Chapter VIII

AUTOMATIC COMPUTERS—ANALOGUE MACHINES

THE concept of information described in earlier chapters was developed by communications engineers who were concerned merely with the transmission of messages from one place to another. Whether they dealt with wire or wireless communications or for that matter with any other information-handling system like tape recording, public address, or television, the principal objective of the various gadgets devised was to reproduce the input signal with as high fidelity as possible at the output device. But, as we remarked in Chapter I, from mere hi-fi transmission of information by coding to its "processing" is but a step. The simplest machines that process the information they receive instead of merely transmitting it in its original purity are the automatic computers. Since the living brain is also a neurophysiological machine (though of infinitely greater complexity) that processes the information transmitted within it by the different sensory modalities such as vision, hearing, or touch, it is but natural that the study of computers should provide at least a foothold for a first glimpse into the nature of human intelligence. As we shall see more clearly in Chapter X, the principles underlying the organization of computers have suggested various types of mechanisms that are at work in the functioning of the animal nervous system even if they as yet elucidate its mystery in a relatively minor way. Thus the servomechanism principles made familiar by the inventions of analogue computers described in this chapter have been found to be at work in the reflex arcs of the animal nervous system, in the prevention of runaway excitation phenomena such as occur in epileptic seizures, or in the regulation of sensitivity to select aspects of the sensory input data. Likewise, digital computers, the subject of the next chapter, have suggested possible choice mechanisms that may be employed by

the living brain in perception and selective attention, or what the psychologists call "set." This is why we take up now the theory of automatic computers, the first artificial automata that modern technology has enabled us to build, as a prelude to that of natural automata.

Automata or automatic computers, as the name implies, are concerned with computation. Even those automata which are used to control processes, such as directing anti-aircraft fire or running an oil refinery, are really computers in that the control function in question is fundamentally the outcome of a computation. Thus the power of the anti-aircraft gun predictor to direct the gun's aim stems from its ability to compute the future rendezvous of the aircraft and the shell it fires by solving a pair of simultaneous equations. In all such cases, whether of pure computation or control via computation, the input is the data of the numerical problem to be solved, and output is the solution arrived at after manipulation of the information fed into it. There are just two ways of feeding a computer with input information. First, it may obtain such information by means of a device which translates numbers into physical quantities measured on specified *continuous* scales such as lengths, angular rotations, voltages, currents, magnetic states, forces, torques, deformations, flow, and so on. After operating with these quantities it measures some physical magnitude which gives the result. Secondly, it may employ a contrivance which operates with numbers directly in their digital form by counting *discrete* events like electrical pulses or discrete objects like the teeth of a gear wheel, as is the case, for example, with the ordinary desk calculating machines commercially known as Brunsviga and Marchand. We shall deal with digital computers in the next chapter and confine ourselves here to computers of the former type.

An example of a computer that operates with continuous magnitudes may be seen in the engineer's slide rule. Here, to multiply two numbers x and y, we read on the first scale the length corresponding to the logarithm of the number x and, by sliding to the required extent a second scale alongside the primary one, add to it the length corresponding to the logarithm of the other number y. We then read the number corresponding to the combined lengths to obtain the product xy, making use of the basic law of logarithms according to which the logarithm of any *product* (xy) is the *sum* of the logarithms of the individual multiplicands x and y, that is,

$$\log (xy) = \log x + \log y.$$

In general, continuously variable physical quantities such as distance between marks on a foot rule, angles through which a disk is rotated, electric currents in wire networks, or the quantity of light transmitted through optical media are used to represent numbers, and various mathematical operations are performed on them by means of specially contrived components exploiting some known physical law. Thus Ohm's law, $c = \frac{v}{r}$, connecting the current (c) flowing through a wire, its resistance (r), and voltage (v) across its terminals may be used to derive the quotient of two numbers v and r by simply measuring the current flow c through it or alternatively the product cr by measuring the voltage v. Appliances like slide rules and electrical networks, which perform mathematical operations such as multiplication and

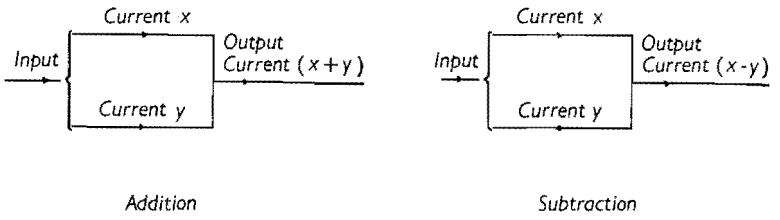

Fig. 15

division by reproducing analogous physical situations implementing the operation in question, are known as analogue computers.

Now any numerical computation, no matter how complex, is basically a series of additions, subtractions, multiplications, and divisions performed on numbers according to a predetermined plan. Consequently, if we devise on the analogy principle organs or components capable of performing on the representative numbers measured by the physical quantities these four basic operations of arithmetic, we can in theory solve any numerical problem by a sufficiently ingenious arrangement of such components and their proper orchestration. Since it is not difficult to realize these four arithmetical operations by mechanical, electrical, optical, and other means, it is possible in theory to make analogue machines capable of performing any kind of computation, even though, as we shall see later, we may not in every case be able to implement the theory in practice. The construction of analogue computers is therefore the

art of assembling and orchestrating components that carry out the four fundamental operations of arithmetic already mentioned.

Components that execute addition and its inverse subtraction are quite easy to make. Thus we may add or subtract two numbers x and y by representing them as currents in wires and then merging them in parallel or antiparallel directions (see Fig. 15). Multiplication (xy) of two currents as well as division is more difficult; but various kinds of electrical circuitry yielding both can be devised.* If, on the other hand, we choose to represent numbers by the angles through which certain disks are rotated, instead of addition ($x + y$) and sub-

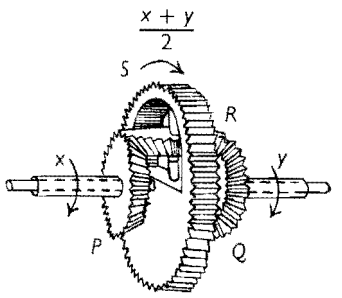

Fig. 16 Differential gear (adding mechanism).

traction ($x - y$), the operations $\frac{x+y}{2}$ and $\frac{x-y}{2}$ are offered because they emerge prefabricated out of a readily available mechanism, the so-called differential gear universally used on the back axle of an automobile. A glance at Fig. 16 showing the arrangements of its main parts indicates how it can yield the semi-sum or difference of any two numbers. It will be seen that two input shafts P and Q both mesh with a third gear R which can turn freely about an axis fastened to the inside rim of a fourth larger gear S. As a result, when we turn the shaft P by x and shaft Q by y the gear S turns by $\frac{x+y}{2}$. The same

* Thus currents may be multiplied by feeding them into the two magnets of a dynamometer thereby producing a rotation. The rotation is then transformed back into current which is the product of two input currents. The transformation mechanism is rather complex. It employs a rheostat to turn the rotation into resistance which is then connected to two sources of fixed but different electrical potentials.

device generates the semi-difference $\frac{x-y}{2}$, if only we turn the input shaft Q in the opposite direction. Likewise, instead of multiplication xy and division $\frac{x}{y}$ a mechanical analogue computer uses an entirely different operation called integration. The reason is that the product xy of two numbers x and y can be expressed as the sum of two integrals, and a very simple device—a wheel turning on a surface—enables us to accomplish integration. To understand how such a mechanism performs this recondite mathematical operation, consider first a wheel provided with a counter which counts the number of turns it makes about its axis. If we let such a wheel roll without slipping over a plane

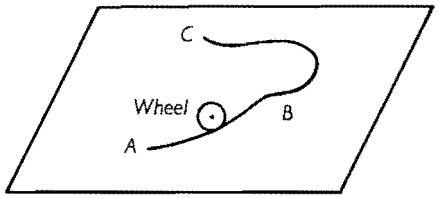

Fig. 17

surface like the floor of a room, along any curve such as ABC in Fig. 17, the number of turns the wheel makes obviously will indicate the length of the curved path it traverses. This is precisely how an automobile milometer works. By merely counting the number of turns the automobile wheel makes (which it converts into miles by multiplying the number of turns by a conversion factor) it measures the length of the curve the wheel traces as the automobile meanders along a winding roadway. A rolling wheel with a counter thus provides us with a physical analogue of the simplest form of *integral*, the length of the curve ABC, which incidentally is the sum of the lengths of a very large number (n) of very small quasi-straight bits of lengths ds_1, ds_2, ds_3, . . ., ds_n into which the curve ABC may be sliced. Thus the number of turns of the wheel equals the sum

$$ds_1 + ds_2 + ds_3 + \ldots + ds_n.$$

The smaller the elementary bits into which we fragment the curve, the straighter and more numerous they become. We may denote the quasi-infinite sum of such elementary bits by the symbol S and write

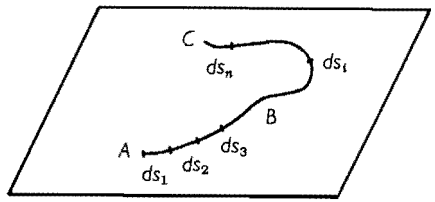

Fig. 18

it as Sds_i as an abbreviation of the phrase "sum of bit lengths typified by ds_i where i runs successively through all the integers 1 to n."

When the number of bits is increased indefinitely, with each bit length becoming vanishingly small, the limiting infinite sum is called the integral and is abbreviated as $\int ds$, where the integral sign $\int$ is merely a distortion of the summation symbol S, and ds is the length of any typical infinitesimal straight-line slice, all of which in the aggregate span the full curve ABC.

If further we make the wheel turn on a plane platform that itself rotates, we obtain what is called an *integrator*. For the new mechanism can be made to compute any kind of integral whatever. Thus consider a wheel W turning vertically around a long horizontal axle T that stretches across a rotating platform P from side to side but crossing exactly its center of rotation (see Fig. 19). The platform P rotates horizontally about a vertical shaft S. By merely turning a screw Q going through the support of the platform P it is possible to shift the center of the platform in relation to the edge of the wheel. If we turn the platform a little, the wheel pressing on it will also

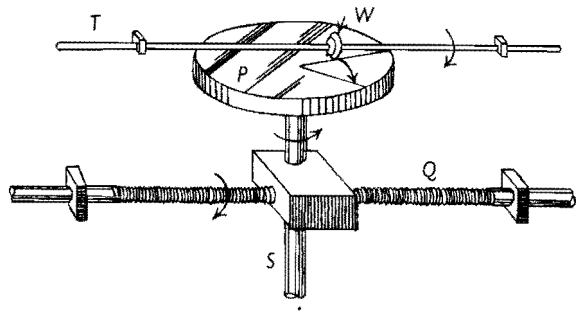

Fig. 19 Mechanical integrator.

correspondingly turn. But as the turning in the wheel induced by the turning of the platform is directly proportional to the distance of the resting wheel edge from the center of the platform, it follows that a small turning dx of the platform superimposed on a turning y of the screw causes the wheel to turn $y\,dx$. Consequently the total turning of the wheel when the platform undergoes a succession of several infinitesimal turnings $dx_1, dx_2, dx_3, \ldots$, corresponding respectively to the screw Q turning synchronously $y_1, y_2, y_3, \ldots$, is the sum

$$y_1\,dx_1 + y_2\,dx_2 + y_3\,dx_3 + \ldots.$$

This sum is precisely the integral $\int y\,dx$. A mechanical integrator is therefore merely a wheel turning on a rotating platform whose center can be shifted by turning another screw so that with an input

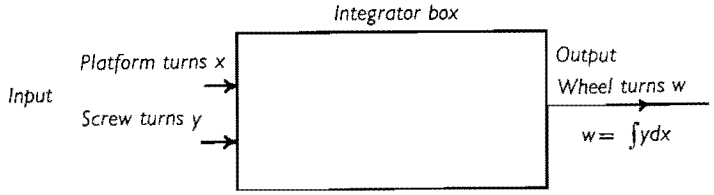

Fig. 20

of x turns of the platform and the concomitant y turns of the screw we have an output of w turns of the wheel measuring the integral $\int y\,dx$. (See Fig. 20.) The wheel-and-platform integrator we have described is a purely mechanical device. But we can make its electrical counterpart easily enough. All we need do is to replace the input variable x of the mechanical integrator by the universal variable, time, which continues to flow at all times and everywhere independently of us. In other words, an electrical integrator can be constructed provided we wish to derive the time integral of any given function. To understand how we may obtain such an integral, consider an electrical capacitor which stores up electrical charges. As current flows into it, the voltage or potential difference across the capacitor changes. But the change in voltage (dv_1) during any short period of time (dt_1) is proportional to the current y_1 and to the length of time it flows, so that

$$dv_1 = y_1\,dt_1.$$

It therefore follows that after the lapse of several such successive time intervals, say, $dt_1, dt_2, dt_3, dt_4, \ldots, dt_n$, the voltage difference will be the sum

$$y_1\,dt_1 + y_2\,dt_2 + y_3\,dt_3 + \ldots + y_n\,dt_n.$$

As before, if the intervals of time $dt_1, dt_2, dt_3, \ldots$, considered are infinitesimally brief, the sum is the integral $\int y\,dt$.

Now it can be proved that the product xy of any two numbers x and y can be expressed as the sum of two integrals, though the proof will not be reproduced here:

$$xy = \int x\,dy + \int y\,dx.$$

Consequently, to obtain multiplication we require two integrators and one adding device. Integrators can be built to yield division $\left(\frac{x}{y}\right)$ too but we have to resort to a rather different artifice for the purpose.

Since, as we remarked before, any numerical problem, however complex, can be resolved into a series of basic arithmetic operations of addition, subtraction, multiplication, and division performed in a predetermined order, analogue computers to solve such problems can be built by proper sequencing in tandem of components like differential gears, wheel-and-platform integrators, electrical networks, and other simulators which perform these basic operations. To make such a chain of components actually work, however, we have to employ yet other kinds of catalytic devices called amplifiers and servos. If the components are electrical, such as circuits and other electromechanical appliances, we need amplifiers to make up the huge losses of input power used to drive them. If, for instance, a computer component transmits, say, one-tenth of its incoming power to its next neighbor in the chain, by the time the sixth component is energized its power punch is attenuated by a factor of $(\frac{1}{10})^6$, or a millionfold. Clearly, with such prodigious enfeeblement en route, no amount of initial power input will suffice to activate the final components in the chain unless it is replenished on the way by means of amplifiers. The amplifiers used in the computers are in principle similar to their counterparts in communications engineering, though they have a more sophisticated design. In mechanical computers built out of components like wheels and gears, since the amplifiers' electrical output cannot be directly used, we have to employ its mechanical counterpart to

secure the required uplift. Such a device is called a "servo," meaning a mechanical slave. It is a power-boosting invention initially called for by the need to manipulate increasingly unwieldly rudders of ships as ships began to grow in size. Marine designers therefore built steam steering engines to turn the rudders, arranging their control valves admitting steam to the appropriate engine cylinder in such a way as to keep the ship automatically on a fixed course, come wave, wind, gust, or gale. There is nothing mysterious in this type of self-steering mechanism. For it functions in exactly the same manner as the erstwhile helmsman that it displaced. As we know, the helmsman at the ship's wheel estimates by means of a compass the deviation between the ship's actual and desired course and uses this information to turn the wheel to the precise extent required to correct the deviation. The ship's self-steering servomechanism likewise uses signals

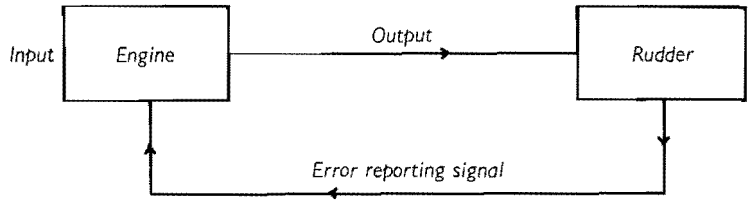

Fig. 21 Servo steer.

reporting the deviation or angular difference between the actual and prescribed course to modify the output of the engine activating the rudder exactly to the extent required to rectify the reported error (see Fig. 21). In other words, the output of the steering engine is made to influence its own input so as to steer the ship automatically on its prescribed course. This feature whereby the output of a system is used to control the source of its driving power in such a way that power is throttled, on the one hand, if its output rises beyond a determined point but is amplified, on the other, if the output lags, so that the system regulates itself automatically at the prescribed level, is the heart of all control systems simulating purposeful or teleological behavior. It is called "feedback" by communications engineers; "servo system," "closed-loop" or "closed-cycle control system" by systems engineers; "homeostasis" by physiologists; "reflex neural circuit" by neurologists; "*petitio principii*" by logicians; "vicious circle" by psychologists, and "boom and slump cycle" by econo-

mists. It appears even as the first act of creation in the following metaphysical limerick:

> Said a fisherman at Nice,
> "The way we began was like thees
> A long way indeed back
> In chaos rode Feedback
> And Adam and Eve had a piece."

Its underlying rationale is that physiological processes such as those that help keep constant our body temperature, blood pressure, or glucose concentration in the blood stream are as good examples of

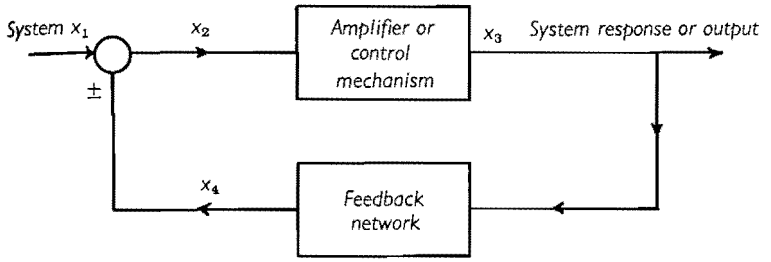

Fig. 22 Feedback mechanism.

feedback principle in action as a thermostat controlling a furnace to steady the room temperature, a computer guiding a missile to its zigzagging quarry in the sky, or a servo steer navigating a ship on its prescribed course. In all these cases feedback adapts the behavior of the system to its desired goal by using incoming error-reporting signals to correct the error caused by outside random disturbances.

Although a powerful tool in the design of control systems, the feedback principle is not without a serious pitfall in that systems based thereon are often liable to uncontrolled oscillations or hunting. Consider the case of ship's servo steer itself. Because of the inevitable time lags on the one hand, between the actual state of the rudder and arrival of the input signal reporting this state to the steering engine, and, on the other, between the arrival of the signal and the completion of the rectifying action it initiates, it is quite likely that by the time the ship's rudder reaches the desired alignment it has acquired sufficient momentum to overshoot the mark. The error then is reversed and the steering system applies the correction, but because

of the time lags it may overshoot again in the opposite direction. A system liable to overshoots of this kind behaves in one of two ways. Either the amplitude of each overgo progressively diminishes and tends to die out, thereby settling the system into a stable state, or it goes on increasing, throwing the system off balance in a frenzy of violent yawing. It is the latter eventuality that often comes to pass, unless the system is cured of it by minimizing the time lags as well as dissipating the system's surplus momentum by means of some kind of braking or damping device.

In general, there has to be quite a precise tie-up between the output response of a servo system and its error-reporting input signal, as will be seen from Fig. 22 representing the essential elements of all such systems employing feedback. The servo's own input x_1 is modified into x_2 by the receipt of error-reporting feedback signal x_4 by addition or subtraction as shown in the equation

$$x_2 = x_1 + x_4$$

The resultant signal x_2 is the actuator or amplifier of the system yielding the output or response x_3 which in turn determines x_4, thus completing the feedback circuit. If for any reason the error-reporting incoming signal x_4 fails to modify its own input x_1 to the precise extent required to ensure its stability, it may spark an orgy of uncontrolled oscillations that makes the system utterly useless for any self-regulation. The phenomenon "intention tremor" wherein the patient's hand merely ends in violent to-and-fro swings in a vain attempt to pick up a cigarette is a case in point of such a failure of feedback due to an injury in the cerebellum. To avoid similar oscillatory swings in electrical and mechanical feedback systems we need a very refined sophistication of the principle, one we cannot go into here. Suffice it to remark that H. Nyquist first developed it to evolve the design of electrical feedback amplifiers. The advanced state of electrical communications theory permitted rapid advances in the art. As is often the case, these advances were applied back to the mechanical servo systems so that we now have a vast armory of devices. For example, servos may accept weak mechanical signals in the form of shaft rotations and retransmit them as powerful mechanical motions. Alternatively, they may take in weak electrical signals and transform them into mechanical motions, thus performing a dual amplifying-*cum*-conversion function changing electrical data into mechanical. Servos can also be made in sizes and powers ranging all the way from

the Lilliputian transistors operating light dials to Brobdingnagian transformers lashing giant guns into a crescendo of fury but with a deftness as great as that of Arturo Toscanini's baton. This is why feedback principle is a new milestone in the theory of servo systems and automation. It has already led to automatic control processes that not only keep ships on their courses and guide missiles to their quarries in the sky but also enable us to run chemical plants, oil refineries, or steel, textile, and paper mills, as well as to process office data concerning sales, inventory, production, and payroll—all without human intervention.

As we mentioned before, all these control processes are the outcome of elaborate computations. Most of the computations handled by analogue machines in common use fall into three main categories. One group solves finite equations, both algebraic and transcendental, and in typical instances makes use of cams, linkages, gears, or variable electrical elements to establish the required connection among the given variables. A second group evaluates integrals and derivatives by means such as wheels rolling on surfaces of various shapes, or charges and currents in electrical circuits, or the quantity of light transmitted through specially prepared optical media. The third group solves partial differential equations and may employ elastic membrances, electric currents in conducting sheets of polarized light.

Despite the great diversity of problems that analogue computers enable us to solve, there are three serious limitations in the way of their further development. First, the accuracy of an analogue computer is limited by the accuracy with which physical quantities can be measured. In practice it can rarely reach a precision of about one part in 10,000 and has often to make do with much less—sometimes 50 to 100 times less precise. While this level of exactitude is adequate in some cases there are many others where it is not. Secondly, there is a definite physical limit to the extent to which the various components in an analogue computer chain can be cascaded. If it is a mechanical computer, longer and ever longer links of gears, shafts, cams, wheel-and-platform integrators, and other gadgets must be strung together. If, on the other hand, it is an electrical computer, more and more amplifiers must be provided to rejuvenate the continually fading actuating power. In the former case, the inevitable looseness in the gears and linkages, though acceptable in simple setups, will eventually cumulate to the breakdown point where the total "play" in the machine will exceed the significant output

quantities, making it utterly useless. In the case of electrical computers, the random electrical disturbances called "noise" inevitable in electrical circuits, will similarly snowball to such an extent as to drown the desired signals although the breakdown stage in electrical structures is reached much later than in mechanical systems. As a result, electrical analogue computers can be built to far more complicated designs than mechanical machines. Nevertheless, each type has its own limiting complexity beyond which it will not work. For example, the Radio Corporation of America's analogue computer, Typhoon, which simulates the flight of a guided missile, closely approaches this limit, being perhaps the most complicated analogue device existing. Thirdly, an analogue computer's dependence on specific physical situations to simulate complicated mathematical functions, as with the wheel-and-platform integrator, has the consequence that when one has been found to reproduce a mathematical relation it has little or no application beyond that relation. As a result, analogue computing devices are simpler but inevitably less versatile. For more versatile computers that can take in their stride any mathematical problem and can handle equally well two vastly different situations such as the flow of heat in a cylinder and the determination of an optimum inventory pattern for a warehouse, we have to make machines that do not have to search for physical simulations to represent the mathematical operations they perform. Such machines are the digital computers which operate with numbers directly in their digital form by counting *discrete* objects or events. In essence, they attempt to reduce control and computation to a still more basic elementary level. As we saw, analogue computers reduce control functions to computation and the latter to arithmetic. Digital computers carry this reduction a stage further by resolving arithmetic, in turn, to mere counting. They thus provide a new rationale for the *double-entendre* of the verb *compter* in Anatole France's aphorism: "People who won't count do not count." This extension of the normal meaning of "count" is not peculiar to French and English. It is common to most other languages in implicit recognition by the human race of the pre-eminence of *counting* in its affairs.

Chapter IX

AUTOMATIC COMPUTERS—DIGITAL MACHINES

AS we remarked in the last chapter, a digital computer operates with numbers directly in their digital form by counting discrete events or objects such as electrical pulses or gear teeth instead of *measuring* continuous magnitudes like distances, currents, voltages, angles, rotations, and the like, as in analogue machines. In doing so it scores several advantages over its analogue rival, among which two are pre-eminent. In the first place, not having to rely on specific physical situations to simulate mathematical operations it is certainly much more versatile even though the simplest digital machine is likely to be a good deal more elaborate than its analogue counterpart. In the second place, while the accuracy of the latter inevitably depends on that of the construction of the continuous scale on which the physical quantity is measured, that of the former relies only on the sharpness with which a discrete set of events like electrical pulses or wheel teeth can be distinguished from one another. Since it is easier to distinguish a set of discrete events than to construct a fine continuous scale, the former, that is, the digital machine, is preferable for high-precision work. Further, since it is simpler still to distinguish two discrete events than ten, digital machines constructed on the binary scale are superior to those on the decimal scale of our daily use. A digital machine on the binary principle is therefore merely an assemblage of components such as switches or relays each one of which is capable of only two states instead of the myriads that the analogue type can assume. One advantage of the simplification is that the component can be made to compute as well as decide, as will be apparent later on. This ability to decide is a particularly valuable asset in an automatic computer. For it is obviously a machine which is designed to turn out the final answer with as little human interference as possible after the initial

input of data. This means that after the initial insertion of the numerical data the machine must be able not only to perform the computation but also to decide among the various contingencies that may arise during the course of the calculation in the light of blanket instructions inserted at the beginning with the numerical data. In other words, a digital computer is also a logical machine capable of making a choice between "yes" and "no," the choice of adopting one or the other of two alternative courses open to it at each contingency that may arise during the course of the computation. Since both its number vocabulary and its range of choices at each stage are binary, its structure need be only a bank of relays or switches each capable of two conditions—say, "on" and "off," one of which may denote the number "0" or the choice "no" and the other the number "1" or the choice "yes." All that we have to do to make the bank of relays function in the desired manner is to ensure that at each stage the relays are able to assume positions determined by the position of some or all relays of the bank at a previous stage by incorporating a clocking device* for progressing the various stages.

The arrangement envisaged is possible because the fundamental rules of any arithmetical calculation, namely, addition and multiplication, are formally identical with those of the logical calculus, as we shall presently show. It is because of this identity that the apparatus designed to mechanize calculation is also able to mechanize acts of logical choices, that is, decisions. To show the identity of the logical and arithmetic calculi, consider first arithmetic. As we explained in Chapter II, the binary system of representing numbers by using only two digits, 0 and 1, is fully as effective as the decimal system of our daily use even though it is about 3.32 times as lavish in the number of digits used to express any given number. If we write numbers in the binary notation, we can add or multiply any two numbers by repeated applications of the following basic rules:

I. *Addition Rules:* $0 + 0 = 0$; $0 + 1 = 1$; $1 + 0 = 1$; $1 + 1 = 10$. If we remember that while adding one to one, we get a "one" which should be "carried" to the next place, we can summarize the addition rules in Table 19 of arithmetical addition.

To read the result of the addition of any two numbers, say, 0 and 1, take the row 0 and column 1; these are easily seen to intersect at 1.

* There are asynchronous computers requiring no clocking arrangements but we shall not deal with them here.

The same rule applies in reading all the subsequent tables described in this chapter.

TABLE 19

Arithmetical Addition		
+	0	1
0	0	1
1	1	10

Consider now the rules of ordinary multiplication. They are as follows:

II. *Multiplication Rules:* $0 \times 0 = 0$; $0 \times 1 = 0$; $1 \times 0 = 0$; $1 \times 1 = 1$. Obviously they too can be summarized in a similar table of arithmetical multiplication:

TABLE 20

Arithmetical Multiplication		
×	0	1
0	0	0
1	0	1

There are no doubt specific rules of arithmetical subtraction and division. But both can be reduced to addition and multiplication. Thus subtraction may be turned into addition by what is called complementation. Consider, for instance, any ordinary subtraction like $542 - 123 = 419$. We may substitute for the subtrahend 123 its nines complement, that is, the number 876 obtained by subtracting from nine successively the three digits 1, 2, 3. The original *difference* $(542 - 123)$ may then be obtained from the *sum* of the minuend and the complement $(542 + 876)$. The latter sum is 1418. We notice that this sum is the same as the actual difference 419 except that the last digit 1 on the extreme left needs to be shifted to the units place. That is, $\overset{\curvearrowright}{1418}$ becomes 419 by adding 1 to 8 as indicated by the arrow. In the digital computer operating with only two digits 0 and 1 the summit in our new scale of reckoning is 1 instead of nine so that we have to employ complements of 1 instead. Consequently, to subtract any binary-digit number from another, we *add* to the latter

the complement of one in respect of the former with the endaround "carry" as before. Thus the difference (1011 − 100) = 0111 is the *sum* (1011 + 011) = (1110), which, with the end digit 1 carried around as indicated, becomes 0111, the desired difference. Note that one's complement of 100 is the number 011, obtained by changing 1 into 0 and 0 into 1 in the original number.

Division is iterated subtraction just as multiplication is iterated addition. By trial-and-error subtraction procedures we may actually execute any given division. It therefore follows that all the four basic arithmetic operations may be reduced to only two—addition and multiplication—or even only one, with multiplication becoming merely repeated addition. In consequence, all arithmetical operations may be reduced to patterns of alternative actions organized in highly repetitive sequences and governed by strict rules embodied in the afore-mentioned tables of addition and multiplication.

To see the close affinity between these rules of arithmetic and those of the logical calculus we may remark that in logic we deal with meaningful statements or propositions, not numbers. Nevertheless, since all such statements are either true or false, they too can be assigned truth values which can be handled in a manner very similar to that of the manipulation of numbers. Thus let us assign the truth value $T = 1$ when a proposition is true, and $T = 0$ when it is false. Every proposition such as "Socrates drank the hemlock"—which we may denote by the letter A for facility of subsequent reference—will then have the truth value 0 or 1. Obviously every proposition like A yields another, its contrary or not-A, usually symbolized as $\bar{A}$ or $\sim A$. By definition $\bar{A}$ merely affirms the denial of A so that if the truth value of A is 1, that of its denial, not-A or $\bar{A}$, is 0, and vice versa.

If we have another proposition, say, "Voltaire wrote *Gulliver's Travels*" (which we may denote by B), we can form a compound proposition from A and B in various ways. First, we may produce a compound proposition S which is considered true provided *either A or B or* both are true. In this case S* is the logical *sum* of A and B and the process of obtaining it is the analogue of numerical addition. Second, we may obtain another compound proposition P which is considered true if, and only if, *both A and B* are true. P† is known as

* S is also known as the disjunction of A and B and is written as $S = A \vee B$ or $A + B$.

† P is also known as the conjunction of A and B and is written as $P = A \cdot B$ or $A \wedge B$.

the logical product of A and B and the process of obtaining it is the counterpart of arithmetical multiplication. In the instances we have chosen, S, the logical sum of A and B, is the compound proposition:

$$S \begin{cases} \textit{Either} & \text{"Socrates drank the hemlock"} \\ \textit{or} & \text{"Voltaire wrote } \textit{Gulliver's Travels.}\text{"} \end{cases}$$

P, the logical product of A and B, is, on the other hand, the compound proposition:

$$P \begin{cases} \text{"Socrates drank the hemlock"} \\ \textit{and} \\ \text{"Voltaire wrote } \textit{Gulliver's Travels.}\text{"} \end{cases}$$

Since we know that A is true and B false, S will be true but P false. Consequently when the truth value of A is 1 and of B zero, that of S will be 1 and of P zero. In the same way we can easily work out the truth values of S and P, given those of A and B in any other case. In general, as mentioned earlier, for S to be true *only one* of the two constituents A and B need be true, whereas for P to be true *both A* and B have to be true. This rule suffices to evaluate the truth values of S and P, as we shall now show.

Suppose both A and B are true so that the truth values of both are 1. Since S is true when either A or B or both are true, the truth value of S is 1. This leads to the summation rule:

$$1 + 1 = 1.$$

If both A and B are false, then obviously their logical sum S too is equally false so that the summation rule now is

$$0 + 0 = 0.$$

But if only one of the two, namely, A or B, is true, then S is also true, because S is true when *either* of them is true. This leads to the summation rules:

$$0 + 1 = 1; \quad 1 + 0 = 1.$$

TABLE 21

Logical Addition		
+	0	1
0	0	1
1	1	1

We may summarize these summation rules in the table of logical addition (Table 21).

Consider now the product proposition P. Since P is true only when both A and B are true, its truth value is 1 if and only if that of both A and B is 1. In every other case P is not true and therefore its truth value is 0. This leads to the product rules:

$$1 \times 1 = 1; \quad 0 \times 1 = 0; \quad 1 \times 0 = 0; \quad 0 \times 0 = 0.$$

They may be summarized in the table of logical multiplication:

TABLE 22

Logical Multiplication		
×	0	1
0	0	0
1	0	1

In defining both the notions of logical addition and multiplication of any two given propositions A and B, all that we have done is to link them by means of a conjunction—the conjunction "or" for their logical sum and the conjunction "and" for their product. But we could also couple them by means of other connectives like "if," "if and only if," "or else," and others of similar character. One of the chief tasks of logic in its study of the structure of propositions and of the general conditions of valid inference is the analysis of the properties of such logical connectives and the evaluation of truth values of compound propositions made up by stringing together a number of elementary propositions by means of them. This is the reason underlying Tweedledee's witticism: "Contrariwise, if it was so, it might be; and if it was, it would be; but as it isn't, it ain't. That's logic."

Take, for instance, the connective "if" used by Tweedledee. It is the basis of the fundamental notion of logic called *implication*. When we say that a proposition A implies another proposition B, we mean, in ordinary parlance, that "*if* A is true, *then* B is true," a statement which may be abbreviated as, "B if A." But what if A is *not* true? In that case we, the lay folks, do not care whether B is true or false. But not so the logician. He is less concerned with the truth or falsity of elementary propositions than with their mutual relations. For his motto is the verse:

"Not truth, nor certainty. These I forswore
In my novitiate, as young men called
To holy orders must abjure the world.
'If . . . , then . . . ,' this only I assert;
And my successes are but pretty chains
Linking twin doubts, for it is vain to ask
If what I postulate be justified
Or what I prove possess the stamp of fact."

This renunciation of truth and certainty is the price the modern logician has had to pay to purge logic of the proverbial cantankerousness both Bacon and Newman obviously had in mind when the former attributed to it the power to make men contentious and the latter recollected an acquaintance's complaint that the Oriel Common Room stank of "logic." For centuries men have used logic to confute one another like Omar's two-and-seventy jarring sects; it is only recently that they have realized that the "if-then" tack of modern logic is the only way of eliminating controversy and obtaining universal assent in at least some spheres of human activity. The logician no longer asks anyone to believe in the truth of his premises. He merely asks them to agree that certain stated conclusions really do follow from them. This way of persuading may seem to be evasive, skirting as it does the truth of the matter, but it has one merit over such other alternatives as persuasion by brain washing, intimidation, invocation of sacred texts, or confusing the issues by trickery. Unlike all its other rivals it lacks completely the property of doublethink in that a logical argument cannot now be used to establish false statements just as readily as to establish true ones. This is why when the logician takes over our everyday notion of implication for further refinement he cannot afford to neglect the contingency when A happens to be false. He faces it squarely by subsuming the two cases—the case when A is *false* and B is *true* as well as the case when A is *false* and B is *false*—in the same notion of implication. Accordingly, when he says that A implies B or when he writes its symbolic shorthand $A \rightarrow B$, he excludes only the last-mentioned of the following four possible cases, namely, the coincidence of the truth of A and the falsity of B:

"B if A" or "A implies B" or "$A \rightarrow B$" means any of these three cases, but *not* the fourth.

- Case I: A true, B true;
- Case II: A false, B true;
- Case III: A false, B false;
- Case IV: A true, B false;

which is to say that the implication rules are

$$1 \rightarrow 1 = 1; \quad 0 \rightarrow 1 = 1; \quad 0 \rightarrow 0 = 1; \quad \text{and } 1 \rightarrow 0 = 0.$$

They too may be summarized in an analogous table of logical implication:

TABLE 23

Logical Implication		
$\rightarrow$	0	1
0	1	1
1	0	1

From the manner in which we have defined the extended notion of logical implication it is obvious that $A \rightarrow B$ merely means that though A is a sufficient condition for B, it is *not* necessary. For while A true suffices to ensure the truth of B, A false will imply a false B as readily as a true one. One curious consequence of this extension of our usual notion of implication is that a *false* proposition implies *any* proposition. Another is that $A \rightarrow B$ does *not* necessarily imply $B \rightarrow A$. Thus if $A \rightarrow B$, we may have the conjunction of *A false* and *B true*, case II of the four listed above, which is the one contingency expressly excluded if $B \rightarrow A$ is deemed to hold. However, since $A \rightarrow B$ requires the exclusion of case IV (A true, B false) and $B \rightarrow A$ that of case II (B true, A false), the simultaneous validity of $A \rightarrow B$ and $B \rightarrow A$ means either that A and B are both true or both false. In other words, A and B both have the same truth value 0 or 1. This

TABLE 24

Logical Equivalence		
$\equiv$	0	1
0	1	0
1	0	1

is why A and B are said to be equivalent and the notion of equivalence defined by the identity of their truth values is more restrictive than that of implication. For it is the outcome of a double implication, namely, $A \rightarrow B$ as well as $B \rightarrow A$, holding together, that is, "B if A" and "A if B," which may be compressed into the single expression

"*B if and only if A*" and written symbolically as $A \equiv B$ or $A \leftrightarrow B$. Hence the equivalence rule governing the use of the new connective "if and only if." While both 0's and both 1's combine to yield 1, the other two couplings, namely, 0 and 1 and 1 and 0, yield 0 as shown in the table of logical equivalence (Table 24).

Finally, the connective "or else" must be distinguished from the connective "or" used earlier to define the logical sum of A and B. The former is the exclusive "or" in the sense of the Latin *aut* (A or B but *not* both A and B) whereas the inclusive "or" of the logical sum has the sense of the Latin *vel* (A or B or both) so that the possibility of both A and B being true simultaneously is admitted. It therefore follows that the rules of the *aut* or exclusive sum, symbolized $A \mid B$ to distinguish it from the *vel* or inclusive sum $A + B$, would be as shown in Table 25.

TABLE 25

Logical Sum (exclusive)		
/	0	1
0	0	1
1	1	0

All the results of the various logical operations like addition, both inclusive and exclusive, multiplication, implication, and equivalence, shown in the afore-mentioned tables may be summarized in a single master table.

TABLE 26

Proposition A	Proposition B	A and B	A or B	A or else B	B if A	B if and only if A
		Logical product	Logical sum (inclusive)	Logical sum (exclusive)	Implication	Equivalence
		$A \cdot B$	$A + B$	$A \mid B$	$A \rightarrow B$	$A \equiv B$
1	1	1	1	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	0
0	0	0	0	0	1	1

The table shows that all the rules of logical addition, multiplication, implication, and the like, can be mathematized by an appropriate mathematical symbolism so as to make the rules of logical reasoning

formally similar to those of numerical computation in the binary scale, e.g.,

$0 + 1 = 1;\ \ 1 + 1 = 1;\ \ 1 \times 1 = 1;\ \ 1 \times 0 = 0;\ \ 1 \rightarrow 0 = 0$, and so on.

This process of symbolizing logic confers on it great power. For logic deals with rules whereby we can derive from a given set of elementary propositions, called premises, another set, called valid conclusions. Now we may be able to obtain directly without the help of a symbolic notation conclusions of the simple kind which are referred to in textbooks of logic by a syllogism in Barbara like

All men are mortal,
Socrates is a man,
Therefore, Socrates is mortal.

But when we are confronted with long concatenations of propositions, it is impossible to see the structure of their complicated relations without recourse to symbols. Take, for example, the proposition "*Either* not-both A and B *or* both of A or not-B and of not-A or B, is always true." If we try to analyze it in its linguistic form as written, it is difficult to see why it should necessarily be always true no matter what the truth values of its elementary constituents A and B. But if we use the symbolic language of logic we have been employing in writing the above tables, the proposition becomes the expression

$$\overline{A \cdot B} + (A + \bar{B}) \cdot (\bar{A} + B) \qquad (1)$$

Now the truth value of A and B may be 0 or 1. If we take the truth value of A to be, say, 1 and that of B, say, 0, and substitute them in expression (1), we have

$$\overline{1 \cdot 0} + (1 + \bar{0}) \cdot (\bar{1} + 0). \qquad (2)$$

Recalling that denial of a true proposition is falsity and vice versa so that $\bar{1}$ is 0 and $\bar{0}$ is 1 and applying the rules of logical addition and multiplication already cited, expression (2) becomes

$$\bar{0} + (1 + 1) \cdot (0 + 0) = 1 + (1) \cdot (0) = 1 + 0 = 1.$$

In fact, no matter what combination of truth values of A and B we choose to adopt, the truth value of the expression turns out to be 1 in every case, even when both A and B happen to be false. The compound proposition in question whose symbolic rendering is expression (1) is therefore always true as already stated. Any logical expression like (1), which is always true independent of the truth of the elementary propositions it contains, is called a *tautology*. Its logical antithesis or negation is called *contradiction*. The latter is therefore a logical expression that is always false no matter what the truth value of its constituent elementary propositions. Thus $A \cdot \bar{A}$, which asserts the truth of A as well as of not-A at the same time, is obviously always false. It is therefore a contradiction in terms. A less obvious instance of contradiction is the expression not-$(\overline{A \cdot B} + A)$, which too is always false. Thus suppose both A and B to be true. It follows therefore that the truth value of $\overline{A \cdot B}$ is 0 and that of $(\overline{AB} + A)$ is 1 so that the truth value of not-$(\overline{AB} + A)$ is 0. It can be shown in like manner that no matter what truth values we may choose to assign to A and B, that of the expression, not-$(\overline{AB} + A)$, is always zero.

To test whether any given logical expression is a tautology/contradiction or not, all we need do is to make a table presenting all possible combinations of its truth values. To do so we may proceed as follows. If the expression contains only one proposition A, we write

TABLE 27

A	1	0

(3)

If it contains two propositions A and B, we write

TABLE 28

A	1	1	0	0
B	1	0	1	0

(4)

If it contains three propositions A, B, C, we write

TABLE 29

A	1	1	1	1	0	0	0	0
B	1	1	0	0	1	1	0	0
C	1	0	1	0	1	0	1	0

(5)

Each of the Tables 28 and 29 and others of like pattern are formed from the previous one by rewriting each column of the previous one twice, first with a 1 below it and the second time with a 0 below it. After writing the table it is easy to evaluate the truth value of the given expression corresponding to each possible combination of the truth values of the constituent elementary propositions like A, B, C, and so on. Thus to evaluate $(A \,.\, B) \rightarrow A$ we begin with Table 28 and use it to compute the truth values of A, $A \,.\, B$, and thence $(A \,.\, B) \rightarrow A$. These truth values follow from the definitions of logical multiplication and implication already given. The result is as shown below:

TABLE 30

A	1	1	0	0
B	1	0	1	0
$A \,.\, B$	1	0	0	0
A	1	1	0	0
$(A \,.\, B) \rightarrow A$	1	1	1	1

It will be seen that no matter what the truth value of A and B, that of $(A \,.\, B) \rightarrow A$ is always 1 so that the expression is a tautology. We

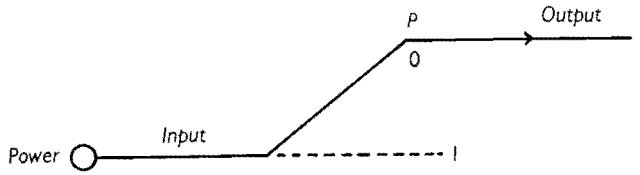

Fig. 23 Inversion switch. Current flows only if switch is in the open position denoted by 0 and is inhibited when it is in the closed position denoted by 1.

may similarly construct truth tables to prove that the expression $A + \bar{A}$, $\overline{A \,.\, B} + (A + \bar{B}) \,.\, (\bar{A} + B)$ and $\overline{AB} + A$ are always true.

Such symbolic evaluation of the truth values of long chains of elementary propositions by means of truth tables is grist to the computer mill because, as Shannon was the first to show, symbolic logic

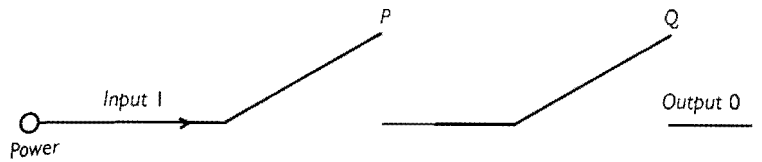

Fig. 24 Logical product circuit. Current flows if and only if both P and Q switches are closed.

devised by logicians to illumine the process of logical deduction can in fact be applied to automatic switching circuits used by communications engineers. Indeed, to each logical connective like "not," "and," "or," "implies," and so on, used to link elementary propositions there corresponds a circuit which is the physical embodiment of corresponding logical processes such as negation, multiplication, addition, implication, and equivalence. Thus a simple inversion switch P shown in Fig. 23, where the current flows only if it is open or "off," implements negation, that is, yields a proposition which holds only if its denial is false. Likewise, two simple switches P and Q joined in series as in Fig. 24 embody the logical idea of multiplication, for both propositions, namely, "switch P is closed" and "switch Q is closed," have to be true if current is to flow from the input I to the output O. On the other hand, a union of two switches in parallel as in Fig. 25 is an analogue of logical addition of the afore-mentioned two propositions, for either one or the other or both suffice to let the current flow in the channel. The dual control switch in a home that enables one to put on or off a light both from upstairs or downstairs is a slightly amended version of the parallel circuit embodying logical addition. As we saw, the parallel circuit of Fig. 25 allows current flow when either switch P or Q or both switches P and Q are closed. But a household switch is an arrangement whereby either P or Q but

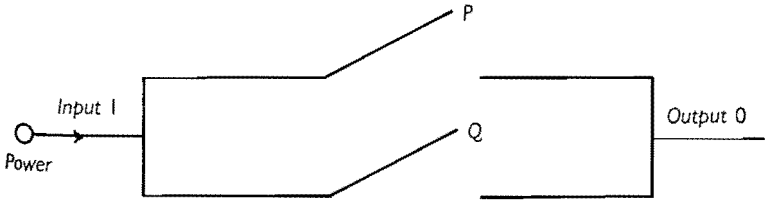

Fig. 25 Logical (inclusive) sum circuit. Current flows if either P or Q or both switches are on.

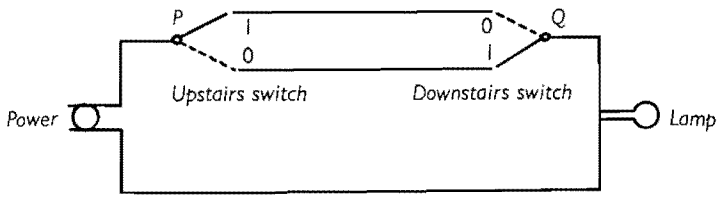

Fig. 26 Logical (exclusive) sum circuit. Current flows if only one of the two switches P or Q is on.

not both can light the lamp. Thus consider Fig. 26 where P denotes the upstairs and Q the downstairs switch. If we use the symbols 0 and 1 to represent the off and on positions of the two switches and the lamp, we have the following four alternatives:

TABLE 31

Physical Indication	Symbolic Representation
P off and Q off = lamp off	$0 + 0 = 0$
P on and Q off = lamp on	$1 + 0 = 1$
P off and Q on = lamp on	$0 + 1 = 1$
P on and Q on = lamp off	$1 + 1 = 0$

In other words, the circuit is an embodiment of the exclusive or *aut* logical addition. Incidentally it also represents arithmetical addition except that in the case of $1 + 1$ it yields 0 instead of 10. To turn it into a binary (arithmetic) adding appliance all that we need therefore do is to provide a place for the "carry" digit 1 that is missing in our scheme. We do so by including in the circuit a *second* lamp which lights if and only if both P and Q are on as in Fig. 27. The joint indication of both, that is, the second lamp on and the first off, symbolically written 10, yields the correct result of adding 1 to 1, obtained

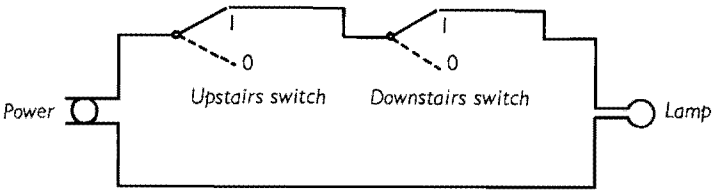

Fig. 27 Carryover circuit.

by putting on both switches. The amended arrangement therefore may be used as a binary adding machine.

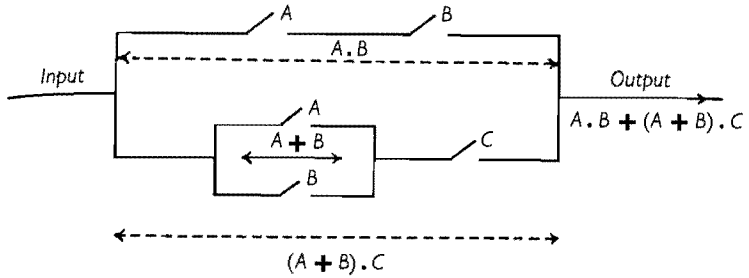

Fig. 28

Of the afore-mentioned five circuits materializing some of the basic logical operations, the first three, namely, the negation (Fig. 23), multiplication (Fig. 24), and addition (Fig. 25) circuits are fundamental. For we can build out of them others to implement *any* logical operation or their combination. Consider, for example, the logical operation of implication denoted by the symbol [→]. It can be shown that the proposition "$A \rightarrow B$" is equivalent to $\bar{A} \vee B$. For if we form the truth table corresponding to these two propositions in the manner described earlier, we find that both are true or false together no matter what may be the truth values of their elementary components A and B. See Table 32:

TABLE 32

TRUTH TABLE OF "$A \rightarrow B$" AND $\bar{A} \vee B$

A	1	1	0	0
B	1	0	1	0
$\bar{A}$	0	0	1	1
$A \rightarrow B$	1	0	1	1
$\bar{A} \vee B$	1	0	1	1

Likewise equivalence, which is double implication, may be defined in terms of these three operations. For $A \equiv B$ means in effect that

$A \rightarrow B$ *and* $B \rightarrow A$. Consequently $A \equiv B$ is merely the conjunction $\bar{A} \vee B \,.\, \bar{B} \vee A$. Since all logical operations as well as their combinations may be reduced to a sequence of three fundamental operations of negation, addition, and multiplication, and since the latter in turn can be actualized by electrical circuits, it follows that we can build electrical machines which can compute truth values of long chains of propositions strung together by means of logical connectives. This is why, as A. M. Turing, W. Pitts, J. von Neumann, and others have shown, effectively constructive logics are best studied in terms of automata with electrical networks providing physical replicas of logical propositions. Consider, for instance, the compound proposition $A \,.\, B +$

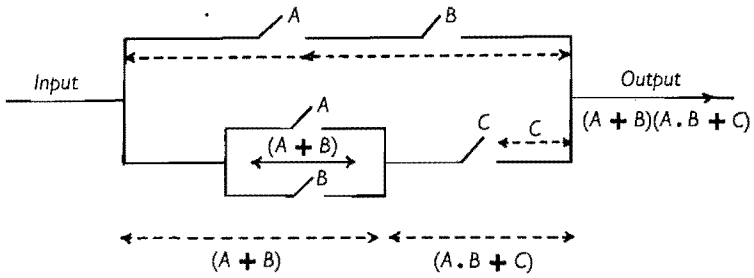

Fig. 29

$(A + B) \,.\, C$. Its physical representation is the electrical network shown in Fig. 28. Viewed from the input end, the circuit is evidently the *sum* of two *products* $A \,.\, B$ and $(A + B) \,.\, C$ yielding the expression $A \,.\, B + (A + B) \,.\, C$. But the same circuit may also be viewed differently from the output end. It is then seen to be the *product* of two *sums* $(A + B)$ and $(A \,.\, B + C)$. (See Fig. 29.) It therefore follows that both expressions $AB + (A + B)C$ and $(A + B)(A \,.\, B + C)$ are equivalent in respect of their truth value, as we may also verify directly by evaluating the truth tables of both expressions in the manner previously described. This is one elementary instance of how a purely logical problem may be evaluated by resort to switching circuits. Another is the well-known Kalin-Burkhart logical truth calculator, which is merely a piece of complicated electrical circuitry that translates various kinds of connectives like "and," "or," "if," "if and only if," etc., into their electrical counterparts like switches in series or parallel and other similar outfits. The problem it is required to solve may be of the type given by Irving M. Copi in his *Symbolic*

Logic: "If old Henning wants to retire, then he will either turn the presidency to his son or sell the business. If old Henning needs money, then either he will sell the business or borrow additional capital, but old Henning will never sell the business. Therefore, if he neither turns the presidency over to his son nor borrows additional capital, then he neither wants to retire nor needs money." With the letters R, T, S, N, and B denoting respectively retiring, turning, selling, needing, and borrowing, the problem boils down to evaluating the truth value of the compound proposition:

$$[R \rightarrow (T + S)] \,.\, [N \rightarrow (S + B) \,.\, (\bar{S})] \rightarrow [(\overline{T} \,.\, \bar{B}) \rightarrow (\bar{R} \,.\, \bar{N})] \qquad (6)$$

As will be observed, expression (6) involves five elementary propositions (and their negations) linked together with connectives like [→], [+], [.], and so on. To evaluate its truth value we may adopt the procedure described earlier to test whether or not a given logical expression is a tautology. In this case, however, we have to start with a longer truth table of 32 columns to obtain all possible combinations of the truth values of five propositions. If we form such a table, we may easily derive the truth value of each bracket like $(T + S)$, $[R \rightarrow (T + S)]$, and so forth, appearing in expression (6) by repeated application of rules of logical addition, multiplication, implication, and negation. A glance at Table 33 at the end of this chapter shows how the truth value of the total expression (6) may be derived from piecemeal evaluation of its components for each one of the possible 32 combinations in question. It will be seen that the truth value of expression (6) is always 1, so that it is a tautology. The logical truth calculator merely turns this column-wise bracket-by-bracket scanning into a routine and evaluates the truth value of each combination by recourse to suitable switching circuits.

Because electrical circuits can be devised to carry out the operations of logical addition, multiplication, denial, implication, equivalence, and so on, and because of the formal *identity* of the operations of addition and *multiplication* in logic and arithmetic in the binary notation we have already noted, the apparatus devised to perform logical operations can also be made to do those of arithmetic. Such devices, that is, automatic digital computers, simulate "thinking" by making calculations in a manner entirely analogous to that of any ordinary computation of our daily life. If we analyze the process of cogitation that such a computation entails, we may distinguish a number of stages. To start with, our brain calculates by recalling the results of

multiplication and addition tables learned by heart in early childhood. It then remembers the partial answers of each such calculation either by retaining them in the head or by recording them on paper if they exceed the brain's retention capacity. In all this it also exercises a strict control over the sequence of calculations to be performed, deciding at each step what to do next. There are, no doubt, various short cuts to computation such as consulting logarithmic tables for multiplying and dividing large numbers or alternatively employing ordinary desk machines for all the four basic operations of arithmetic. But, even so, the benefit derived from these methods is not fully realized because of our own all too human incapacity to work as fast as a machine. For example, a mechanical desk machine that we may press into service adds, subtracts, and multiplies much faster than it takes us to read an answer off it and note it down on a work sheet. In a fraction of the time it takes us to hunt through a book of tables for a particular number, say, the logarithm of 567, a machine can scan the table and sense the coincidence of 567 with its logarithmic associate. It follows that if we could also entrust to a machine the progressing of the various stages of computation, in addition to each basic arithmetical operation involved in the computation, we could give our usual computational techniques a new appearance. This possibility was first clearly foreseen over a hundred years ago by Charles Babbage though its full realization came only in our own day barely twenty years ago. Babbage saw that of the two functions of the human computer programing—planning the sequence of computational work and executing the program—only the former requires intelligence and therefore cannot be delegated to a machine. But if a program can be made sufficiently explicit and detailed, then the second function of slavishly following it could be done better and faster mechanically.

A glance at Fig. 30, depicting the parallel ways in which a human and a digital computer function, shows how closely the two resemble each other. In place of the program sheet according to which the human computer proceeds and the work sheet on which he records his interim results, the digital computer has respectively a program store and a register. Instead of the desk machine for doing basic arithmetic it carries an arithmetical unit, and in lieu of the sets of tables the human computer consults or carries in his head, it has another more permanent store called memory. The human computer himself yields place to a control system, which, when one

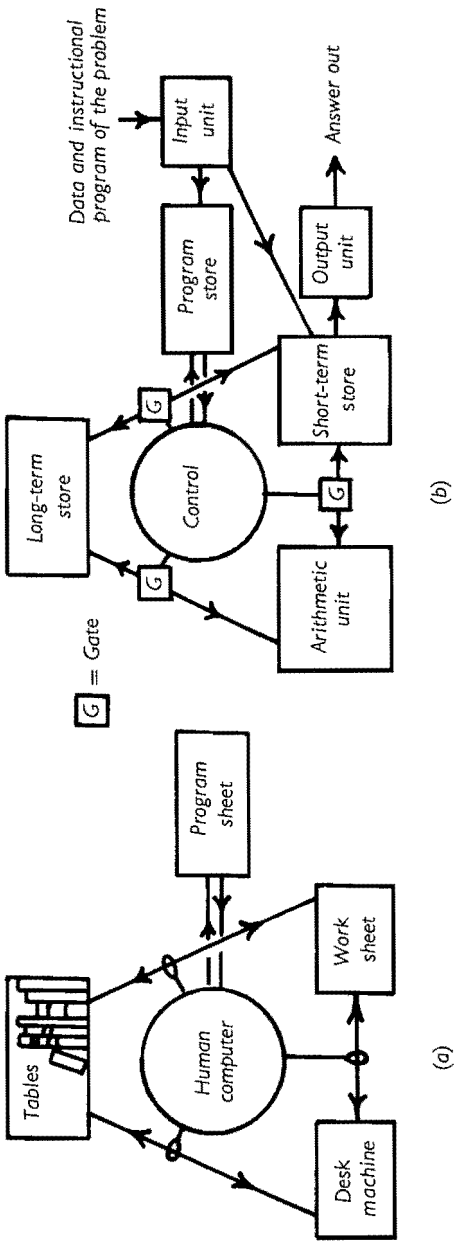

Fig. 30

operation in the program sequence is complete, "informs" the program store and receives from it the next instruction to be carried out. This is secured by suitably operating a number of logical control organs called gates. But besides program store, register, memory, and arithmetic unit—the respective counterparts of the human computer's work plan, work sheet, sets of tables, and desk machine—a digital computer has to incorporate two additional organs not needed by the human computer. They are the input and output units. The former is required to translate the data or information from the outside world in a language intelligible to the machine, and the latter to reconvert the result produced by it in a form that we of the outside world can understand. Now the language of the digital computer is determined by the fact that it is essentially an assembly line of

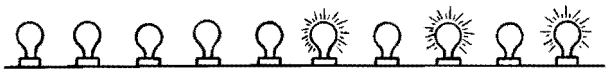

Fig. 31

switches or relays each one of which is capable of exactly two states, "on" or "off," which we may designate as the "energized" or "stimulated" state and the "de-energized" or "unstimulated" state. If we denote the former by 1 and the latter by 0, any input signal in such a machine is merely a row of 0's and 1's. That is, the alphabet of the machine language in which its "words" are written is the set of binary digits 0 and 1.

As we have seen in Chapter II, numbers may be written in the binary notation that uses only two digits, 0 and 1, instead of the ten digits of the familiar decimal notation. Thus the number 21 in the decimal notation is the number 10101 in the binary notation. (We may recall that it is merely the number $1(2)^4 + 0(2)^3 + 1(2)^2 + 0(2)^1 + 1 = 16 + 0 + 4 + 0 + 1 = 21$ in the usual notation.) It may therefore be represented by a sequence of, say, ten relays or electric lamps of which the first, third, and fifth from the right are on and the rest off as shown in Fig. 31. Incidentally we choose ten lamps to enable the same setup to represent larger numbers when required. Otherwise, for the number 21 itself five lamps would do. But rows of 0's and 1's or their physical counterparts like rows of lit and unlit lamps or rows

of on and off relays suffice to provide a common lingua franca for writing not merely numbers but also instructions specifying the operations to be performed on those numbers. For its "words," which are mere chains of 0's and 1's, can be stored in the computer memory, which is composed of a large number of memory boxes like mailboxes. To do so each memory box includes a row of electric lamps (or relays) wherein a "word" is stored by keeping an appropriate number of lamps on and the remainder off. In this way we can manage to place in the memory box the information-loaded word to be memorized almost as mail is put into a mailbox. The mailbox analogy extends even to the point of assigning to each memory box an "address" which again is a number like the postbox number. Thus if a computer has 128 memory boxes, each a set of lamps or relays, their respective addresses might be numbered successively from 0 to 127. This enables us to make a "word" or chain of 0's and 1's represent an instruction in addition to a number. All we need do is to adopt a code that will tell the machine whether a word stored in its memory is a number or an instruction. For example, we may adopt the following code: that if a word is an instruction, the three digits farthest left will be all 0's; the next three digits, the fourth to sixth, will denote the operative part of the instruction; the seventh digit will be 0, and the last seven the address of the operation in the instruction. The reason for reserving three digits for the specification of the operation in the instruction is that there are in all eight possible permutations of three binary digits, each of which can serve as a code for eight different operations like addition, multiplication, division, or transfer from one memory box to another, and so forth. Thus we may represent addition by 000, subtraction by 001, division by 101, and so on. The last seven digits yield $2^7 = 128$ numbers from 0 to 127 and thus can be used to denote 128 addresses of the memory boxes of our foregoing illustration. For example, the seven-digit sequence 0010101 denotes the memory box address 21, and so on. Under such a code a fourteen-digit word like 000, 101, 0, 0010101 means the instruction, "Divide the present content of the register by that of the memory box number 21," though the chain of 0's and 1's looks like a number. A similar code suffices to write even a blanket type of instruction which is somewhat more elaborate. For it does not specify any particular operation but only a ramification of possibilities among which the machine is required to choose in the light of results of its earlier computation. Thus the machine may be made to do one thing

if the result of a computation happens to exceed, say, 50, and quite another otherwise. In other words, the machine itself decides what to do next depending on the result of its own computation one stage earlier. The machine therefore makes the decision on the basis of its present stage, information about which is "fed back" into its controls. This is, indeed, our old friend feedback in yet another guise. It has been utilized to spiral computer programs to increasingly dizzy if more sophisticated heights.

Having devised a way of translating the program instructions both specific as well as blanket into the machine's language, all that is required is to arrange them in a sequence and process each successively under the guidance of a master clock making each instruction nominate its own successor. This may take the form that the successor of an instruction at address x will be the command at address $x + 1$ unless the machine is otherwise ordered that it is to be found at another address y. Once the complete program of instructions as to what the machine is to do at each stage of computation is written out in the binary alphabet 0 and 1 of the language the machine can comprehend, it is no great matter to embody it in computer hardware. For there are many physical outfits that can record the input information in computer language for feeding into the machine. Earlier we mentioned the electric lamp and the relay whose unlit or "off" position denoted the digit 0 and lit or "on" position the digit 1. But there are many others. A location on a punched card can represent 0 if no hole is punched there and 1 if a hole is punched; a location on a magnetic tape can represent 0 if it is not magnetized and 1 if it is magnetized. A punched paper or magnetic tape may therefore be used as an embodiment of input instructions. If we use punched paper tape for the purpose, an electromechanical device ensures that a hole in the tape will be interpreted by the computer as 1 and its absence as 0. In the case of a magnetic tape, another appliance exists that interprets a magnetized location as 1 and a non-magnetized spot as 0. After the program is taped out, the paper or magnetic tape is run through the computer. The tape reader translates the holes or magnetic spots into 0's and 1's according to the pattern impressed thereon. The computer then calculates the result and offers it naturally in its own language, which may be recorded by making the computer punch a paper tape. Thus with an input tape which is recording received information the computer yields an output tape of "processed" information. The latter is then put into a machine that

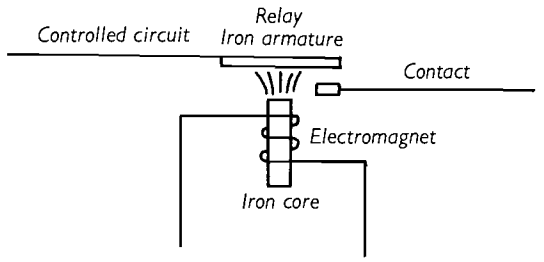

Fig. 32

does the reverse of punching, that is, looks at the paper tape and translates the message punched thereon back into the language of our daily intercourse.

Although we have explained the functioning of computers by means of such simple components as lamps and relays, they are in practice subject to several crippling limitations. First, a memory or "store" built out of them would occupy a great amount of space even if it had no more than a score or two of memory boxes. Secondly, they consume a great deal of power which finally appears as heat and must be drained off lest the computer melt. Thirdly, a memory box of electric lamps is very awkward to manipulate physically although this particular handicap is less serious in the case of relays whose reaction time is much less. For the latter can be turned on or off in less than a hundredth of a second as against a second required to reach and turn a light switch by hand. For these reasons vacuum tubes, crystal

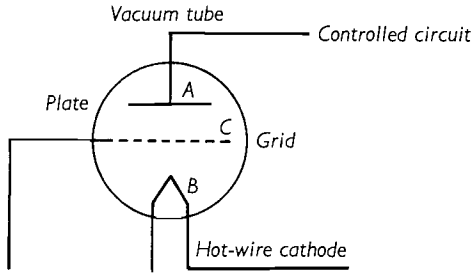

Fig. 33

diodes, transistors, ferromagnetic cores, and so forth, are better suited than lamps or even relays to serve as basic active components of digital computers. The vacuum tube, for example, acts 1000 or 10,000 times faster than a relay and also occupies much less space because it employs a more sophisticated technique. Thus while the relay depends for its action on the ability of a current swirling round an iron core to magnetize it (see Fig. 32), the vacuum tube employs the less tangible thermionic property of heated metals to emit electrons to obtain a current flow. Fig. 33 is a block diagram of a vacuum tube where a positive potential on the plate A attracts electrons emitted by the hot-wire cathode B thus allowing current to flow in the controlled circuit. It can also be inhibited by keeping the grid C at a suitable positive potential so that it traps all the electrons emanating from the cathode before they can reach the anode A. A transistor in all essentials is a still tinier version of the vacuum tube wherein the crucial grid-to-cathode distance BC of about 10^{-1} to a few times 10^{-2} cm is further reduced by a factor of 10. But even a transistor is by no means the ultimate in miniaturization. Thanks to superconductivity of certain materials, whose electrical resistance vanishes completely at temperatures near absolute zero, or 273.1° C. below the freezing point of water, it is possible to make a yet more compact device wherein the counterpart of the grid-to-cathode distance is diminished by another factor of 10^2 to 10^3. Perhaps a better idea of the relative compactness of these various kinds of components available for assembly into a computer can be given by quoting the number of components of different kinds that may be packed in any given space rather than the size of the operative cathode-to-grid distance in the case of a vacuum tube and its counterparts, the so-called "whisker electrodes" of a transistor and the thickness of the superconducting film in a cryotron. A cubic foot of space which could barely house a few hundred vacuum tubes could comfortably accommodate a few thousand transistors and some million cryotrons. Despite the enormous range of sizes of these three types of components, their reaction times are pretty much the same, about one to twenty microseconds.

Unfortunately memories built out of such swift-acting components have drawbacks. They are expensive and yet have rather low storage capacities. Because quick-in-the-uptake memories, despite their high cost, do not lead to extensive memory capacities,* most

* This may not be true of computers built with cryotrons but none such exist at present as swift-acting cryotrons have been invented only very recently. Their use as computer components depends on our ability to solve a number of unexpected problems that have cropped up though the cryotron principle still looks reasonably promising.

machines at present carry a short-term store of small capacity made up of swift components (vacuum tubes) and a long-term store of great capacity built out of slow-acting parts (relays), with a few stores of intermediate capacities in between these two extremes. Thus arise hierarchies of memories of various levels that a large-scale modern high-speed digital computer contains. For instance, it may incorporate a sequence of four or more kinds of stores: a large and a small store with two or more buffer stores of capacities N_1, N_2, N_3, N_4 "words"† and of reaction times t_1, t_2, t_3, t_4, respectively, so that these capacities become more extensive as the reaction times become less exacting. That is, the larger the storage capacity (N), the greater the reaction time (t) of the components of which it is made. The N- and t-sequences therefore march in step. As the former progresses with $N_1 < N_2 < N_3 < N_4$, the latter advances with $t_1 < t_2 < t_3 < t_4$, ensuring that N_1 words are required at reaction time t_1, N_2 at t_2, and so on. The reaction time t_1 of the first level corresponds to the basic switching time of the machine. It is the smallest store the machine can have, usually having a capacity of at least three or more words and reaction time of 5 to 20 microseconds. The memory organs in use in the second level have memory capacities N_2 ranging from 10^3 to 10^4 words, with reaction time t_2 about 5 to 10 times that of the previous level, t_1. The capacities of the higher levels increase geometrically by a factor of 10 at each step with reaction times rising even faster. As a general rule, memories appearing earlier in the hierarchy which require the fastest components are built out of certain electrostatic devices and magnetic core arrays. For the later levels of the memory hierarchy magnetic drums, magnetic tapes, and relays are mostly in use at present.

It is the lightning speed with which even the slowest of these components acts that is the chief asset of the digital computer. Without it the machine would be much worse at its job than a human calculator, considering the myriads of basic arithmetical operations the machine has to perform before it can answer the simplest problem put

† If each word is a sequence of, say, p binary digits 0 or 1, the capacities would be N_1p, N_2p, N_3p, N_4p "bits" or pieces of yes-no information in which memory capacities are usually measured. For obviously a word sequence of p binary digits carries p bits of information.

to it. Thus, to obtain a single addition of only two numbers like 5676 and 4198, it may have to perform some thirty or more separate additions of binary digits 0 and 1 including "carryover" additions. For more complex problems the total number of elementary arithmetic operations would run into millions and more. While the electronic speed of operation of the individual components enables the computer to handle them all in a few minutes or hours, the programing of such enormous numbers of individual computations may seem at first sight to be a hopeless affair.

Fortunately the situation is made a good deal easier by resort to a type of recurring or iterative process which has merely to be repeated many times with different numerical data. Once such a routine has been found and programed, it is possible to make the machine do it over and over again as often as required, supplying the machine with numerical data for each encore of the same basic computational theme. Consider, for example, the evaluation of the square root of any number N. The usual method of square-root extraction we learned at school is not very suitable for mechanization. But there exist iterative schemes which yield progressively a more exact approximation x_{i+1} from a less exact approximation x_i. It is simply the formula

$$x_{i+1} = \frac{1}{2}\left(x_i + \frac{N}{x_i}\right). \tag{7}$$

Thus suppose we require the square root of $N = 4$. We know, of course, that it is 2. But if we did not know it and made any wild guess by taking it (x_1) as, say, 4, then our formula (7) provides us a much closer approximation:

$$(x_2) = \frac{1}{2}\left(4 + \frac{4}{4}\right) = \frac{5}{2} = 2.5.$$

From $x_2 = 2.5$, we may obtain a still better approximation (x_3) by a second application of the same formula, namely,

$$x_3 = \frac{1}{2}\left(\frac{5}{2} + \frac{4}{\frac{5}{2}}\right) = \frac{41}{20} = 2.05.$$

From x_3 we may derive a yet closer approximation (x_4) by another repeat performance of the same routine, so that

$$x_4 = \frac{1}{2}\left(\frac{41}{20} + \frac{4}{\frac{41}{20}}\right) = \frac{3281}{1640} = 2.0006.$$

Thus even if we start initially with a grossly inaccurate estimate, as we have done, we begin to converge upon the actual result in barely four to five iterations. Although we cannot usually foresee and program how many iterations will be necessary to obtain an answer of the prescribed level of accuracy, the machine can itself be made to determine the number of encores it need perform. For if the machine at the end of each iteration is made to square the answer and subtract it from N, then the result is obviously zero only when the answer is the actual square root $\sqrt{N}$ we are in search of. So long as the answer continues to be non-zero, the machine can easily be made to recognize the fact, and programed to continue the iteration until the difference shrinks to zero. Finally, when the difference does vanish, the machine can be instructed to stop the iterative process and proceed to execute the next instruction taped in the program.

Iterative routines are the warp and woof of computer programs. For the machine finds it much easier to execute such routines than more familiar methods of computing a required result. The routines, no doubt, often involve many more arithmetical operations than the computations, but in view of the rapidity with which digital computers execute them (e.g., a million additions per hour) such proliferation of arithmetical operations is no longer an obstacle. As a result there has come to the fore a machine's-eye view of computation very different from the usual one. First, as we have seen, the machine prefers iterative routines even if they involve far more numerous elementary operations than the more familiar and direct methods of computation we know. This is one instance where a roundabout way is quicker than a crow flight. Secondly, there has occurred an inversion of means and ends. What were formerly considered as means of achieving certain mathematical ends have in this machine age become ends, and vice versa. For example, it was usual to consider a problem as "solved" if we managed to obtain a set of linear simultaneous equations leading to the postulated result even if the number of equations happened to be too large to be soluble in any practical way. Thus, suppose we wanted to find the values of x and y for which the quadratic expression

$$(4x - 3y - 5)^2 + (6x + 7y - 19)^2 \tag{8}$$

is minimum. Since this is the sum of two square terms, it cannot possibly be negative. Consequently, the minimum value is zero, which it assumes for just those values of x and y for which both

expressions $(4x - 3y - 5)$ and $(6x + 7y - 19)$ taken separately vanish. In other words, quadratic expression (8) attains its minimum value for such values of x and y as satisfy the two simultaneous equations

$$\begin{aligned} 4x - 3y - 5 &= 0 \\ 6x + 7y - 19 &= 0. \end{aligned} \tag{9}$$

But if quadratic expression (8) had been a sum of fifty squares of fifty linear expressions, each containing fifty variables, its minimum value *mutatis mutandis* would be given by those values of variables which are solutions of a set of fifty linear equations. We could still regard the

TABLE 33

TRUTH TABLE FOR EVALUATING THE TRUTH VALUE* OF

$$\underbrace{[R \rightarrow (T + S)]}_{a} \cdot \underbrace{[N \rightarrow (S + B)}_{b} \cdot \bar{S}] \rightarrow \underbrace{[(\bar{T} \cdot \bar{B}) \rightarrow (\bar{R} \cdot \bar{N})]}_{c}$$

Elementary Proposition	Truth Value													
	Column Number:													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
R	1	1	1	1	1	1	1	1	1	1	1	1	1	1
T	1	1	1	1	1	1	1	1	0	0	0	0	0	0
S	1	1	1	1	0	0	0	0	1	1	1	1	0	0
N	1	1	0	0	1	1	0	0	1	1	0	0	1	1
B	1	0	1	0	1	0	1	0	1	0	1	0	1	0
$T + S$	1	1	1	1	1	1	1	1	1	1	1	1	0	0
$R \rightarrow (T + S)$ (or a)	1	1	1	1	1	1	1	1	1	1	1	1	0	0
$S + B$	1	1	1	1	1	0	1	0	1	1	1	1	1	0
$N \rightarrow (S + B)$ (or b)	1	1	1	1	1	0	1	1	1	1	1	1	1	0
$a \cdot b$	1	1	1	1	1	0	1	1	1	1	1	1	0	0
$\bar{S}$	0	0	0	0	1	1	1	1	0	0	0	0	1	1
$a \cdot b \cdot \bar{S}$	0	0	0	0	1	0	1	1	0	0	0	0	0	0
$\bar{T}$	0	0	0	0	0	0	0	0	1	1	1	1	1	1
$\bar{B}$	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$\bar{T} \cdot \bar{B}$	0	0	0	0	0	0	0	0	0	1	0	1	0	1
$\bar{R}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\bar{N}$	0	0	1	1	0	0	1	1	0	0	1	1	0	0
$\bar{R} \cdot \bar{N}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\bar{T} \cdot \bar{B} \rightarrow \bar{R} \cdot \bar{N}$ (or c)	1	1	1	1	1	1	1	1	1	0	1	0	1	0
$a \cdot b \cdot \bar{S} \rightarrow c$	1	1	1	1	1	1	1	1	1	1	1	1	1	1

problem as solved because a set of fifty linear equations is in principle soluble by recourse to the well-known determinant method. But such a solution is a mere figure of speech, as the actual values of fifty unknowns of a set of fifty linear equations are buried as deep under the dead weight of fifty-one fifty-order determinants as in the original fifty equations, it being no simple matter to evaluate even a single one of them. Under such circumstances, because of the computing facilities now available it is more feasible to think the other way around, that is, to regard the solution of the simultaneous equations being reduced to the minimizing of a quadratic form. In other words, given any set of simultaneous equations like the two shown

TABLE 33 (Contd.)

TRUTH TABLE FOR EVALUATING THE TRUTH VALUE* OF

$$\underbrace{[R \rightarrow (T + S)]}_{a} \cdot \underbrace{[N \rightarrow (S + B)}_{b} \cdot \bar{S}] \rightarrow \underbrace{[(\bar{T} \cdot \bar{B}) \rightarrow (\bar{R} \cdot \bar{N})]}_{c}$$

Truth Value																
Column Number:																
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1	0
0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1
1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	0	1	1
1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	1	0	1	0	0	1	0	1	0	1	0	1	0	1
1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

*The truth value of each elementary expression may be worked out by the application of the rules of ical addition, multiplication, negation, and implication described in the text.

in (9), we can derive a quadratic expression like (8) by simply adding the squares of their left-hand sides. The problem of solving the equations is then simply that of finding those values of the variables which minimize the quadratic expression so derived. In a similar manner the older method of solving a differential equation was to approach the solution through a sequence of functions each of which satisfies the equation but not all the initial or boundary conditions. However, in view of the computing facilities now provided by computers, the final solution is approached through a sequence of functions each of which satisfies all the boundary conditions but not the differential equation. The reason is that many forms which appear unduly complex according to the older point of view are very suitable for the new methods of numerical computation by calculating machines. Then again it is likely that the digital computer may also induce a more fundamental revision of the outlook of the mathematical physicists. For example, such physicists have hitherto adhered to the hypothesis of continuity of space and time not so much because of its logical validity, which a deeper analysis of Zeno's paradoxes has shown to be fictitious, but because of its mathematical convenience in that differential equations are usually more tractable than difference equations, at least analytically. But the increasing use of digital computers to solve such equations, which it does by turning them into difference equations, may well lead to the abandonment of the hypothesis or at least erode its main justification, that of mathematical facility or expediency. All this, however, is a small change in the transvaluation of old mathematical values required for the next significant advance in computer theory. For what exists today is merely "an imperfectly articulated and hardly formalized body of experience" underpinned by formal logic. Its debilities, in particular those of formal logic, begin to be evident when we compare the artificial automata with their natural counterpart, the human brain. A deeper mathematical study of both bids fair to inaugurate a mathematical revolution far more profound than the invention of the decimal system or the calculus, as we shall see in the next four chapters.

Chapter X

THE COMPUTER AND THE BRAIN

THE feats of control and communication that computers nowadays perform have earned them the nickname of "giant brains." The sobriquet is well deserved because of their ability to imitate, with a speed, skill, and accuracy far beyond the powers of any genius alive, certain processes of thought, though of a routine, repetitive, and rudimentary kind. But in reality a fair comparison of the powers and debilities of these artifacts vis-à-vis their natural model, the living brain, would show that the "giants" are yet in many ways mere playthings beside the "midget" they aspire to simulate. Consider, for instance, the basic component of the human nervous system, the nerve cell or the neuron, which is to the brain what a vacuum tube or a transistor is to the computer. It is still a marvel of compactness and economy. For reasons that will be apparent later on, it beats in respect of size its artificial analogues, the vacuum tube and the transistor, the basic components of computers, by a factor of a hundred to a thousand million. Even if we succeed in replacing the transistor with the more miniature superconducting cryotron—a promising possibility beginning to open up and yet to be fully realized —the lead factor will still be a hundred thousand to a million. This is why artificial imitations of living brains tend to be much more lavish in the use of space than their natural prototypes, though making up for their extravagance by being enormously swifter. But the swiftness in turn entails another handicap in that they require a lot of power to activate them. Whereas the energy dissipation by a typical neuron is barely one-billionth of a watt, a quantity so vanishingly small as to be for all practical purposes a synonym for zero, a typical vacuum tube would dissipate 5 to 10 watts and a transistor one-tenth of a watt. Once again the neuron is a more efficient unit outpacing its synthetic