{X} = \bar{x} \text{ for in the case of normally closed contacts,}$$

where X, x, and x' are variables specifying, respectively, the states of the coil, the normally open contact, and the normally closed contact.

This means that a single-coil relay operating alone will perform either repetition or negation.

The great variety of combinational switching circuits that can be synthesized is due to use of different arrangements of normally open and normally closed contacts of single-coil relays. The resulting switching network can be represented by the relationship

$$x_e = F(x_1, x'_1, \ldots, x_n, x'_n),$$

where x_i, x'_i, and x_e are variables specifying, respectively, the states of the contacts of the ith relay and of the equivalent contact.

However complex and involved a switching network may be, it is always possible to represent it in terms of

$$x_e = F(x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n).$$

Such a formula may be derived by a procedure which generalizes the obvious fact that if two contacts x_1 and x_2 are connected in series, we have $x_e = x_1 \& x_2$, while for contacts connected in parallel we have $x_e = x_1 \vee x_2$.

To clarify the principles on which this procedure is based, consider an example. The two-terminal switching circuit to be analyzed is shown in Fig. 2.5,a. We shall gradually simplify this circuit by introducing equivalent contacts. To start with, we shall eliminate all the chains of series-connected contacts. Putting

$$x_6 = x_3 \& \bar{x}_5; \quad x_7 = x_3 \& x_4; \quad x_8 = x_5 \& x_1;$$
$$x_9 = x_2 \& \bar{x}_1; \quad x_{10} = x_2 \& x_5; \quad x_{11} = x_3 \& \bar{x}_2; \quad x_{12} = x_2 \& x_5,$$

we transform the original circuit into its equivalent shown in Fig. 2.5,b.

The next step is to eliminate all the groups of contacts connected in parallel. To do this we write

$$x_{13} = x_2 \vee x_7; \quad x_{14} = \bar{x}_1 \vee \bar{x}_3; \quad x_{15} = x_8 \vee x_6; \quad x_{16} = x_9 \vee x_1,$$

or, using the notation already introduced,

$$x_{13} = x_2 \vee (x_3 \& x_4); \qquad x_{14} = \bar{x}_1 \vee \bar{x}_3;$$
$$x_{15} = (x_1 \& x_5) \vee (x_3 \& \bar{x}_5); \quad x_{16} = x_1 \vee (x_2 \& \bar{x}_4).$$

We then obtain the circuit shown in Fig. 2.5,c.

Again, we shall eliminate the chains of series-connected contacts in this new circuit. We do this by means of the following equivalents:

$$x_{17} = x_1 \& x_{13} = x_1 \& [x_2 \vee (x_3 \& x_4)] = (x_1 \& x_2) \vee (x_1 \& x_3 \& x_4),$$
$$x_{18} = x_{14} \& x_{15} = (\bar{x}_1 \vee \bar{x}_3) \& [(x_3 \& \bar{x}_5) \vee (x_1 \& x_5)] =$$
$$= (\bar{x}_1 \& x_3 \& \bar{x}_5) \vee (x_1 \& \bar{x}_3 \& \bar{x}_5),$$
$$x_{19} = x_{16} \& x_{10} = [x_1 \vee (x_2 \& \bar{x}_4)] \& x_2 \& x_5 =$$
$$= (x_1 \& x_2 \& x_5) \vee (\bar{x}_4 \& x_2 \& x_5).$$

We thus obtain the circuit shown in Fig. 2.5,d. This circuit cannot be further simplified by the above methods of elimination.

Now, let us number all the modes of this circuit, using identical numbers for those nodes which are directly interconnected (without intervening contacts). We thus have Fig. 2.5,d, with nodes 1, 2, ..., m (in our case, $m = 4$). It is now convenient to transform this circuit

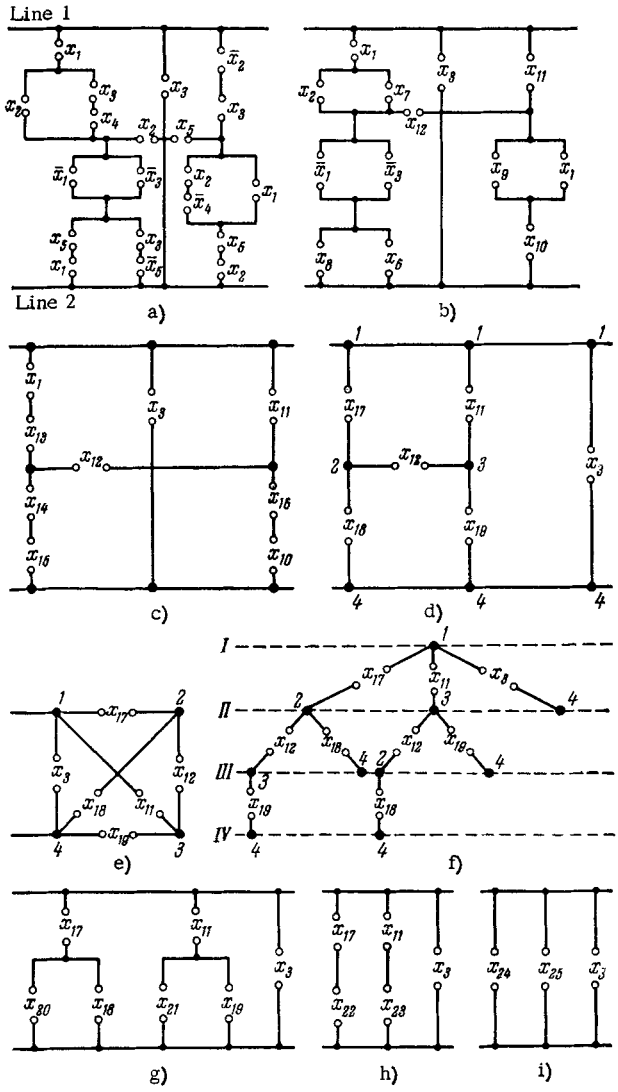

Fig. 2.5.

into the form of Fig. 2.5,e, which is obtained by combining all the nodes bearing identical numbers.

The circuit in Fig. 2.4,e can also be presented as a "tree" (Fig. 2.5,f) constructed in the following manner. We draw several tiers and assign to them the numbers corresponding to the m nodes of Fig. 2.5,e; we thus have tiers *I*, *II*, *III*, and *IV*. Node *1* is placed in

the first tier. From it we draw a cluster of $m-1$ branches, all terminating in the second tier. The ends of these branches are marked with the numbers of the remaining nodes of the circuit [that is, nodes which together with the initial node of the cluster (node 1) constitute the complete set]; in our case, that means the numbers 2, 3, and 4.

Next, we use each of these $m-1$ nodes of the second tier (but not the node m, that is, 4) as the origin of a cluster of $m-2$ branches which terminate in the third tier. Thus, we obtain two clusters of branches, originating at nodes 2 and 3, respectively. The third-tier ends of the cluster drawn from node 2 are given the numbers of all the second-tier nodes, with the exception of 2 (that is, the ends of this cluster carry the numbers 3 and 4). In the same way, we number the terminals of the cluster of $m-2$ branches originating at node 3 (but here we omit 3), and so on.

In the next step, each third-tier node, except those designated by m, serves as the origin of a cluster of $m-3$ branches joining the third and the fourth tiers. The fourth-tier terminals are numbered by the same procedure as the third. It is readily seen that the last (or mth) tier will now hold only nodes designated by m, from which no further branches can be drawn (dead-end nodes).

We now have a "tree" in which each "branch" connecting two nodes corresponds to the wire performing the same function in the circuit of Fig. 2.5,e.

The switching network of Fig. 2.5,f traces all the paths leading from node 1 to node 4 and is equivalent to that of Fig. 2.5,e. Now, we can eliminate all groups of series-connected contacts by using

$$x_{20} = x_{12} \& x_{19} = (x_1 \& x_2 \& x_5) \vee (\bar{x}_4 \& x_2 \& x_5),$$
$$x_{21} = x_{12} \& x_{18} = x_1 \& x_2 \& x_5 \& \bar{x}_3$$

and we get the circuit shown in Fig. 2.5,g. We then eliminate the groups of parallel contacts by using

$$x_{22} = x_{20} \vee x_{18} =$$
$$= (x_1 \& x_2 \& x_5) \vee (\bar{x}_4 \& x_2 \& x_5) \vee (\bar{x}_1 \& x_3 \& \bar{x}_5) \vee (x_1 \& \bar{x}_3 \& x_5),$$
$$x_{23} = x_{21} \vee x_{19} = (x_1 \& x_2 \& x_5 \& \bar{x}_3) \vee (\bar{x}_4 \& x_2 \& x_5)$$

and we get the diagram of Fig. 2.5,h.

Again, we eliminate the groups of series-connected contacts, this time using the expressions

$$x_{24} = x_{22} \& x_{17} = x_1 \& x_2 \& x_5,$$
$$x_{25} = x_{23} \& x_{11} = 0.$$

We thus obtain the circuit of Fig. 2.5,i. This last circuit can be represented by the function

$$x_e = x_{24} \vee x_{25} \vee x_3 = (x_1 \,\&\, x_2 \,\&\, x_5) \vee x_3.$$

This is the logical function performed by the original circuit of Fig. 2.5,a. But it is also performed by the circuit of Fig. 2.6; that is, the circuit shown in Fig. 2.6 is equivalent to that of Fig. 2.5,a.

Incidentally, contact x_4 of Fig. 2.5,a is absent from the equivalent circuit of Fig. 2.6; this means that it serves no purpose in the circuit, a fact which can be readily verified. Indeed, Fig. 2.5,a shows that the circuit can be closed by closing contact x_3 alone. If x_3 is open, then we can close the circuit by closing x_1, x_2, and x_5, and therefore do not need x_4.

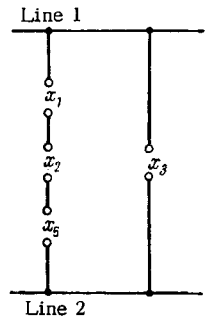

Fig. 2.6.

We have considered only one example of a procedure which allows us to derive the logical function corresponding to any given circuit. This procedure is called the *analysis* of the circuit. The simplification of the starting functions, arrived at by means of Boolean algebra, results in circuits that are equivalent to the starting networks but have the great advantage of being much simpler.

In our example we were able to simplify the function to such an extent that the practical switching circuit performing it could be drawn without further ado. This, however, is not always possible, especially in the more complex cases. In these cases, the logical function so derived is but the starting point in the *synthesis* of a switching network capable of realizing it.

2.3. SYNTHESIS OF COMBINATIONAL RELAY SWITCHING CIRCUITS

Assume that we are given some logical function and that we are required to construct a relay circuit embodying it. We shall discuss only circuits consisting of single-coil relays and an output relay whose coil is connected in series with the contact network. We shall assume that the logical function is given by a table enumerating all possible combinations of values of the variables, each such combination corresponding to some value of the function (such tables were described in Section 1.3).

Table 2.2

x_1	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1
x_3	0	0	0	0	1	1	1	1
y	0	1	1	0	1	0	0	1

Consider the function of three variables $y = L(x_1, x_2, x_3)$, given by Table 2.2. Its complete disjunctive normal form (see Section 1.3) is:

$$y = (x_1 \& \bar{x}_2 \& \bar{x}_3) \vee (\bar{x}_1 \& x_2 \& \bar{x}_3) \vee (\bar{x}_1 \& \bar{x}_2 \& x_3) \vee (x_1 \& x_2 \& x_3).$$

This form is directly translatable into a practical switching network by means of the following rules:

As in the case of analysis, a normally open contact is made to correspond to x_i (in this particular function, $i = 1, 2, 3$), and a normally closed contact to $\bar{x}_i$. Each conjunctive term (in parentheses) becomes a chain of series-connected contacts, whose states are specified by variables contained in the parenthesis. The complete disjunctive normal form corresponds to parallel connection of the above-mentioned series chains.

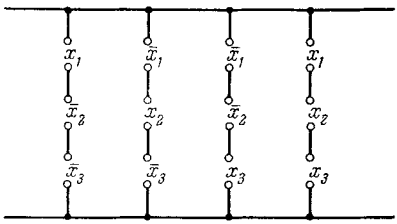

Fig. 2.7.

Applying these rules to our example, we obtain the circuit shown in Fig. 2.7.

This canonical technique will translate any logical function into a series-parallel switching network. In some cases it is then possible to simplify the result, obtaining a circuit containing a smaller number of elements (see the example of Section 2.2).

There are, however, other techniques, which by abandoning the canonical technique and the series-parallel network in favor of the so-called bridge circuit, lead directly to more economical systems embodying a given logical function.*

We shall now present, without proof, one such technique—devised by A. Sh. Blokh.** Returning to Table 2.2, which defined our logical function, let us write out the row containing the values of y, that

*However, these techniques still do not assure circuits with a minimum number of elements. The synthesis of circuits that are "minimum" with respect to some design variable is a serious problem for which there is, at present, no final solution. For a discussion, see Section 2.6.

**See [7], which contains all the necessary proofs. However, Blokh does not use the term "canonical" in the same sense as we do.

is

$$01101001.$$

We then group the symbols of this row into pairs as follows:

$$\underline{01}\ \underline{10}\ \underline{10}\ \underline{01}.$$

Should any pair contain two identical symbols, that symbol is written below that pair (there are no such pairs in this example). Otherwise, we assign the symbols 2 and 3 to the remaining pairs as follows:

$$\begin{matrix}\underline{01} & \underline{10} & \underline{10} & \underline{01}\\ \overline{2} & \overline{3} & \overline{3} & \overline{2}\end{matrix}.$$

These new symbols are again grouped into pairs; the new pairs are again grouped as above, and we assign the symbol 4 to the group 23 and the symbol 5 to group 32. The new symbols are again grouped, and so on, until a complete triangular matrix is obtained. This matrix will always have $k+1$ rows, where k is the number of arguments of the function. Thus the function of Table 2.2 yields the matrix

$$\begin{matrix}\underline{01} & \underline{10} & \underline{10} & \underline{01}\\ \overline{2} & \overline{3} & \overline{3} & \overline{2}\\ \underline{\quad 4 \quad} & & \underline{\quad 5 \quad} & \\ \underline{\qquad 6 \qquad} & & & \end{matrix}.$$

Now we proceed with the design of the switching network. To start with, we draw one horizontal line for each row of the matrix (in this case, $k+1=4$). We then enter each of the elements of the matrix as a point on the corresponding line, copying the respective numeral above that point; we thus obtain a set of nodes which we join into a "tree." In drawing the tree, we omit branches that lead to nodes denoted by 0 (see Fig. 2.8). On the branches originating in the lowest tier we place the contacts of the third relay (x_3 and $\bar{x}_3$), with $\bar{x}_3$ on the left- and x_3 on the right-hand branch. Similarly, contacts $\bar{x}_2$ and x_2 are positioned on the branches originating in the second lowest tier, while contacts $\bar{x}_1$ and x_1 are located on the branches starting from the third tier. If the tree contains a branch joining two nodes denoted by the same numeral, then the nodes are short-circuited (no contact is placed on the branch). For our example, we obtain the circuit shown in Fig. 2.9.

For all practical purposes, we now have a network performing the given function. This network may be simplified to its final form

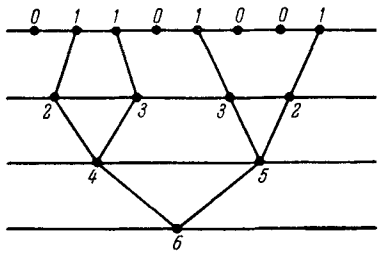

Fig. 2.8.

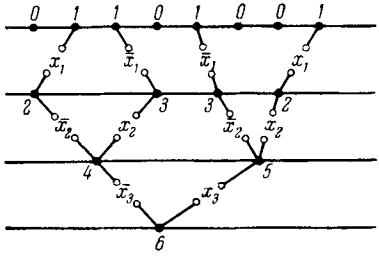

Fig. 2.9.

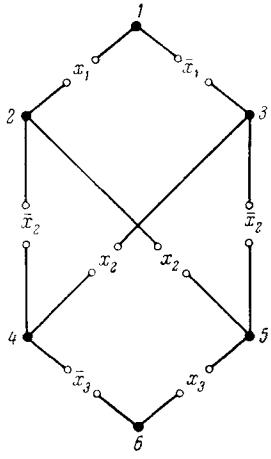

Fig. 2.10.

by combining the nodes denoted by the same numeral, thus reducing the number of component contacts. Our final design will then be the bridge circuit of Fig. 2.10.

The above technique yields a combinational relay switching circuit for any given logical function. This circuit usually contains a smaller number of elements than that synthesized by means of the canonical method employing the full normal disjunctive form of the function.

2.4. OTHER METHODS FOR CONVERTING LOGICAL FUNCTIONS INTO PRACTICAL DEVICES

Aside from electromechanical relays, there are other practical devices embodying logical functions, that is, capable of executing the operations of propositional calculus. We shall now give a few examples of these.

a) Diode Logic

A diode is an element with a nonlinear characteristic such that a flow (an electric current, a stream of air or of liquid, or any other flux) can pass through it in one direction virtually without resistance while a practically infinite resistance to this flow is offered in the opposite direction. Thus, the diode acts as a gate, allowing flow in one direction and blocking it in the other. In diagrams it is usually represented by the symbol shown in Fig. 2.11, where the triangle points in the allowed direction of flow.

In relay switching circuits, the input of the logical variables X_1, X_2, and so on, is accomplished by feeding current to the relay input. Wherever a negation (complement) of these variables is desired, one employs a normally closed contact. However, this cannot be done with diode circuits, because these circuits are incapable of performing the operation of negation.

Fig. 2.11.

For this reason, not only the variables x_1, x_2, and so on, but also their negations (complements) $\bar{x}_1$, $\bar{x}_2$, and so on, must be fed as inputs. These negations are performed outside the diode circuit by other devices, for instance, by electromechanical relays.

We shall now show how any logical function can be embodied in circuits employing only diodes and linear resistances. Let the function be given in its complete disjunctive normal form

$$y = y_1 \vee y_2 \vee y_3 \vee y_4 =$$
$$= (x_1 \& \bar{x}_2 \& \bar{x}_3) \vee (\bar{x}_1 \& x_2 \& \bar{x}_3) \vee (\bar{x}_1 \& \bar{x}_2 \& x_3) \vee (x_1 \& x_2 \& x_3).$$

This function has three independent variables and so the circuit must contain three pairs of lines—x_1 and $\bar{x}_1$, x_2 and $\bar{x}_2$, x_3 and $\bar{x}_3$. The number of output lines must equal the number of conjunctive terms (in parentheses) of the function being performed. In our case, there are four such terms (Fig. 2.12). All the output lines terminate in diodes whose terminals are, in turn, tied to a single output resistance. Such a circuit performs a disjunction in the same way as any other parallel connection. The input signals are also fed through resistances.

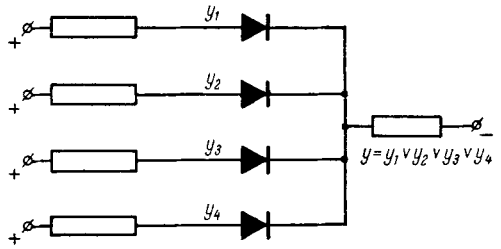

Fig. 2.12.

Each output line represents one of the conjunctive terms of our functional form. But the logical variables, at least in our case, are contained in all such parentheses. For this reason, each output line

must be connected, via diodes, with all those input lines which carry the variables contained in a given conjunctive term. The connecting diodes are arranged so as to permit the current to pass from the output to the input lines. In our example, the first conjunctive term is

$$y_1 = x_1 \& \bar{x}_2 \& \bar{x}_3 .$$

Its corresponding diode switching circuit is shown in Fig. 2.13; the complete diode circuit, performing the complete logical function, is shown in Fig. 2.14.

Any other logical functions may be performed in a similar fashion.

This technique starts from the complete disjunctive normal form of the function and is therefore as canonical a method as that employed for the synthesis of the relay switching circuits of Section 2.3. However, it usually yields circuits that are uneconomical because they require too many diodes. Although there are methods for designing more economical circuits, we shall not dwell on them here and shall refer the reader to the original publications (see, for example, [127]).

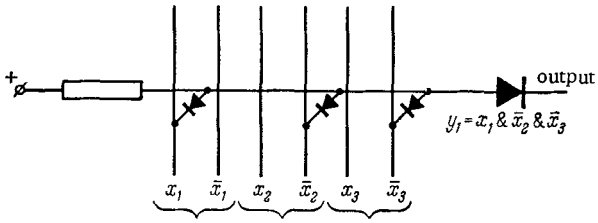

Fig. 2.13.

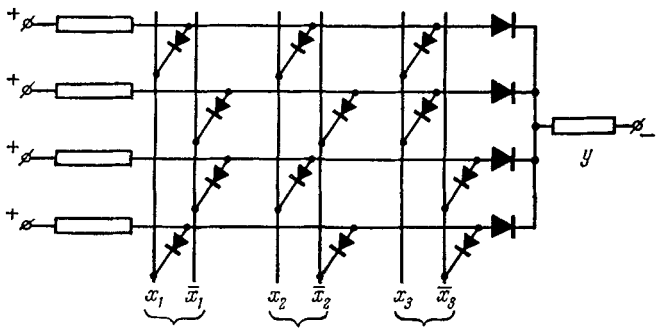

Fig. 2.14.

b) Triode Logic

A triode is an element exhibiting a variable resistance in response to a control signal. All other conditions being equal, the plate current of a triode (vacuum) tube, is determined by the grid voltage; in a transistor triode, the resistance varies as a function of an externally applied signal.

The characteristic curve of any triode device exhibits a saturation, at which the resistance is constant and maximum. We can use as logical variables (levels 0 and 1) the control signal levels which produce the minimum and maximum resistances of the triode. These resistances become the output of the device. Then various combinations of these triodes with constant passive resistances allow us to realize the logical functions of one (Fig. 2.15) and of several (Fig. 2.16) independent variables.

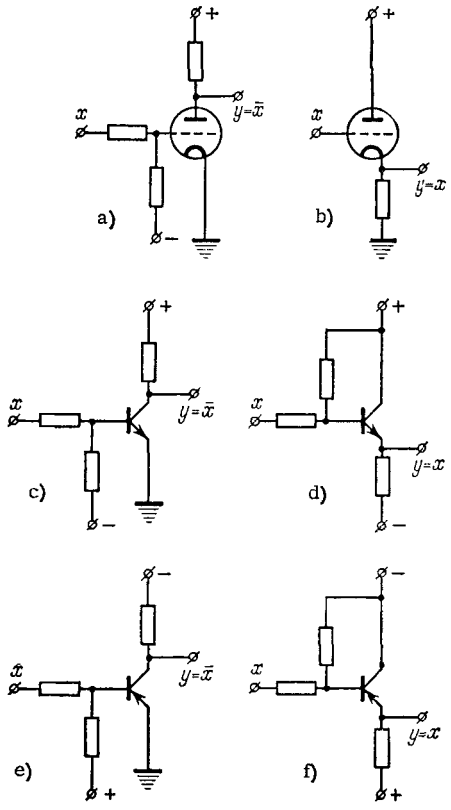

Fig. 2.15.

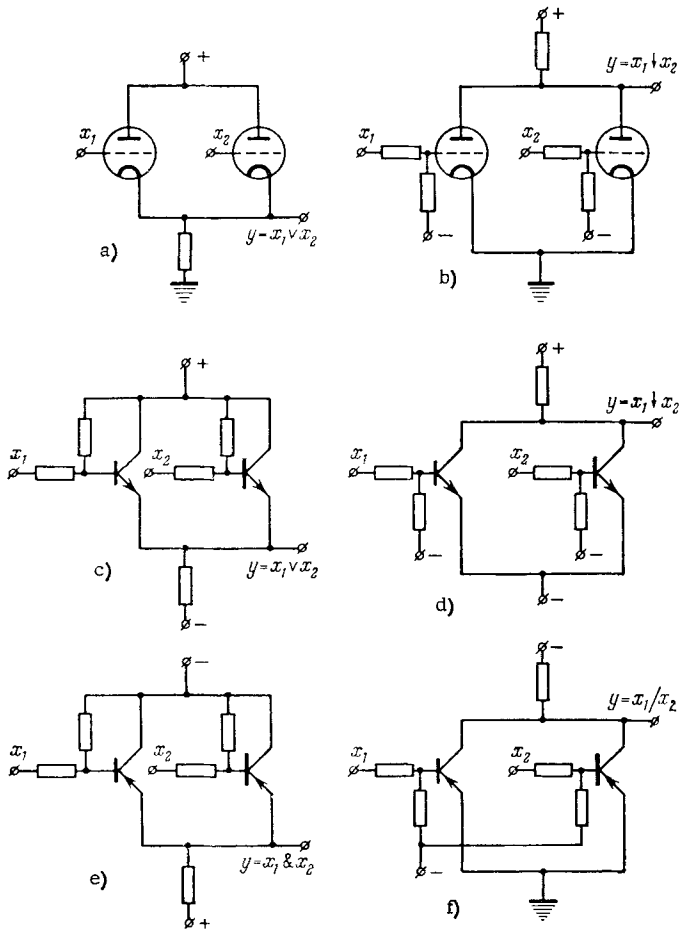

Fig. 2.16.

If one can execute negation, conjunction and disjunction by means of a set of devices, one can also perform any conjunctive term of the complete disjunctive normal form of any logical function, as well as the disjunction of these terms. This being the case, triode circuits can embody any logical function. Again, however, the canonical synthesis proves uneconomical (it yields circuits with redundant elements), so that one usually designs with more advanced techniques.

c) Networks using Magnetic Components

There are many ways of designing logical systems based on magnetic amplifiers, but we shall give only a brief description of a

greatly simplified version of one such system. Figure 2.17 is a schematic of a magnetic amplifier with positive feedback, consisting basically of a magnetic core *1* which is associated with several windings*. The alternating current is supplied to windings $w_{\sim}$ and $w'_{\sim}$, from which it passes, via the diode bridge *2* and the load resistance R_e, to the positive feedback windings w_{fb} and w'_{fb}. In addition, the core carries bias windings w_b and w'_b, as well as one or more control windings: $w_{con\,1}$ and $w'_{con\,1}$, $w_{con\,2}$ and $w'_{con\,2}$, and so on. The bias windings are supplied with a constant direct current i_b. The control windings are also dc-fed, and the levels of this direct current are used as the input variables of the system. The output of the device is the rectified current i_1 in the load circuit.

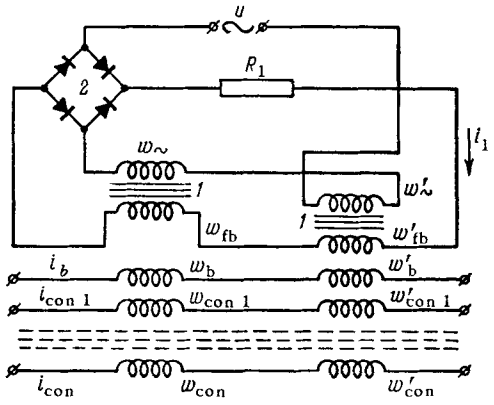

Fig. 2.17.

Consider first an amplifier with only one pair of control windings, $w_{con\,1}$ and $w'_{con\,1}$. Figure 2.18 shows the characteristic of this amplifier, that is, the dependence of the output (current i_1 in the load circuit) on the input (current i_{con} in the control windings) at zero bias current (a bias current shifts this characteristic along the i_{con} axis).

If the value $i_{con} = 0$ is made to correspond to the 0 level of the input variable and any value $i_{con} < -i'_{con}$ to the 1 level, and if the lowest and highest levels of the output current (these being the only possible levels, in accordance with the characteristic of Fig. 2.18) are made the 0 and 1 levels of the output variable, then, at zero bias current, the amplifier will be a negation element.

*The use of a split core and several pairs of windings eliminates ac pickup in the circuits carrying dc currents.

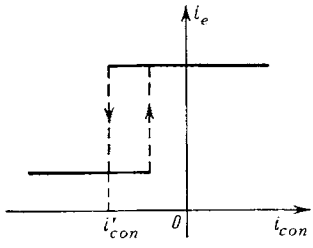

Fig. 2.18.

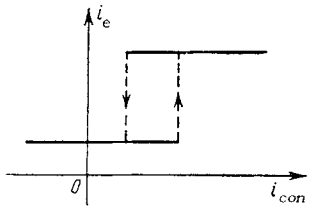

Fig. 2.19.

The same amplifier can perform "repetition." In this case, a bias current i_b is used to shift the characteristic to the right (as shown in Fig. 2.19), and the polarity of the control signal is reversed. Now the lowest level of output dc appears at $i_{con} = 0$; that is, the output logical variable is at level 0. If the input signal is 1 (i.e., i_{con} is high), then the level of the logical variable at the output will also be 1 (the current in the load circuit will be at maximum).

The magnetic amplifier is thus a contactless analog of the electromechanical relay with normally closed or normally open contacts.

We shall now consider a magnetic amplifier with several control windings. The characteristic of Fig. 2.19 will still hold at the appropriate bias current but the abscissa now denotes the total ampere-turns of all the control windings.

Retaining the same 0 and 1 levels of the individual input variables (that is, the same current values) in the corresponding windings as in the case of an amplifier with a single control winding, we now obtain a device performing a disjunction of all the n input variables. Indeed, it is now sufficient to set any one of the control windings at level 1 to obtain the maximum level of the output current.

If, however, the input current corresponding to level 1 in each winding is now reduced by a factor of n (where n is the number of input variables), then the number of ampere-turns necessary to obtain the same output level 1 can be achieved only if *all* the inputs are set equal to 1. The magnetic amplifier then embodies a conjunction of n variables and is the contactless analog of the multiple-coil electromechanical relay.

If the output of one magnetic amplifier is connected to the input of another magnetic amplifier (or to the inputs of several amplifiers), we have a network. In particular, we can use a set of these amplifiers to synthesize any desired combinational switching circuit. And since the individual magnetic components can perform negation, conjunction, and disjunction, a system of containing a multiplicity of such components can embody any desired logical function.

d) Pneumatically Operated Switching Circuits

A schematic diagram of a pneumatic switch is shown in Fig. 20.20,a and Fig. 2.20,b shows the conventional notation for it. The switch housing contains four chambers (K_1, K_2, K_3, and K_4) formed by the diaphragms M_1, M_2, and M_3 carried by a common piston rod R. Set-point controlling pressures P_a and P_b may be maintained in chambers K_1 and K_2 through ducts L_1 and L_2; chamber K_3 is connected to a compressed-air supply line via the axial duct C_3, and chamber K_4 is vented to the atmosphere via duct L_4. Axial duct C_4 from chamber K_4, and duct L_3 from chamber K_3 are interconnected on the outside by means of feedback line FB, in which we establish the output pressure P.

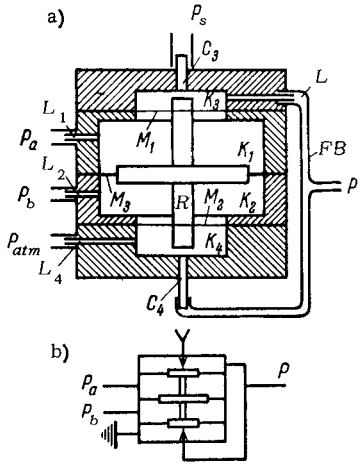

Fig. 2.20.

When the piston rod R is in its extreme "up" position, it blocks duct C_3 and opens up duct C_4; this produces atmospheric pressure in the FB line at the output of the switch. However, when the piston rod is in its extreme "down" position, it blocks C_4 and opens C_3; the output pressure then equals that in the supply line.

The position of the piston rod depends on the direction of the forces acting on its diaphragms, with the magnitude and direction of these forces determined by the pressures in chambers K_1, K_2, and K_3, that is, pressures P_a, P_b, and P. The opposing force exerted by the output pressure P on the diaphragm-rod assembly constitutes a positive feedback.

The response of this pneumatic switch, illustrating the above properties, is presented in Fig. 2.21.

Now consider this switch when a constant pressure $P_b = P_{h1}$ (the back pressure, or bias pressure) is maintained in K_2. The shape of the response remains unchanged from that of Fig. 2.21, but it is displaced to the right, the magnitude of the displacement increasing with back pressure (bias) P_b (Fig. 2.22).

Such a device can be used to perform the logical operation of "repetition." This is done by assigning the level 1 to a pressure higher than P_{h1}, and the level 0 to a pressure lower than $(P_{h1} - \Delta P)$.

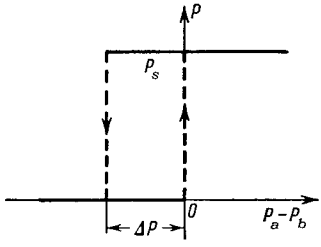

Fig. 2.21.

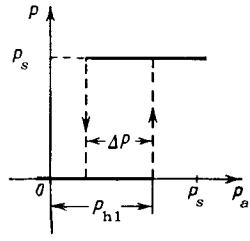

Fig. 2.22.

Obviously, the supply pressure P_s is that exceeding P_{h1}. The switch performing the repetition is shown in Fig. 2.23 in the conventional notation of Fig. 2.20, with the chamber in which pressure P_{h1} (first back pressure) is maintained, indicated by cross-hatching.

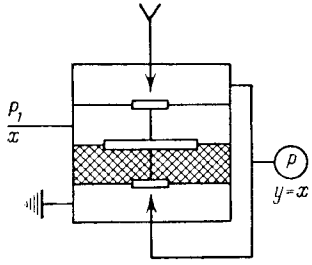

Fig. 2.23.

The pneumatic switch may also be used to perform negation. In this case a constant pressure P_{h2} is maintained in K_1, and the response of the switch is shown in Fig. 2.24. The simplified diagram is shown in Fig. 2.25, with K_1, in which the constant pressure P_{h2} (differing from P_{h1}) is maintained, indicated by hatching. Pressures P_{h1} and P_{h2} differ because P_{h1} determines the location of the right-hand and P_{h2} that of the left-hand vertical line of the hysteresis loop. Our device now performs a negation. Thus we make the independent logical variable $P_1 = 1$ at $P_b > P_{h2} + \Delta P$; the output signal then assumes level 0; if $P_b < P_{h2}$ (that is, when $P_1 = 0$), the output pressure $P = P_s$, and the output signal equals 1.

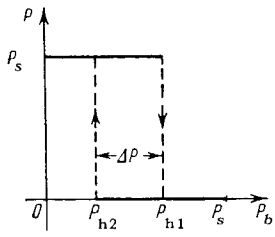

Fig. 2.24.

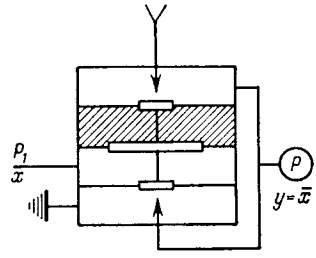

Fig. 2.25.

So far, we have shown how the switch performs logical functions of one independent variable. We shall now show how the same switch can perform logical functions of two or more independent variables.

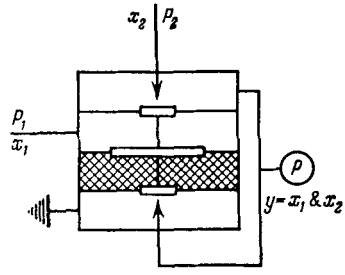

Fig. 2.26.

The schematic diagram of Fig. 2.26 shows that the duct previously leading to the supply line (Fig. 2.23) is now connected to the line producing a second independent input variable P_2. The switch will now perform the conjunction of two independent variables because an above-atmospheric pressure will exist in the output line if, and only if, both input signals are at level 1.

A circuit of $n-1$ devices, assembled as in Fig. 2.27, will perform the conjunction of n independent variables.

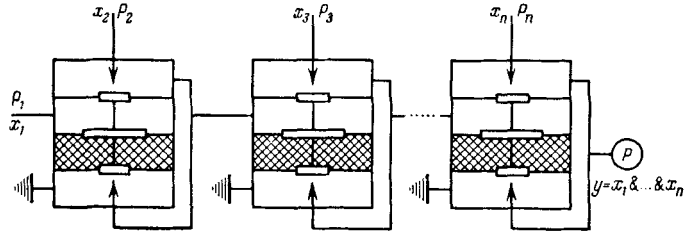

Fig. 2.27.

Figure 2.28 shows a device performing the disjunction of two independent variables while the circuit of Fig. 2.29, which consists of $(n-1)$ pneumatic switches, performs the disjunction of n independent variables.

Since we now have pneumatic devices performing negation, conjunction, and disjunction, we can design pneumatic switching circuits to perform any logical function. Here, too, the canonical method may be used (as we have already stated several times, this method starts with a given function in its complete disjunctive normal form). But, as before, this general procedure yields switching circuits that

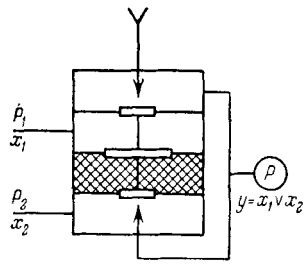

Fig. 2.28.

are uneconomical because they require too many components. We can see this from the mere fact that our pneumatic switch, which may be employed as a device performing negation, repetition, conjunction, and disjunction of two independent variables, can also be used as an implication (Fig. 2.30) or as an inhibit device (Fig. 2.31). The figure shows that implication can be achieved by means of a single switch, whereas the canonical method, which expresses implication by means of negation, conjunction, and disjunction, calls for two such devices. This follows from the formula

$$P = P_1 \to P_2 = \bar{P}_1 \vee P_2.$$

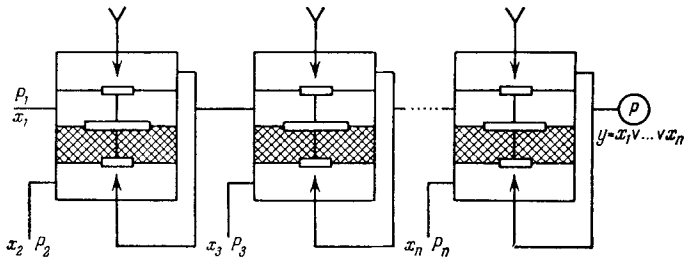

Fig. 2.29.

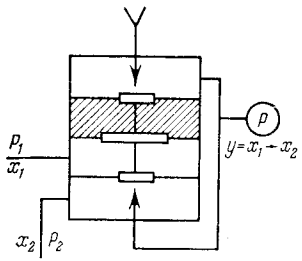

Fig. 2.30.

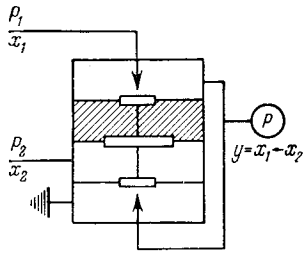

Fig. 2.31.

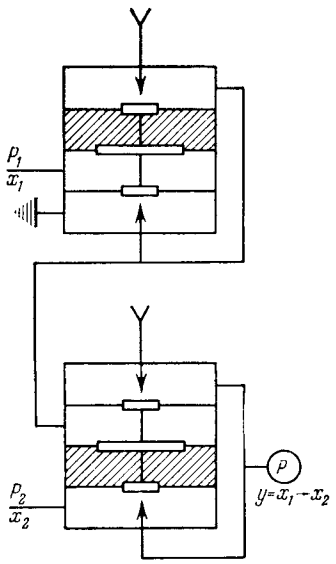

Fig. 2.32.

A switching circuit performing implication as required by the canonical method is shown in Fig. 2.32. This cumbersome arrangement performs the same function as the simple switch of Fig. 2.30.

2.5. THE PROBLEM OF MINIMIZATION OF DEVICES PERFORMING LOGICAL FUNCTIONS

The design of devices performing given functions immediately entails the following problem: Given a set of blocks (elements) capable of performing simple logical functions, each kind of block being associated with some positive number called its "price" (this may be the actual price or some conventional factor), and given also the function to be executed (for example, in its full normal disjunctive form), we want to determine which of the switching circuits capable of performing this function (and consisting of the blocks of the given set) will have the minimum total price, defined as

$$p = \sum_{i=1}^{r} \alpha_i h_i,$$

where α_i is the number of blocks of a particular kind, h_i is the price of one block, and r is the number of different blocks in the set.

This problem, often referred to as the minimization problem, is of fundamental importance in engineering applications of propositional calculus. A great deal of work has been devoted to it, and numerous algorithms* (procedures) suggested for its partial solution. All these procedures consists of more or less complex scanning methods (that is, examination of all the different existing possibilities), so that so far there are no convenient, practical techniques for minimization; all that has been developed is various "paths" along which one may hope to come across more or less economical designs.

To give the reader at least some broad idea of what is involved, we shall briefly describe one of the many procedures for partial solution of the minimization problem.

Suppose the logical function F is given in its complete disjunctive normal form. If the set of blocks consists of the AND, OR, and NOT elements (both AND and OR having two inputs each) and all the elements carry the same price tags, then the minimization problem is reduced to finding that analytical expression of this function which

*The term "algorithm," translated here for convenience as procedure, shall be frequently encountered in subsequent chapters. It shall be defined in Chapter 7.

contains only the symbols $\bar{}$, &, and $\vee$ and in which the number of these symbols is minimum.

Let us describe Quine's solution of this problem [214]. The full form is simplified as much as possible by means of the identity

$$(A \& x) \vee (A \& \bar{x}) = A, \tag{2.1}$$

where A may be a conjunction of several variables. Then the same operation is repeated on all the conjunctions obtained as a result of the first simplification, and so on, until no further reduction of terms is possible. The pairs of conjunctions (originating from the terms of the complete form and from the form obtained as a result of the simplification) which cannot be further reduced by means of the simplifying identity (2.1) are called the *prime implicants of* F. Quine has shown that any minimal disjunctive normal expression of F is a disjunction of certain prime implicants of F. Therefore, the next step in finding the minimal expressions of F consists of determining these combinations of prime implicants whose disjunctions yield minimal expressions. This technique (see [185]) gives combinations of prime implicants whose disjunction is equivalent to F, but from which not a single prime implicant may be eliminated without violating the condition of equivalence to F. Such disjunctions are called "irredundant" expressions for F. Then we count the number of symbols $\bar{}$, &, and $\vee$ in each of the irredundant expressions and select the expressions with the least total of these symbols. These are the minimal expressions, according to our criterion of minimality.

Consider an example. We are given the Boolean function

$$F(x_1, x_2, x_3) = (\bar{x}_1 \& x_2 \& x_3) \vee (x_1 \& \bar{x}_2 \& x_3) \vee \\ \vee (x_1 \& x_2 \& \bar{x}_3) \vee (\bar{x}_1 \& \bar{x}_2 \& x_3) \vee (\bar{x}_1 \& x_2 \& \bar{x}_3).$$

As a result of all possible pairwise reductions of terms of the complete form (whereby each term of the disjunctive form may be used in more than one pair), we obtain the conjunctions

$$\bar{x}_1 \& x_3, \quad \bar{x}_1 \& x_2, \quad \bar{x}_2 \& x_3, \quad x_2 \& \bar{x}_3,$$

which cannot be further reduced. These are also the only conjunctions whose combination does not yield a single further reduction. Thus they all are prime implicants of F. Although the disjunction of all these prime implicants is equivalent to F, it may be proven directly that the deletion of the conjunction $\bar{x}_1 \& x_3$ does not violate the condition for equivalence but that no other remaining conjunction can be deleted without violating that equivalence. Hence,

$$F = (\bar{x}_1 \& x_2) \vee (\bar{x}_2 \& x_3) \vee (x_2 \& \bar{x}_3)$$

is one of the irredundant expressions. It may also be shown that

$$(\bar{x}_1 \& x_3) \vee (\bar{x}_2 \& x_3) \vee (x_2 \& \bar{x}_3)$$

is also an irredundant expression. The function has no other such expressions. A comparison of these two irredundant expressions shows that they have the same number of $^-$, &, and $\vee$ symbols; therefore they are minimal to the same degree.

So much for Quine's procedure. We now have dozens of procedures for finding prime implicants of logical functions. Some of these are more suited for manual calculations, others for computations on computers; still others are mainly employed in research on minimization problems. The methods of minimization also differ: one can use special diagrams [180], various constructs on n-dimensional cubes [33], numerical calculations [161, 127], and so on.

Several procedures (for example, [33]) develop minimal normal expressions by starting with the prime implicants.

The finding of minimal normal expressions of logical functions of even a small number of variables (for instance, six or seven) is a rather laborious process. But we now have several useful simplified procedures which give normal expressions that are close to the minimal and entail much less labor [216-218].

The Quine procedure yields minimal disjunctive normal expressions. However, the minimal conjunctive normal expressions may sometimes prove to be "smaller" than the disjunctive forms. Because of that, one must examine both the disjunctive and the conjunctive normal expressions to select the truly minimal expression. Since the techniques for obtaining minimal conjunctive normal expressions are similar to those for the corresponding disjunctive forms, we shall not dwell on them.

The fact that a function yields a minimal normal expression does not necessarily mean that an even simpler expression cannot be obtained. For example, the minimal disjunctive normal expression of the function

$$F(x_1, \ldots, x_6) = (x_2 \& x_3 \& x_4 \& x_6) \vee (x_1 \& x_2 \& x_5) \vee \vee (x_1 \& x_2 \& x_6) \vee (x_1 \& x_3 \& x_4 \& x_5) \tag{2.2}$$

has thirteen & and $\vee$ symbols, whereas the minimal conjunctive normal expression of the same function

$$F(x_1, \ldots, x_6) = (\bar{x}_1 \& \bar{x}_3) \vee (\bar{x}_1 \& \bar{x}_4) \vee (\bar{x}_2 \& \bar{x}_3) \vee (\bar{x}_2 \& \bar{x}_4) \vee \vee (\bar{x}_1 \& \bar{x}_2) \vee (\bar{x}_1 \& \bar{x}_6) \vee (\bar{x}_2 \& \bar{x}_5) \vee (\bar{x}_5 \& \bar{x}_6)$$

contains $8 + 7 + 16 = 31$ $\bar{}$, &, and $\vee$, symbols; thus, (2.2) is the minimal normal expression. But another expression for the same function

$$F(x_1, \ldots, x_6) = [x_1 \& x_2 \& (x_5 \vee x_6)] \vee [x_3 \& x_4 \& [(x_1 \& x_5) \vee (x_2 \& x_6)]] \quad (2.3)$$

contains only nine & and $\vee$ symbols.

In this case we have reduced the minimal normal expression by means of the identity $(A\&B) \vee (A\&C) = A\&(B \vee C)$. Sometimes, however, one can employ this distributive law to the hilt and still not come up with the real minimum expression. For example, our minimal normal function (2.2) can be reduced still further

$$F(x_1, \ldots, x_6) = [(x_1 \& x_2) \vee (x_3 \& x_4)] \& [(x_1 \& x_5) \vee (x_2 \& x_6)].$$

This form may be obtained from (2.3) by expanding the first term of the disjunction

$$x_1 \& x_2 \& (x_5 \vee x_6) = x_1 \& x_2 \& [(x_1 \& x_5) \vee (x_2 \& x_6)],$$

and employing the distributive law to reduce the new expression.

Obviously, the reduction of other functions requires other identities. It is very difficult, however, to select *a priori* an identity suitable for the reduction of a given expression. In fact, it is even difficult to say *a priori* whether a given expression can be eventually reduced to a more manageable form. Thus a great forward step would be a procedure yielding expressions about which one could confidently say that they are as "small" as can be found, that is, that there are no other forms of a given function which are more "minimal" [120, 121]. Such expressions are called *absolutely minimal*, and their finding involves procedures which are far more complex than those for minimal normal expressions. We shall therefore not discuss them in detail, but shall simply point out that each such nontrivial procedure (algorithm) should have the following two features:

1. It should be able to predict the maximum complexity associated with the absolutely minimal expressions of a given function.
2. It should be able to give the absolutely minimal expressions within the limits imposed by the predicted maximum complexity.

For example, one can predict that the absolutely minimal expressions for the function (2.2) are not more complex than a "disjunction of conjunctions of disjunctions" (type I) and a "conjunction of disjunctions of conjunctions" (type II). One then uses special algorithms to express (2.2) in terms of these limiting forms I and II. This gives two expressions of type I

$$F(x_1, \ldots, x_6) = [x_3 \& (\bar{x}_1 \vee \bar{x}_2)] \vee (x_2 \& \bar{x}_3),$$
$$F(x_1, \ldots, x_6) = [x_2 \& (\bar{x}_1 \vee \bar{x}_3)] \vee (\bar{x}_2 \& x_3)$$

and one "degenerate" expression of type II

$$F(x_1, \ldots, x_6) = (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \& (x_2 \vee x_3).$$

Notice that both expressions resemble irredundant forms. With only three forms to scan, it is easy to see which is the smallest.

In general, however, the number of expressions similar to the irredundant forms is extremely large, even if the function has only a few variables, and the above procedures for absolutely minimal expressions are therefore not practical. For this reason the problem was attacked by developing procedures involving considerably fewer elementary operations. Such procedures necessarily yield expressions of a more complex form than the normal, but such expressions in general tend to approach the absolutely minimal. For example, the procedure may involve successive applications of the distributive law to the prime implicants of a given function. The resulting complex implicants may then themselves be treated as prime implicants and serve as a basis for developing irredundant expressions. The final minimal irredundant expressions can then be selected from these irredundant forms in the usual way.

However, there is another problem. Even if the absolutely minimal expression is known, the circuit based on it may prove to be nonminimal. For example, the absolutely minimal expression of the function

$$F = x_3 \& [x_5 \& (x_1 \vee x_2) \vee (x_2 \& x_4)] \vee \vee [x_6 \& [(x_1 \& x_5) \vee (x_2 \& x_4)]], \tag{2.4}$$

immediately yields a switching circuit of ten elements. However, a circuit performing this same function can also be constructed from eight elements (Fig. 2.33). This is due to the fact that, in some cases, one section or block of a system can be used to embody more than one part of the minimal expression. Thus we can represent

(2.4) in the form

$$F = \{x_3 \& [(x_5 \& x_1) \vee (x_5 \& x_2) \vee (x_2 \& x_4)]\} \vee [x_6 \& [(x_1 \& x_5) \vee (x_2 \& x_4)]] = \{x_3 \& [Q \vee (x_5 \& x_2)]\} \vee (x_6 \& Q)$$

where

$$Q = (x_1 \& x_5) \vee (x_2 \& x_4), \tag{2.5}$$

Our actual circuit of Fig. 2.33 can then be reduced to eight elements because the Q operation, which appears twice in the irredundant expression, can be iterated through one and the same circuit block.

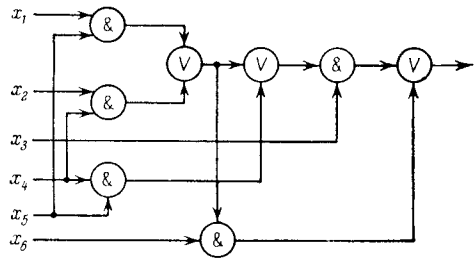

Fig. 2.33.

We have briefly reviewed the minimization problem assuming that all the elements carry the same price tag. It has been shown [217] that the minimization of circuits comprising elements of differing prices can be achieved by modifications of the same methods. The only difference is that a different criterion of the minimum is used in selecting the minimal expressions from the irredundant forms.

Our discussion was confined to minimization of sets consisting only of NOT, AND, and OR blocks, where the AND and OR units had only two inputs each. There are also solutions for similar problems involving other sets. However, each new set requires a new solution of the minimization problem. Thus, if the set consists of blocks of negation as well as of conjunction and disjunction of n variables, the problem reduces to finding irredundant expressions (or expressions similar to irredundant forms if we deal with compound expressions) in which the number of prime implicants is minimum.

The minimization problem has become especially important due to the advent of general-purpose elements, that is, blocks that, either by means of simpler readjustment or by adding external connections

which cost little or nothing, may be used to perform several different functions. A typical example of such a block is the pneumatic switch of Fig. 2.20. There are no accepted solutions of the minimization problem for these systems, despite many attempts at developing one. The present trend in these systems is to develop procedures which would yield circuits that, while not minimal, are sufficiently minimized for practical purposes.