MECHANIZED REASONING

Logical Computors and Their Design

By

D. M. McCALLUM, B.Sc., A.M.I.E.E.

and

J. B. SMITH, M.A., B.Sc., A.M.I.E.E., A.Inst.P.

Reprinted from

Electronic Engineering

April, 1951

MECHANIZED REASONING

Logical Computors and Their Design

By D. M. McCallum, B.Sc., A.M.I.E.E. and J. B. Smith, M.A., B.Sc., A.M.I.E.E., A.Inst.P.*

THE prominent part played by intuition and flashes of inspiration in the process of human thought obscures the fact that much reasoning involves merely the elimination of situations which conflict with clearly definable rules.

Such problems of elimination may be expressed mathematically and may thus be brought within the scope of modern high-speed digital computors. These computors are however generally designed to deal with great precision with numerical problems; on this account they tend to be large and complex. It is not generally realized that the logical problems involved are essentially very simple and may therefore be solved by relatively simple mechanical means.

The present article describes a "slow-motion" demonstration model built from a small number of readily obtainable components, but nevertheless able to solve logical problems which are not entirely trivial. There are in the internal circuits of the instrument only 17 relays and one stepping switch. Compared with the human brain with its 10^{10} cells the "level of intelligence" of the machine is thus extremely low, and the remarkable results obtained from it are due to the extremely specialized function of the elements and their connective units. Some indications are given later in the article of possible developments which lead to more sophisticated structures of greater "intelligence" and correspondingly greater economy of design.

Symbolic Logic

A mathematical treatment of logic has been described by Boole[1] and other writers. The method of treatment employs ideas such as the following: suppose A is a concept which is either true or not true. We denote A and "not A" by A, A̸ respectively. Then if B is another concept, various multiple concepts may be defined, for example:—

A and B	representing the combination AB
A or B	representing the combinations AB, AB̸, A̸B.
A or else B	representing the combinations AB̸, A̸B.
A if and only if B	representing the combinations AB, A̸B̸.
If A, then B	representing the combinations AB, A̸B, A̸B̸.

and so on.

As an example of the use of such symbols the following regulation may be considered. "Only members or their guests may play over the Blankshire Golf Club's course." This is evidently equivalent to

A if and only if (B or C).

where

A denotes eligibility to play over the course.
B denotes membership of the club.
C denotes the status of member's guest.

Now this multiple concept, by the above definitions, represents the combination AD, A̸D̸, where D is "B or C". Also by the above definitions, D represents the combinations BC, BC̸, B̸C, and hence D̸ represents the remaining combination B̸C̸. Combining the two results, the multiple concept represents ABC, ABC̸, AB̸C, A̸B̸C̸. Hence the classes of persons consistent with the regulation are as follows:

(i) ABC. Eligible persons who are both members and members' guests.
(ii) ABC̸. Eligible persons who are members.
(iii) AB̸C. Eligible persons who are members' guests.
(iv) A̸B̸C̸. Non-eligible persons who are neither members nor guests.

Any other combination must involve a logical contradiction, e.g., A̸BC̸. Members not eligible to play over the course, and so on.

Application of Systematic Logical Method

A logical problem may in general be solved by techniques similar to the above provided that it is capable of being reduced to a form which consists of one or more "rules" (the regulation in the above example) connecting a number of "variables" (which have only two possible states, as A, B, C in the example) by standard logical relations. The number of possible relations is not large; the definition of those which shall be regarded as "standard" is a matter of convenience; three are sufficient (e.g. "not," "and," "or") while the use of more than about eight tends to be confusing. The reduction of the problem to the desired form is not always easy, but is in general much less difficult than solving the problem completely. Once the problem has been reduced to the standard form, it may be "solved" by a machine. The solution will in general consist of the finding of all possible "answers," that is to say, combinations of the variables which are consistent with *all* the rules. The example above has one rule, three variables and four answers.

A machine dealing with problems of this nature and known as the Kalin-Burkhart Logical Truth Calculator has been constructed,[2] but is known to the authors only by name.

Fig. 1. General view of the Logical Computor

The main computor is on the right, while on the left is the rack with the connective boxes. In front of the rack is the counting unit

* Ferranti, Ltd., Edinburgh.

The Ferranti Logical Computor

The computor has been built from war surplus relays and components which were readily available. No attempt has been made to produce a highly refined circuit, and improvements may be apparent to the reader.

The general appearance of the machine is apparent from Fig. 1, in which there will be seen seven pairs of signal lamps at the top of the rear panel of the computor; each pair represents one of the variables. The upper lamp of each pair is green in colour and lights when the variable is in the "affirmative" state; the lower (red) lamp indicates the "negative" state. These lamps permit of the reading of the combination of the variables which is being dealt with at any instant; for example, if the lamps show the sequence (reading from left to right) red-red-green-green-red-green-red, then the combination set up in the machine, using the previous notation, is A̸B̸CDE̸FG̸.

When the first variable is in the state A the six paralleled output sockets beneath the indicating lamps for that variable are "live" (this term will be used throughout to denote direct connexion to the +24 volt supply) and when it is in the state A̸ the sockets are connected to earth, and similarly with B, C, D, E, F and G. When the machine runs through a problem, every combination of the seven variables is set up in a prearranged sequence. There are 128 such combinations.

The middle area of the machine is for the insertion of the rule boxes required for the particular problem. These boxes may be seen in the rack on the left of Fig. 1; a number of them are shown plugged into the computor in Fig. 7. Each box contains a Siemens high speed relay and picks up the supply voltage from a pair of sockets into which it is plugged. These rule or connective boxes are all so designed that when the variables connected to the input terminals satisfy a certain logical relation, a green signal lamp on the box lights and the relay makes the output terminals of the box live; when, on the other hand, the required relation is not satisfied, the output terminals are earthed and the lamp does not light. The six types of connective box used on the machine are NOT, AND, OR, OR ELSE, IF AND ONLY IF, IF THEN.

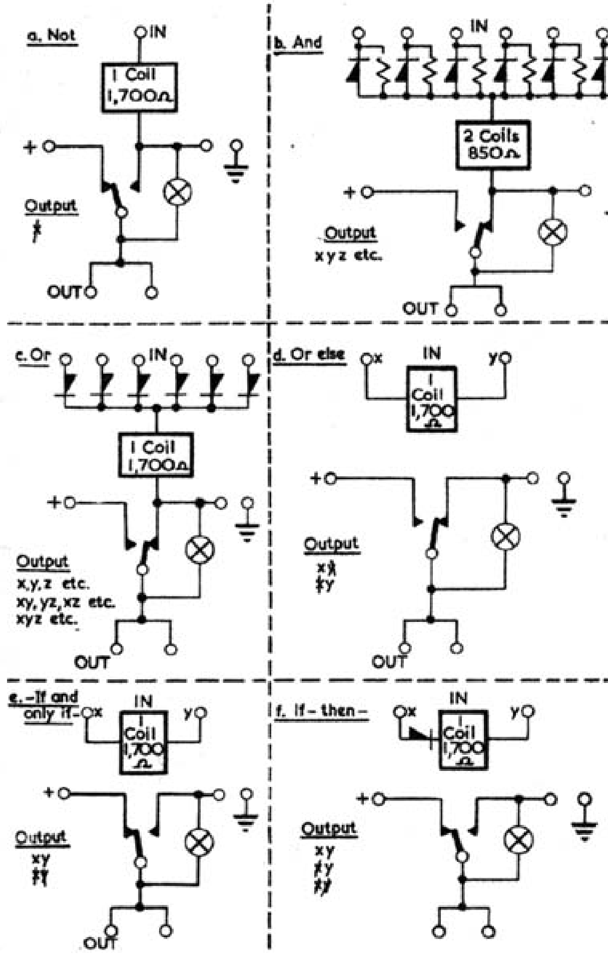


Fig. 2. Connective boxes for the Logical Computor

Connective Boxes

The operation of these boxes is described by denoting the inputs to them as x, y, z, where each of these letters may denote any one of the variables A, B, C, D, E, F, G, or may denote the output of a preceding box. In every case x will denote the variable in its affirmative state and will thus correspond to a live lead; x̸ will conversely correspond to an earthed lead.

Reference should be made to the diagrams of Fig. 2 for details of the circuits used to perform the functions indicated below.

(a) NOT This box has a single input. If this input is earthed (i.e. in the negative state), the condition NOT is *satisfied* and the relay makes the output sockets *live* and lights the green lamp on the box. Conversely, if the input is live, the condition NOT is *unsatisfied* and the relay connects the output sockets to *earth* and extinguishes the lamp.

This may be expressed briefly by saying that this condition is satisfied by x̸.

(b) AND This box may have any number of inputs up to six. The condition to be satisfied is that *all* inputs must be in the affirmative state; if there are three inputs, the condition is satisfied by xyz only.

This is brought about by the circuit shown, the inputs being connected through selenium rectifiers shunted by 1.5kΩ resistors; the cathodes of the rectifiers are connected to the input terminals. When any input is earthed, the resistance to earth in parallel with the relay is so low that even if all the others are live they are unable to develop sufficient voltage across the relay to operate it. Input sockets not required need not be connected.

(c) OR This box may also have any number of inputs up to six and the required condition is that at least one input must be in the affirmative state. Here the anodes of the rectifiers are connected to the input terminals and any one live input generates a sufficiently large potential difference across the relay to operate it. With three inputs the condition is satisfied by any one of xyz, xyz̸, xy̸z, x̸yz, xy̸z̸, x̸yz̸, x̸y̸z.

(d) OR ELSE This box has two inputs and the condition is that they shall be in opposite states. The relay operates and makes the output live when one or other of the inputs is live and the other earthed. To operate this box both input terminals must be connected. The condition is satisfied by xy̸ or x̸y.

(e) IF AND ONLY IF This box has two inputs and requires for satisfaction of the condition that both inputs shall be in the same state (i.e. both affirmative or both negative). The relay bobbin is connected between the input terminals as in the OR ELSE box, but the output is live when the relay is not operated i.e. when the inputs are either both live or both earthed. The condition is satisfied by xy or x̸y̸.

(f) IF—THEN This box has two inputs which correspond to the relation "if x, then y"; it is to be noted that this differs from "if y, then x" so that the order of the input sockets is important. The condition is satisfied by xy, x̸y, x̸y̸ but not by xy̸. The circuit is similar to that for "IF AND ONLY IF" with an additional rectifier which prevents the relay from operating when the inputs are of the form x̸y.

Control Panel

The front section of the machine carries the controlling switches and the lamps which indicate in what state the machine is. There are three such states:—

(1) RESET This is the condition of the computor when ready to start scanning. When switched on, the machine rapidly sets itself in this condition. At any time on pressing the RESET button the machine will return to this condition as it also does on completing a scanning cycle of the 128 combinations.

(2) SCANNING When a problem has been set up the machine is started searching for an answer by pressing the THINK button. If the AUTO-MANUAL switch is at AUTO the machine scans the combinations under the control of the self-oscillating timing relay.

(3) ANSWER On reaching an answer the machine stops; the answer may be read from the lamps showing the states of the variables. After the answer has been noted the machine is started scanning again by pressing the THINK button.

To operate the answer mechanism the outputs of the rules in use are plugged into the sockets at the front of the machine. These go to an AND box which operates the answer relay and lights two large signal lamps when a combination of variables consistent with the rules is found. The machine may also be made to find combinations inconsistent with the rules by switching the CON-INCON switch to INCON. This converts the final AND box into a NOT AND box.

Mechanism of Computor

The arrangement for setting the 128 combinations of the 7 variables consists of 7 relays, one for each variable, a uniselector which opens and closes them in a prearranged sequence, and auxiliary relays. The sequence in which the variables are altered is that known as binary cyclic permuting code in which only one change is made from one combination to the next. For the first four variables a complete sequence is:—

Combination	Changes	Combination	Changes
ABCD	mA	ABC̸D̸	mA
A̸BCD	mB	A̸BC̸D̸	mB
A̸B̸CD	bA	A̸B̸C̸D̸	bA
AB̸CD	mC	AB̸C̸D̸	bC
AB̸C̸D	mA	AB̸CD̸	mA
A̸B̸C̸D	bB	A̸B̸CD̸	bB
A̸BC̸D	bA	A̸BCD̸	bA
ABC̸D	mD	ABCD̸	bD

(mA denotes "A relay makes," bC denotes "C relay breaks" etc.)

and this may readily be extended to seven digits. Using this arrangement the seven relays are operated or released on the pattern shown in the change column of the table. The circuit shown in Fig. 4 is used for the relays. When a relay is to be made the "m" line is earthed and when it is to be released the "b" line is earthed.

A four bank uniselector is used with 50 contacts on each bank. This gives 200 positions of which only 32 on each bank are used for changing the variable relays. The uniselector is stepped round by a timing relay circuit at about 3 steps per second. After 32 contacts on a bank have been used it steps rapidly over the remaining 18 changing the α or β relays (Fig. 4) which switch the circuit through the four banks in sequence.

The complete circuit for the computor is shown in Fig. 4. The other relays perform functions connected with starting, finding an answer and resetting.

Counting Unit

As an additional facility a box is provided to count the number of the seven variables which are in the states A, B, etc. (i.e., having outputs live, as opposed to A̸, B̸, etc. which have outputs earthed). This is done by a digital method using relay switches to transfer the input from one line to the next (Fig. 3). The box can be seen in Figs. 1 and 7 next to the main computor; it has a row of 8 lamps numbered 0 to 7, and lights the lamp corresponding to the number to be indicated. Each lamp has a push-button which when depressed makes the output socket of the box live whenever the lamp lights; one or more of these

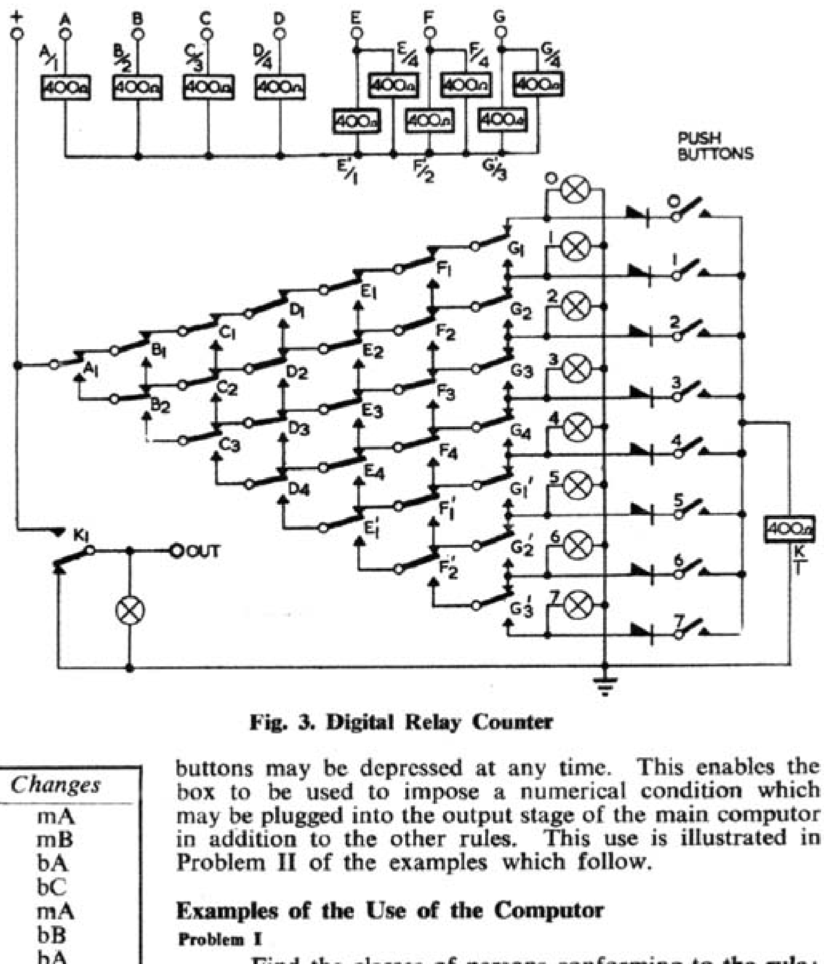


Fig. 3. Digital Relay Counter

buttons may be depressed at any time. This enables the box to be used to impose a numerical condition which may be plugged into the output stage of the main computor in addition to the other rules. This use is illustrated in Problem II of the examples which follow.

Examples of the Use of the Computor

Problem I

Find the classes of persons conforming to the rule: Only members or their guests may play over the Blankshire Golf Club's course.

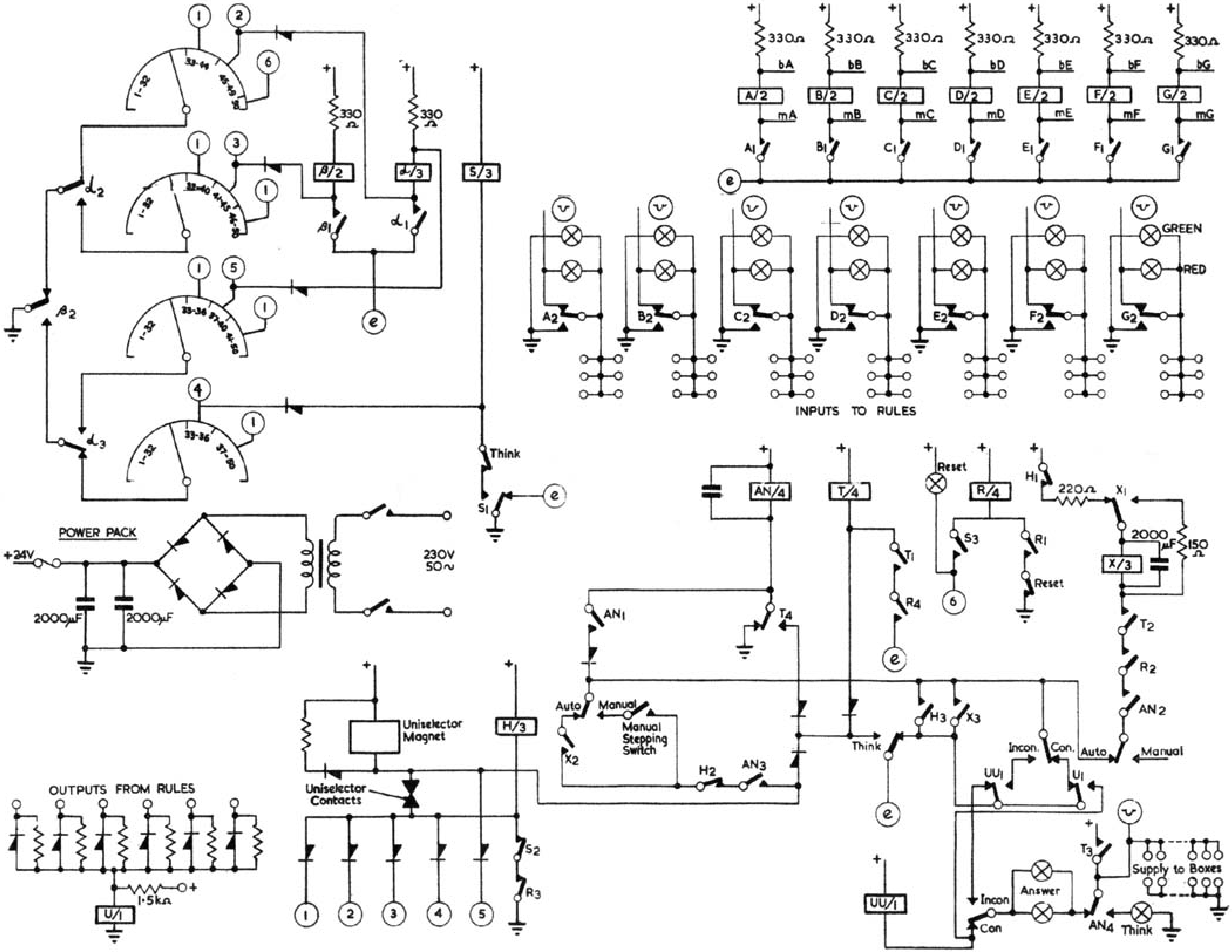


Fig. 4. General Circuit Diagram of the Logical Computor

This problem was explained earlier. The connexions required to set it up in the machine are shown in Fig. 5 and the machine will find the four solutions in succession.

Problem II

I am going to University and have to choose what subjects I will do in my final year at school. The subjects I can choose are Maths, History, Science, English, Latin, German and French.

English is compulsory and I may choose up to five other subjects. If I take Science I must take Maths. If I take Latin then I cannot take German as the timetable clashes. I do not want to take History. For entrance to the course at the University I need to have Science and French. How many ways may I choose the subjects?

The connexions required to set this problem up are shown in Fig. 6(a). There are only two solutions.

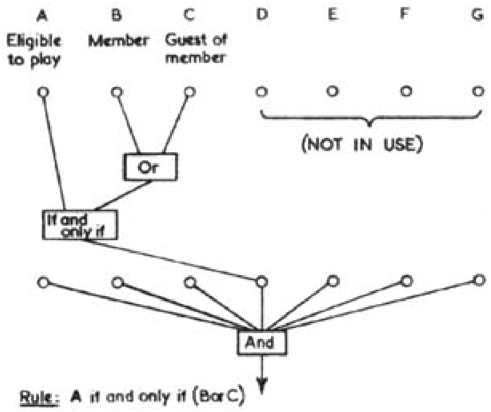


Fig. 5. (Above). Setting up problem with a single rule

Fig. 6. (Right) (a)

Rules (i) *Not History.* (ii) *If Science, then Maths.* (iii) *If Latin, then not German.* (iv) *Science and French.* (v) *5 or 6 subjects.* (vi) *English.*

(b) See text for explanation of rules

Problem III

It is known that salesmen always tell the truth and engineers always tell lies.

B and E are salesmen. C states that D is an engineer. A declares that B affirms that C asserts that D says that E insists that F denies that G is a salesman.

If A is an engineer, how many engineers are there?

This problem is set up by taking A and A̸ to represent "A is an engineer" and "A is a salesman (i.e. not an engineer)" respectively, and similarly for the other variables. The connexions then required are shown in Fig. 6(b). It is not immediately obvious how these are deduced from the problem. The basic guide is that if "A states X," this is equivalent to "A is an engineer *or else* X is true"; repeated application of this formula is *all* that is necessary for setting up the two rather involved rules. It is to be noted, therefore, that the operator does not have to grasp the implications of either of these rules *as a whole*, so that the difficulty of setting up the machine is very much less than the difficulty of solving the problem without the machine.

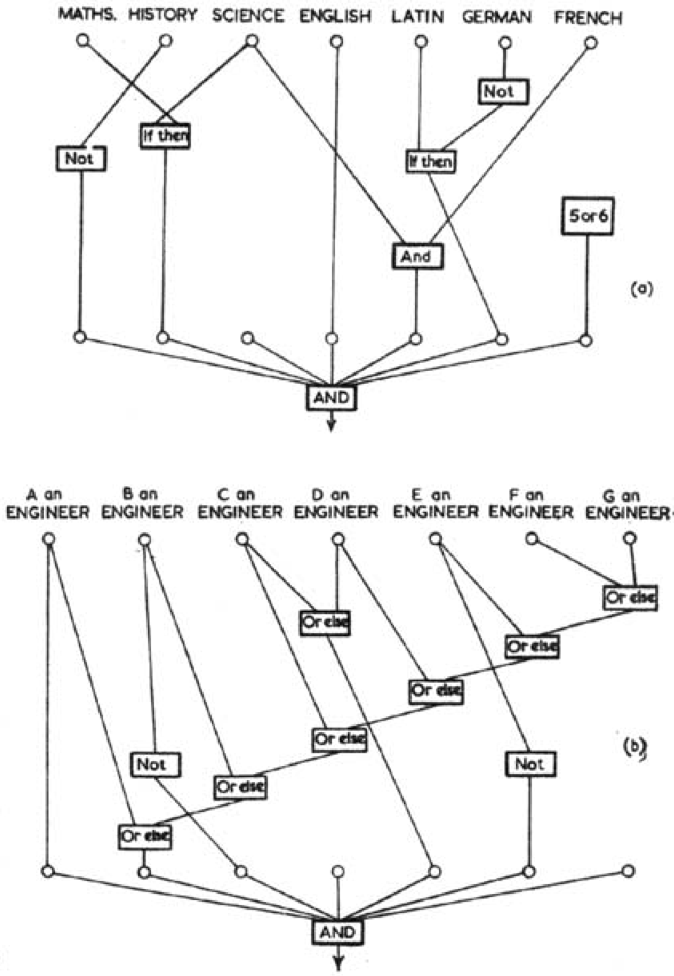
There are four solutions (AB̸C̸DF̸E̸G, AB̸C̸DE̸FG̸, AB̸CD̸F̸E̸G, AB̸CD̸E̸FG̸), and inspection shows that in every case the number of engineers is three.

The appearance of the computor with a problem set up is shown in Fig. 7.

Possible Extension of a Computor of this Type

The reader who has followed thus far may be inclined to question the usefulness of a computor of this type. The demonstration model described is of course severely limited in scope, but it is possible to envisage elaborations of it which would greatly increase its usefulness without adding to its complexity. These will be considered under three headings

(i) Use of the present Computor as control mechanism for a numerical calculating machine.
(ii) Design of a Computor on the same principles as the present one, but of greater capacity.
(iii) Design of a Computor on the same principles as the present one and of greater capacity, but incorporating more advanced ideas in order to reduce greatly the solution time for problems of a certain class.



A Logical Computor as Control Mechanism for Numerical Calculations

This use may best be described by means of an example of the type of problem which may be attacked:

What is the probability of throwing at least two heads when tossing four coins, of which two are normal, and two biased, one in such a way as to throw tails twice as often as heads and the other in such a way as to throw tails nine time out of ten?

This problem is mathematically equivalent to obtaining from the expansion of $(\frac{1}{2}H + \frac{1}{2}T)^2 (\frac{1}{3}H + \frac{2}{3}T) (1/10H + 9/10T)$ the sum of the numerical coefficients of all terms for which the index of H is equal to or greater than 2. There are therefore two distinct steps in obtaining the result, namely, selection of terms and operation upon the coefficients of selected terms. A Logical Computor is ideally suited to perform the first step and may therefore be used to control a numerical computor performing the second.

The selection process is done thus: suppose that each coin is represented by a "variable" of the Logical Computor so that A, B, C, D represent "heads" and $\not{A}$, $\not{B}$, $\not{C}$, $\not{D}$, "tails." Then the condition requiring two or more heads is evidently represented by the single rule:

(A and B) or (A and C) or (A and D) or (B and C) or (B and D) or (C and D) which may be set up directly in a manner analogous to that described for previous problems, and will cause the instrument to signal an "answer" whenever a combination is set up which includes two or more "heads".

If now the Logical Computor performs its usual routine of scanning sequentially the possible combinations (sixteen in all) of "heads" and "tails", it is obviously a simple matter to arrange that for each combination set up, four and only four factors are fed into a multiplier, these factors being as follows:—

First factor	$\frac{1}{2}$	if	the combination	contains	A
	$\frac{1}{2}$	"	" "	"	$\not{A}$
Second factor	$\frac{1}{2}$	"	" "	"	B
	$\frac{1}{2}$	"	" "	"	$\not{B}$
Third factor	$\frac{1}{3}$	"	" "	"	C
	$\frac{2}{3}$	"	" "	"	$\not{C}$
Fourth factor	$\frac{1}{10}$	"	" "	"	D
	$\frac{9}{10}$	"	" "	"	$\not{D}$

The relation of these factors to the problem will be evident; to solve the problem it is only necessary to run through the sixteen combinations and to add the outputs given by the multiplier for *all combinations indicated by the Logical Computor to be "answers,"* and for no others. The routine therefore proceeds thus:

Combination set up by Logical Computor	Factors fed into Multiplier	Product	Conditional Control (Arrow denotes answer)	Running Total
ABCD	$\frac{1}{2} . \frac{1}{2} . \frac{1}{3} . \frac{1}{10}$	$\frac{1}{120}$	→	$\frac{1}{120}$
$\not{A}BCD$	1 . 1 . 1 . 1*	1	→	2
$\not{A}\not{B}CD$	1 . 1 . 1 . 1	1	→	3
$A\not{B}CD$	1 . 1 . 1 . 1	1	→	4
$A\not{B}\not{C}D$	1 . 1 . 2 . 1	2	→	6
$\not{A}\not{B}\not{C}D$	1 . 1 . 2 . 1	2		
$\not{A}B\not{C}D$	1 . 1 . 2 . 1	2	→	8
$AB\not{C}D$	1 . 1 . 2 . 1	2	→	10
$AB\not{C}\not{D}$	1 . 1 . 2 . 9	18	→	28
$\not{A}B\not{C}\not{D}$	1 . 1 . 2 . 9	18		
$\not{A}\not{B}\not{C}\not{D}$	1 . 1 . 2 . 9	18		
$A\not{B}\not{C}\not{D}$	1 . 1 . 2 . 9	18		
$A\not{B}C\not{D}$	1 . 1 . 1 . 9	9	→	37
$\not{A}\not{B}C\not{D}$	1 . 1 . 1 . 9	9		
$\not{A}BC\not{D}$	1 . 1 . 1 . 9	9	→	46
$ABC\not{D}$	1 . 1 . 1 . 9	9	→	55

(* Denominators of all fractions remain constant and are omitted from this point onwards)

The required probability is thus 55/120. Although the description appears complex, the design of the numerical computor is not difficult, and the authors have been able to design a simple relay computor (using about 20 relays only) which could be controlled by the Logical Computor described and which would perform the above calculation with probabilities chosen from the discrete values 0, $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$, $\frac{7}{8}$, 1, for any number of variables up to seven, displaying the final result as an exact vulgar fraction.

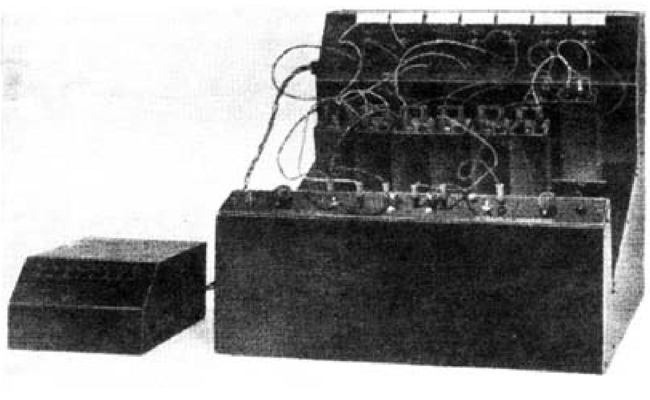


Fig. 7. View of Computor set up for Problem III

Leads come from the variables to the connective boxes and from the outputs of the connective boxes to the sockets in the control panel, the latter sockets being the inputs of the "AND" box which detects answers.

A Logical Computor of Greater Capacity

The desirability of increasing the capacity of the present Computor is evident. There is no difficulty in increasing the capacity by a factor of about 2 or 3, since it is possible to use an identical system with valves replacing the relays. This enables the "scanning" of possible solutions to be done at a much greater rate; a thousand times faster would seem not unduly difficult and since $1000 = 2^{10}$ approximately, this would enable the possible combinations of 17 variables to be scanned in the same time at present used for 7.

Since with a larger capacity the number of "answers" found will tend to be larger, it would be desirable to provide for the machine to record answers automatically and not, as in the case of the present machine, to stop on finding an answer. This presents no technical difficulty.

It is clear, however, that the doubling of the length of the operating cycle which accompanies each increase of the capacity by a single variable will ultimately set a limit to the capacity of such a machine and that a machine to deal with say, 50 variables would be impracticable.

This difficulty is a fundamental one, and can only be resolved by a new approach to the problem. A possible method of attack is described in the next section.

A Logical Computor involving more Advanced Principles

Let it be supposed for the present that the problem consists not of finding *all* "answers" to a logical problem, but only *one*. For definiteness, consider the problem of which the "rules" are

If B, then C.
A If and only if D.
A or else B.

The reader should attempt to obtain mentally a single "answer" to this problem. (There are, in fact three, namely $A\not{B}CD$, $A\not{B}\not{C}D$, $\not{A}BC\not{D}$). He should then consider the mental process by which he arrived at the solution; he will probably find it to be roughly that of *assuming a trial solution* and *modifying it to remove features inconsistent with the rules.*

It was suggested to the authors by Mr. M. K. Taylor that a logical computor might be made to operate on this principle, and they have, in fact, been able to give a simple demonstration of this technique using special "sensitive variable" units in conjunction with the normal connective boxes of the basic Logical Computor. The "sensitive variable" units have the following properties:—

(a) Each possesses a number of output sockets and a number of input sockets (known as "feedback" sockets for a reason to be explained later).

(b) Each has two basic states, corresponding to the affirmative and negative states of the variables of the original Computor. In the affirmative state a green lamp upon the unit is lit, and the output sockets are connected to +24 V; in the negative state a red lamp is lit and the output sockets are connected to earth.

(c) Either state is normally stable; that is, the unit remains in whatever state it happens to be. An exception to this rule is that *when any feedback socket of the variable unit is connected to earth*, this has the effect of making the unit "hunt" between its two basic states, changing its state about twice per second.

Each sensitive variable unit contains only a single relay; the circuit is shown in Fig. 8.

Referring now to our problem, suppose the problem is set up as in Fig. 9(a) with these sensitive variable units linked by connective boxes of the type previously described in accordance with the given rules. The configuration of the variable units will then be an "answer" if and only if all three rules are satisfied, i.e., if the leads marked 1, 2, 3 are all live. If it is not an anwer, one or other of the leads 1, 2, 3 will be connected to earth.

Now let additional connexions be made as in Fig. 9(b) where the dotted lines represent connexions from the leads 1, 2, 3 to the feedback sockets of the variable units made on the principle that *each rule is "fed back" to all variables in it*, i.e., "A or else B" is connected back to A and B, and so on. This gives rise to a network which has the following desirable properties.

(a) If the configuration set up on the variable units is an "answer" then the network is stable and remains in that configuration.

(b) If the configuration set up is not an answer, the network changes its configuration.

(c) The change in configuration occurs only in those parts of the configuration which violate the rules.

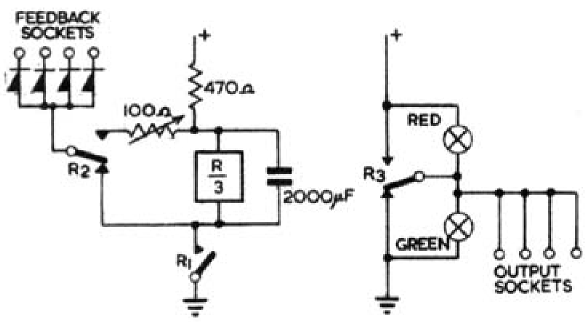


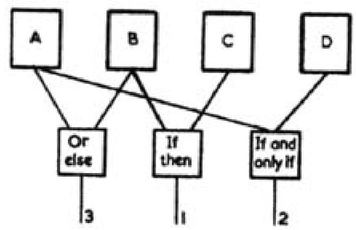
Fig. 8. (Above). Circuit diagram of sensitive variable unit
Fig. 9. (Right). Feedback connexion principle

Referring again to the example and to Fig. 9, suppose the initial configuration is $\not{A}\not{B}\not{C}\not{D}$. Then the first rule (if B, then C) is satisfied, so point 1 is live; the second rule (A if and only if D) is satisfied, so point 2 is live; but the third rule (A or else B) is unsatisfied, so point 3 is connected to earth. This places an earth on the feedback sockets of variables A and B, which begin to hunt. Suppose A changes first, giving A$\not{B}\not{C}\not{D}$. This satisfies rule 3, so the earth on point 3 disappears; but at the same time an earth appears at point 2, since rule 2 is not now satisfied; A and D now begin to hunt. Suppose D changes first, giving A$\not{B}\not{C}$D. This satisfies all the rules, so points 1, 2, 3 are all live and no further change takes place.

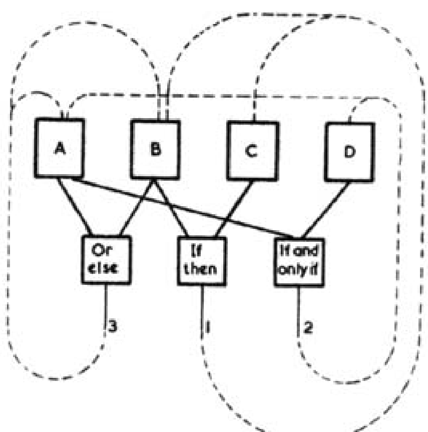
It is seen that this type of system may give a rapid convergence towards a solution however many variables are involved. There are, however, certain basic difficulties which arise, which will be briefly outlined.

The first is concerned with the general stability of th network set up; as would be imagined, the introduction o a large number of feedback loops brings with it the pos ability of continued oscillations of a *periodic* type whicl represents a process of "arguing in a circle." To illus trate this, reference is again made to the above example suppose that the sensitive variable A always hunts mor rapidly than the others. Then as before $\not{A}\not{B}\not{C}\not{D}$ wil require that A and B hunt; A, being the more rapid, wil change first, giving A$\not{B}\not{C}\not{D}$; this will require that A and D hunt; but this time, A being the more rapid, will chang again, restoring $\not{A}\not{B}\not{C}\not{D}$; this cycle will repeat indefinitely

It might appear that this type of instability could b avoided by making the hunting rates of all variables equal but this is not so; referring again to the example, $\not{A}\not{B}\not{C}\not{D}$ would change to AB$\not{C}\not{D}$ (A and B changing simultaneously); this latter combination violates all three rules and all variables would change, giving $\not{A}$BCD, which violates rules 2 and 3. This in turn would give ABC$\not{D}$, whicl still violates the same rules, causing a return to $\not{A}$BCD and the latter two configurations would alternate indefinitely. In practice it is found that even if the huntinք rates are only approximately equal when measured independently, there will be a tendency when they are coupled into a network for "pulling in" to occur so thai all variables hunt at a common frequency.



(a)



(b)

The complete solution of this problem is outside the scope of this article, but it consists in brief of introducing "random" disturbances which momentarily "clamp" one variable (i.e., remove from it all feedback connexions) and thus effectively break up any periodic oscillation.

A computor of this type will in general tend to find an "answer" which differs as little as possible from the initial configuration set up. If all answers are required, it is necessary that all initial configurations be set up, and this type of computor reduces to the simple type considered earlier. It is however possible in general to select from the set of all configurations a subset of any given length which has the property that any configuration lies tolerably close to one member of the subset. (The property is more accurately expressed geometrically; the n variables, regarded as numbers which may have the value 0 or 1, are the vertices of an " n-dimensional cube " in n-dimensional space; the required subset is a group of approximately equidistant vertices of this figure). For example, with four variables, every configuration lies within one change from at least one member of the subset $\not{A}\not{B}\not{C}\not{D}$, $A\not{B}\not{C}D$, $ABC\not{D}$, $\not{A}BCD$, but the members of the subset all differ by at least two changes. Hence, if these be used as initial configurations in four trials with a computor of the feedback type, the probability of obtaining several *different* answers is much enhanced. The operations of recording the answer found and setting up a new initial configuration may, of course, be made automatic in cases where a large number of variables are used.

The properties of such a feedback computor may therefore be summarized thus:

(i) The time taken to obtain a single " answer " is much reduced.

(ii) If the number of " answers " is small and the " answers " are isolated (i.e., all differ appreciably) then steps may be taken to increase the chance of obtaining *all* answers early in the operating cycle. This is a particularly useful feature if the number of answers is known.

(iii) In general, if *all* answers are required, every initial configuration must be set up; in this case there is no advantage to be gained from the feedback method.

Conclusion

The authors feel that the possibilities of small specialized computors of this type are tending to be overlooked and that intensive investigation of their capabilities should be made. It is hoped that this note may stimulate further work in this direction.

The authors wish to thank Messrs. Ferranti, Ltd., for permission to publish this note and to acknowledge the numerous valuable suggestions made by Mr. M. K. Taylor, Chief Research Engineer, in connexion with the work therein.

BIBLIOGRAPHY

[1] Boole, G. The Mathematical Analysis of Logic (Macmillan, Barclay and Macmillan, Cambridge, 1847).
[2] Berkeley, E. C. Giant Brains, or Machines that Think (Wiley, New York, 1949).