object plane is 50 cm from the target and an area of 10×10 cm is viewed with a resolution of 0.5 mm, then the optimum value of $t\left(\frac{c}{\beta}\right)^2$ is approximately 2 ms. This means that with a contrast ratio of 0.1 (β taken as one, as is customary in physical measurements) the optimum integrating time, t, is equal to 0.2 second, which is the approximate integrating time of the eye.

There are other uses for the scanning x-ray system than that of intensifying the fluoroscopic image. It may be used to study the decay of fluorescence by employing the crystal that is to be studied as the detector screen for the photomultiplier tube, and viewing a small slit or aperture as the object. Since the writing time per line is of the order of 60 μs, then any fluorescence that lasts long compared to a fraction of a microsecond will appear to distort the image in the direction of scanning. In its present form the instrument serves as a low-power microscope with a magnification of some 10 diameters. With pinholes of a few microns in diameter, however, and a triple coincidence photocell circuit, the photomultiplier tube noise can be reduced to a negligible value, and, with the small number of photons that emerge through such a small hole, an image can be constructed if sufficiently long integrating time is used. Higher magnifications may be achieved in this way. Other uses include means for the production of short time pulses of x-rays, either singly or repetitively, and a means of a rocking type of Laue experiment for tiny crystals, wherein the crystal is held fixed and the source moves.

References

1. Broser, I., and Kallmann, H. *Z. Naturforschg.*, 1947, **2a**, 439, 642.
2. Coltman, J. Private communication.
3. Moon, R. J. *Am. J. Roentgenol. Radium Therapy*, 1948, **59**, 886.
4. *Ibid.* 1949, **62**, 637.

The Relations between Symbolic Logic and Large-Scale Calculating Machines[1]

Edmund C. Berkeley

Edmund C. Berkeley and Associates, New York City

SCIENTISTS IN MOST FIELDS are becoming familiar with the large-scale calculating machines—the so-called mechanical brains—that have made possible the solution of many mathematical problems hitherto considered insoluble. Only a relatively few scientists, however, understand symbolic logic. It is off the main path. What is it? And why is it important in relation to mechanical brains?

Symbolic Logic

Symbolic logic, in its broadest sense, is a new science that has the following characteristics:

(*a*) It studies mainly nonnumerical relations.

(*b*) It seeks precise meanings and necessary conclusions.

(*c*) Its chief instrument is efficient symbols.

Its closest cousin among the sciences is mathematics. But symbolic logic differs from mathematics; to make the differences clear, mathematics and symbolic logic may be compared in a number of respects.

Mathematics deals with words like *plus, minus, times, divided by.* Symbolic logic deals with more basic words like *yes, no, and, or, not, the, of, is, same, different, some, all, none.* Mathematics deals mainly with numbers and their properties. Symbolic logic deals mainly with statements, classes, and relations. Mathematics concentrates on answers to questions like: "How much?" "How many?" "How far?" "How long?" Symbolic logic deals with questions like: "What does this mean?" "Does this set of statements have conflicts or loopholes?" "What is the basis of this proof?"

An example of a rule in mathematics is, "The reciprocal of the reciprocal of a number is the number itself." An example of a rule in symbolic logic is, "The denial of the denial of a statement is the statement itself."

Historically, symbolic logic is the result of applying the powerful technique of mathematical symbolism to the subject matter of logic.

Content of Symbolic Logic

Many scientists comprehend the content of mathematics. But what is the content of symbolic logic?

[1] Based on an address before the Association for Computing Machinery, April 19, 1949, Oak Ridge, Tenn.

There are four or five fairly well-recognized branches of symbolic logic. One of these is Boolean algebra, the algebra of *and, or, not* and statements (or classes). For example, a rule from Boolean algebra is that "neither *a* nor *b* is the same as not *a* and not *b*." Here *a* and *b* are statements or classes, but not numbers. As a result of work by Claude Shannon, Boolean algebra has proved to be useful in designing and checking electrical circuits using relays or electronic tubes. This application of symbolic logic is important in the design and construction of automatic computers.

Another branch of symbolic logic is the one that deals with the foundations of mathematics. It has studied such questions as these: "What is a number?" "What is a variable?" "What is a mathematical function?" It has answered these questions to a large extent. One of the great books in the history of symbolic logic is *Principia Mathematica,* by Bertrand Russell and A. N. Whitehead (published 1910–13), which aimed to furnish a logical foundation for all of mathematics.

A third branch of symbolic logic is called the algebra of relations. This deals with such concepts as symmetric relations, transitive relations, connected relations, series, etc.

Still another branch deals with what is called the decision problem, i.e., the procedure for deciding that a statement is true or false. Symbolic logicians have investigated the problem of proving statements in any mathematical system. These studies have produced some remarkable results. For example, it can be shown that there are statements in arithmetic, and in other mathematical systems, that can never be decided as true or false. Nevertheless, mechanical brains can be applied to deciding statements that can be decided, in problems that would take years of human labor to decide.

So much for an introduction to symbolic logic. How is it related to large-scale calculating machinery?

Logical Operations in Large-Scale Calculating Machinery

When we use desk calculating machines, we notice the numerical operations in a calculation; but we tend to carry out the logical operations in our heads or on paper and we tend not to notice them. When we begin to use automatic computers that perform calculations in long sequences, we find at once that we must pay attention to nonnumerical reasoning operations, as well as to numerical operations. In fact, we notice a long series of questions, all in the territory of symbolic logic rather than in mathematics. Here are some of them:

1. What is the best way to give a machine instructions or orders?
2. What is a sufficient set of orders to instruct a machine to solve any possible problem?
3. What is the smallest set of orders which still leaves a well-functioning machine? (For example, in the method proposed by John von Neumann for instructing the automatic computer ENIAC, only about 70 orders are needed.)
4. How can machines be constructed to determine almost all their own orders?
5. Machines do elementary arithmetical operations like addition, subtraction, multiplication, division, and reference to a table; but what are elementary logical operations that machines should also do?

The last question can be answered in part. Some of the logical operations that may be found in some mechanical brains, and that need to be built into really versatile automatic computers, are decision, comparison, selection, choice, checking, matching, merging, tabulating, sorting. Additional logical operations, such as implication, exception, denial, are in process of being built into mechanical brains.

As long as we do mathematics using pencil and paper and, perhaps, a desk calculating machine, we can get along moderately well without symbols that express the logical operations. When we start dealing with mechanical brains, we find it helpful to express the logical operations in symbols. With these symbols, we grasp a power that we did not have before. Furthermore, these symbols can often be made to fit into ordinary mathematical language. Then machine routines are no longer half flesh and half fowl; they all take on the same mathematical, logical, symbolic nature.

Symbolic Logic in Expressing the Operation of a Counter Mechanism

As an example of the fusion of mathematics and logic called forth by the requirements of large-scale calculating machinery, let us look into an International Business Machines punched-card tabulator, and consider a counter mechanism that can handle numbers of 6 decimal digits. A tabulator is a machine that either lists the information contained in a series of punched cards, or prints totals derived from them, or does both. Its operation is controlled by a plugboard, an assembly of plugwires (or patchcords) connecting hubs (or terminals), all set in a frame. This wired-up frame expresses the instructions for a problem and is changed from problem to problem.

If we examine the plugboard frame of a tabulator, we find 4 inputs and 1 output for the counter mechanism. The 4 inputs are:

1. A set of 6 hubs, one for each digit, called counter entry, which takes in a 6-digit number *a*.
2. A single hub, called the add hub, which takes in a pulse *p*.
3. A single hub, called the subtract hub, which takes in a pulse *q*.
4. A single hub, called the counter total control hub, which takes in a pulse *r*.

Logically, a pulse can be only 1—the presence of a

pulse—or 0—the absence of a pulse; in the machine, however, the timing of the pulse is also important. The output is a set of 6 hubs, called counter total exit, which puts out a number b.

We can express the operation of the counter mechanism with two simple algebraic equations. Let the number held in the counter at any time be h. Then at any cycle x

$$h_x = h_{x-1}(1 - r_{x-1}) + a_x(p_x - q_x) + 999999\, p_x q_x,$$

and

$$b_x = h_x r_x.$$

What do these two equations mean? If just p is impulsed, the counter adds a. If just q is impulsed, the counter subtracts a. If both p and q are impulsed, the counter adds 999999. If r is impulsed, the counter total exit reads out the number held in the counter, and the counter is at the same time cleared. Obviously, the mechanism would be more flexible if we could read out the number in the counter mechanism without, necessarily, clearing it. In fact, IBM provides a modified counter mechanism with which this is possible.

Since a, b, and h are variables that can be regular numbers, this department belongs to mathematics. But p, q, and r are variables that can only be 1 or 0, like "yes" or "no," and this department belongs to symbolic logic. Only when we fuse the two departments can we exactly express the operation of the counter mechanism.

Symbolic Logic in Programming

We have not yet gone very far, however, into the application of symbolic logic to mechanical brains. Let us take another and more elaborate example of the uses of symbolic logic in large-scale calculating machinery. Let us take an actual problem, see how it can be programmed in a mechanical brain, and note how questions of logic actually occur. A problem that will require several subroutines is the one of finding the square root of a number using an iterative formula, or one that gives a better result each successive time we apply it. One such formula for square root is

$$x_{n+1} = \tfrac{1}{2}(x_n + Y/x_n),$$

where Y is the number for which we want the square root, and the x's are successive approximations. Each time we apply this formula we get a better approximation to the true square root. We begin by making any kind of rough guess about the square root of the number Y, and we call this first rough guess x_1.

To test the procedure, let us obtain the square root of 67.2. We choose 8 as a first guess, because 8 times 8 is 64, and 9 times 9 is 81, and 67.2 is in between these results. So 8 is our first approximation, x_1. In Round 1, 8 divided into 67.2 gives 8.4. The average of 8 and 8.4 is 8.20. This is x_2, our second approximation. It is correct to 3 figures. In Round 2, 8.20 divided into 67.2 gives 8.195122. The average of this number and 8.20 is 8.197561. This is x_3, our third approximation. It is correct to 6 figures. The result of the next round is 8.1975606125, x_4. This is correct to 11 figures. So we see that, with a reasonable guess and two or three divisions, we can obtain all the accuracy we can ordinarily use.

Good iterative formulas are like this: they approximate the true value quickly, and they are very useful on automatic computers. To perform this problem on a machine we recognize 5 subroutines: the subroutine for reading data from input into storage; for carrying out the formula once; for deciding whether to repeat the formula for another round; for preparing to repeat; and the subroutine for sending the answer from storage to output, and stopping. The third subroutine, deciding whether to repeat, is purely logical.

Mechanical Brain ZAC

Let us imagine and stipulate a simple mechanical brain, or automatic computer (ZAC, the "Z Automatic Computer"), able to do this problem.

ZAC has an input tape, an output tape, and 70 registers for storage ordinarily of 8 decimal digits. ZAC has a computer, and the computer can take in an operation OP on one channel, take in 2 numbers a and b on 2 more channels, and give out the result c on the fourth channel:

$$c = a\ OP\ b.$$

ZAC has a program register, which holds each successive instruction that governs the machine. We can transfer numbers or orders into and out of the program register. Some, but not all, numbers will have meaning as orders to ZAC. The program register regularly holds 5 digits: the first 2 digits will be the number of the order, n; the middle digit will be the kind of the order, k; and the last 2 digits will ordinarily be the number of a register, r. At any cycle, the order, or instruction, n, k, r, stored in the program register tells the machine what it is to do at that cycle.

The kind-of-order numbers, k, we shall suppose, can vary from 0 to 9. Using these 10 numbers, we have sufficient flexibility to tell the machine all that we want it to do in finding square root.

The register numbers r vary from 10 to 79. Registers 10–49 will usually store orders having the same order number as the register number; registers 50–69 will usually store numbers in the calculation; and registers 70–79 will usually store constants.

TABLE 1

FIVE-DIGIT ORDER (STORED IN REGISTERS 10–44)

Subroutine	Number, n	Kind, k	Register, r	Meaning
1. Reading data from input into storage	10	4	50	Input of Y to 50
	11	4	56	Input of x_1 to 56
	12	4	51	Input of ½ to 51
	13	6	19	Go to order 19
2. Carrying out the formula once	19	0	74	*Division* to computer
	20	1	50	Y to computer
	21	2	56	x_n to computer
	22	3	52	Y/x_n to 52
	23	0	71	*Addition* to computer
	24	1	52	Y/x_n to computer
	25	3	53	$x_n + Y/x_n$ to 53
	26	0	73	*Multiplication* to computer
	27	1	53	$x_n + Y/x_n$ to computer
	28	2	51	½ to computer
	29	3	57	½ $(x_n + Y/x_n) = x_n + 1$ to 57
	30	6	32	Go to order 32
3. Deciding whether to repeat or not	32	0	75	*Inequality* to computer
	33	1	56	x_n to computer
	34	2	57	$x_n + 1$ to computer
	35	3	69	T $(x_n \neq x_n + 1)$ to 69
	36	7	38	Go to order 38 if 1 is in 69
	37	6	43	Go to order 43
4. Preparing to repeat	38	0	70	*Transfer* to computer
	39	1	57	Old $x_n + 1$ to computer, equal new x_n
	40	3	56	New x_n to 56
	41	6	19	Go to order 19
5. Sending answer from storage to output and stopping	43	5	57	$x_n + 1$ in 57 to output
	44	9	44	Stop

The most interesting of these sets of numbers is the kind-of-orders, k. Every k is followed by an r. This is the meaning:

If k is 0, the machine transfers the operation stored in register r to the computer and goes to the next order numerically.

If k is 1, the machine transfers the number in register r to computer register a and goes to the next order numerically.

If k is 2, the machine transfers the number in register r to computer register b and goes to the next order numerically.

If k is 3, the machine transfers the computer result c to register r and goes to the next order numerically.

If k is 4, the machine transfers the number on the input tape to register r and goes to the next order numerically.

If k is 5, the machine transfers the number in register r to the output tape and goes to the next order numerically.

If k is 6, the machine is instructed to go to order r (obtaining it from register r), instead of (as the machine normally does) going to the next order numerically.

If k is 7, and if the number in register 69 is 1, the machine is instructed to go to order r; if k is 7, and if the number is not 1, the machine goes to the next order numerically.

If k is 8, and if the number in register 69 is 1, the machine is instructed to go to the order number stored in register r; if k is 8, and if the number is not 1, this order tells the machine to go to the next order numerically.

If k is 9, the machine stops.

Some of the register numbers that may follow the kind-of-order $k = 0$ are 70, 71, 72, 73, 74, 75. These registers contain signals that set the computer for 6 operations, respectively:

70, transfer,	$c = a$
71, addition,	$c = a + b$
72, subtraction,	$c = a - b$
73, multiplication,	$c = a \cdot b$
74, division,	$c = a \div b$
75, inequality,	$c = T(a \neq b)$

In the last operation, the expression $T(\ldots)$ means "the truth value of . . ." and is equal to 1 if . . . is true, and 0 if . . . is false.

THE PROGRAM FOR SQUARE ROOT

The program for square root using ZAC is shown in Table 1.

The first subroutine consists of orders 10, 11, 12. Here we read out from the input tape into registers of the machine. Then we proceed to order 19.

Subroutine No. 2 is now carried out. In orders 19–29, we cover the division, the addition, and the multiplication required by the iterative formula. Then we go to order 32.

In subroutine No. 3, in orders 32–35, we cover inequality; we test 2 successive approximations to see if they are equal or unequal. If they are unequal, we record a 1 in register 69. (In practice, a difference less than a certain tolerance would be accepted as equality.)

Now we come to a choice of program. Using order 36, we go to order 38 if, and only if, there is a 1 in register 69; in other words, if x_n and x_{n+1} are unequal. If there is a 0 in register 69—in other words, if the last two x's are equal—then we go to the next order, 37, and that routes us to order 43.

In orders 38–40, we remove x_n from the iterative formula, subroutine No. 2, and insert x_{n+1} instead. Then with order 41 we go to subroutine 2, which will now compute the next approximation.

In order 43 we read out the final value of x into the output tape, and with the next order stop the machine.

Thus we see how we can program square root with a machine.

In this program, we have to recognize the operation of inequality: this is logic rather than mathematics. We have to recognize different subroutines: this is logic rather than mathematics. We have to provide for the branching of instructions: this is logic rather than mathematics. We have to provide for the machine's deciding for itself when it will stop using a formula and, instead, give out the answer: this, too, is logic rather than mathematics.

These several examples illustrate some of the relations between symbolic logic and automatic computers. We can anticipate that there will be more and more

fusion between numerical mathematics, on the one hand, and nonnumerical reasoning, or symbolic logic, on the other. Machines that play games, machines that separate true combinations of statements from false combinations, other kinds of information-handling machines where emphasis is on logical competence rather than on mathematical competence, are already in existence. Symbolic logic, large-scale calculating machinery, and mathematics will continue to enrich one another in many significant ways.

The Traveling-Wave Linear Accelerator

E. R. Wiblin

Atomic Energy Research Establishment, Harwell, England

THE STUDY OF NUCLEAR PROCESSES commonly involves the bombardment of one nucleus with one or another of the fundamental particles. In many experiments, particularly early ones, the bombarding particles were those emitted by naturally radioactive substances, whereas in later work artificially accelerated particles have been extensively used. In experiments involving neutrons as the bombarding agent, production is normally a secondary process following the bombardment of a primary target either by a charged particle or by gamma radiation.

Familiar particle accelerators are direct current generators of, for example, the Cockcroft and Walton type, and cyclic accelerating devices typified by the cyclotron and betatron. In the former, particles are accelerated by passing through a single large-voltage gradient, whereas in the latter, energy is imparted to the accelerated particles by causing them to traverse a short intense gradient many times when constrained into an approximately circular path by a very strong magnetic field. For this purpose a large and expensive electromagnet is required.

Among the many electrical devices that have been made practicable by the great advances in radio technique of recent years is a new form of high-energy particle accelerator known as the Traveling-Wave Linear Accelerator. This machine has been fully described elsewhere, but a brief description of its mode of operation is, perhaps, not out of place here.

Electrons are introduced axially into a special form of evacuated radio wave guide, along which electric waves are made to travel so that their phase velocity increases steadily from a speed equal to that of the entering electrons up to nearly that of light. Most of the electrons are then constrained into "bunches" moving in constant phase relationship to the waves and are, therefore, accelerated with them.

In the accelerator recently installed at the Atomic Energy Research Establishment at Harwell in southern England, the final electron energy obtained may be as high as 3.2 mev with a mean current of about 120 μa. The electrons may be extracted from the machine by allowing them to emerge through a thin metal "window" at the end of the accelerating wave guide.

Use of magnetron valve. The radio waves are generated in a magnetron valve, such as was developed for radar use, at a wavelength of 10 cm. They occur in very intense pulses, 2 μs in length, and up to 500 pulses every second may be used. The current of electrons during the pulse is, therefore, of the order of 120 ma. If the large current of electrons from the machine is allowed to strike a heavy metal target, intense bursts of gamma rays are produced, and one use of the machine is to provide heavy doses for irradiation purposes.

This particular machine is, however, installed primarily as a neutron source. The gamma rays are converted into neutrons by photodisintegration in a target of heavy water. Some of the nuclei of the deuterium in the heavy water disintegrate and emit a neutron. Some of the neutrons emerge from the target and are available for experimental purposes. The machine will be used as a neutron source for time-of-flight measurements and, it is hoped, will prove a better source than has hitherto been available.

The linear accelerator is inherently suitable for this use since the neutrons are produced in bursts—corresponding to the pulses of radar waves from the magnetron. By a technique similar to that used for range determination in some radar equipment, it is possible to measure the time—which may vary in practical cases from a few microseconds to a few milliseconds—taken by neutrons to travel over a fixed distance from their origin in the source, to a detector (usually a proportional counter).

Calculating neutron velocity. A series of electronic "gates," opened in succession, allows only neutrons of velocities corresponding to the delay between the