Reprinted from
JOURNAL OF ELECTRONICS AND CONTROL, Vol. 6, No. 2, p. 168, February 1959

K.T. VASWANI.

Ph.D. THESIS
APRIL, 1960.

The Optimization of Logical Goal-seeking Procedures †

By A. Porter‡
College of Engineering, University of Saskatchewan, Canada

and P. K. T. Vaswani
Imperial College of Science and Technology, London

[Received December 16, 1958]

Abstract

The property of decision-taking is basic in all types of control systems, and the goal-seeking characteristic of such systems can be considered as a direct consequence of this decision-taking capability. The difficulty of optimizing goal-seeking procedures in systems in which the rules, or constraints, are based on two-valued logical statements is discussed. In the present study it is shown that if the variables in a particular system, and the corresponding rules of behaviour, can be expressed in terms of a many-valued logic, a function, called the measure function, can be defined which provides a crude method of optimizing the goal-seeking process. The approach is essentially empirical. A high-speed computer is used to study selected 16-valued logical problems. The main object of the paper is to stimulate interest amongst control specialists in the possibility of applying many-valued logics in the control of complex processes.

§ 1. Introduction

In the control of certain classes of complex processes the criteria upon which optimum performance depends may frequently be stated in terms of logical relationships between the significant process variables. These relationships are described as rules or constraints. For example, in air traffic control systems some of the rules of operation can be expressed in terms of simple two-valued logical statements, and it is the task of the controller, man or machine, to ensure that these logical constraints are satisfied at all times. On the other hand, other processes may involve many-valued variables and the corresponding rules of operation will involve many-valued logical statements. It is probable in the future that a degree of mechanization will be introduced into such processes and this will inevitably lead to problems involving the solution of logical inference.

The procedure involved in determining the state of a system, which satisfies a given set of logical rules, may be regarded as a goal-seeking process, and the consideration that the achievement of the goal usually involves the taking of a decision to initiate some particular action has

† Communicated by Professor Porter.
‡ Lately at Imperial College of Science and Technology, London.

resulted in the term ' decision-element ' being used to describe automatic elemental logical connectives. It should be noted that all control systems are essentially goal-seeking, and all incorporate manual or automatic decision-taking elements. For example, a simple regulator or servo-mechanism incorporates a single decision-element which determines and initiates the action necessary for the behaviour of the control system to approach a desired state, or goal, in an optimum manner. In such systems, the decision-element is a simple comparator. On the other hand, in multi-variable process controls it is frequently impossible to define the overall control problem in terms of such simple decision-elements because the operational decisions usually involve interactions between variables.

It is worth noting that the enormous power and potential of the high-speed digital computer is due essentially to the introduction of the conditional transfer (or jump) instruction into the machine order code; this involves a logical decision-element of a simple kind. This facility appears to be adequate for all computational purposes although it may not be adequate to deal with the fairly complex logical statements inherent in many data-processing systems.

An example of a problem in which complex logical decisions may be involved is the control of an electrical power distribution system, in which it is necessary to mechanize the processes of load-shedding and load-distribution in order to optimize performance—this problem would be further complicated if nuclear power stations were incorporated into the power distribution system. Moreover, all traffic control problems and most problems involving time-tabling can be stated in terms of logical connectives, special cases being the optimization of certain production-line processes, and the automatic servicing of complex electronic equipment, perhaps involving large numbers of components.

This paper deals with the general problem of handling logical goal-seeking processes with special reference to optimizing procedures. Although, in the present state of development, the approach is essentially empirical, it is felt that the importance of the subject justifies an early statement of the basic problems involved.

§ 2. Statement of 2-valued Logical Problems

2.1. *Logical Connectives*

We adopt the notation in which variables, assumed in the first place to be two-valued, are denoted by the letters A, B, C, etc., and the value 1 is assigned to a variable which is true and the value 0 is assigned to a variable which is false—the assigned number is referred to as the ' truth-value ' of the variable. We define logical connectives as functions which operate on the truth-values of a set of variables, i.e. F (A, B, . . . X) to produce a resultant set of truth-values. The most common two-valued logical connectives are listed below, and the corresponding truth-tables are given in fig. 1.

Table 1

Connective	Written	Symbol
Negation	Not A	$\sim$ A or $\bar{A}$
Conjunction	A and B	A . B
Disjunction	A or B	A v B
Implication	If A then B	A$\rightarrow$ B
Equivalence	A is equivalent to B	A $\equiv$ B

Fig. 1

Variables		Connectives				
A	B	$\sim$A	A . B	A v B	A$\rightarrow$B	A $\equiv$ B
1	1	0	1	1	1	1
1	0	0	0	1	0	0
0	1	1	0	1	1	0
0	0	1	0	0	1	1

A statement such as " A implies (B or C) " is described as a ' rule ', and this particular rule is written symbolically as A$\rightarrow$(B v C). The corresponding truth-table for this rule is given in fig. 2.

Fig. 2

1	2	3	4	5	6	7	8
A	B	C	A	$\longrightarrow$	(B	v	C)
1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	0
1	0	1	1	1	0	1	1
1	0	0	1	0	0	0	0
0	1	1	0	1	1	1	1
0	1	0	0	1	1	1	0
0	0	1	0	1	0	1	1
0	0	0	0	1	0	0	0

Consider the following elementary problem which can be readily stated in terms of logical connectives:

" A manufacturer must decide which items to produce in order to maximize his profit. Suppose these items are designated by the letters A, B, C, D, and E. If he chooses item A, he must also produce items B, or C, because these items involve similar raw materials. However, it is uneconomical to manufacture both items B and C simultaneously. Further, if item D is chosen, it is convenient for item B but not item C to be produced. It is also well known that if item E is produced then D should not be produced. Determine the desirable combinations of items which can be manufactured."

This problem can be stated symbolically as follows:

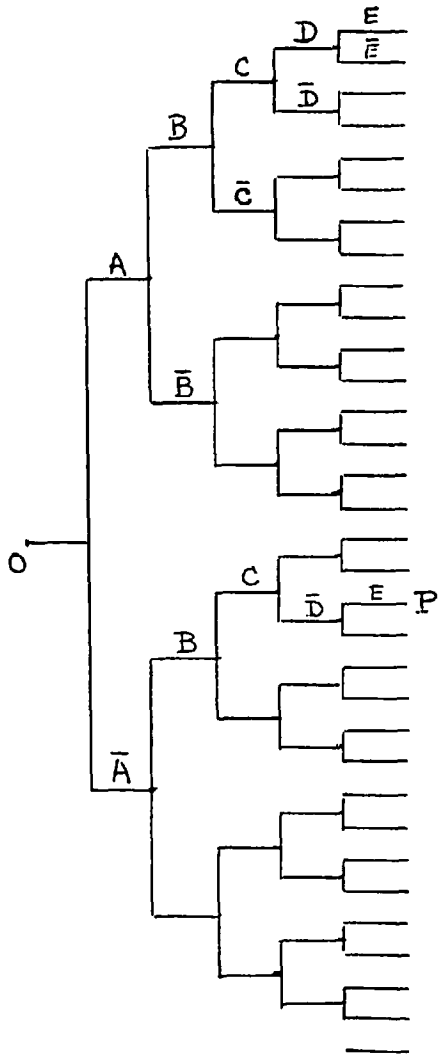
$$[A\rightarrow(B \text{ v } C)] \,.\, [D\rightarrow(B \,.\, \sim C)] \,.\, \sim[D \,.\, E] \,.\, \sim[B \,.\, C]$$

and it can readily be shown, using the truth-table method†, that there are 11 non-trivial permissible combinations of items some of which are: A . B . D; A . B . E; A . C . E, etc.

2.2. *Interpretation of Logical Connectives as Relay Switching Networks*

Perhaps the most straightforward way of setting up a model for logical goal-seeking processes is in terms of relays or the equivalent diode and

Fig. 3



Switching tree.

† See, for example, Basson and O'Connor 1957, *Introduction to Symbolic Logic* (University Tutorial Press).

magnetic core circuits which can be used for line selection. The representation of the fundamental connectives in terms of simple switching circuits is trivial. However, it may be useful in visualizing the goal-seeking nature of logical problems to consider the problem in terms of seeking unique paths through a relay tree. Given n relays we can set up a relay tree capable of selecting any one of 2^n lines—a 5-variable switching tree is shown in fig. 3. Each of the terminal states shown can be reached by a unique path starting at point O. Given a set of logical rules for which a single solution exists it is clear from the diagram that we seek the ' line ' whose interconnections to point O satisfies the given rules. Generally, the rules involve redundant information and we can regard the solution-seeking procedure as a means of eliminating the redundancies and reducing the statement of the rules to the elemental logical connectives AND and NOT. In fig. 3 the required ' path ' to satisfy a particular set of rules is shown, as the path defined by $O\bar{A}BC\bar{D}E$. Having established this unique path we have effectively reduced the logical rules to their most elementary form, and incidentally, we have achieved the required goal which in this case can be regarded as a selected line. We will investigate how the process of reaching the goal can be achieved in the least number of steps.

2.3. *The ' Problem-space '*

Although the relay-tree system introduced above gives a physical picture of the nature of two-valued logical problem-solving procedures it is, nevertheless, of limited value in handling these problems. It is convenient to consider such problems in terms of a so-called ' problem-space '.

If a process involves n independent two-valued variables we can represent the 2^n possible combinations of variables as the vertices of an n-dimensional hypercube. The edges of the hypercube represent all possible single transitions between states of the system. In the case of $n=4$ the corresponding state transition diagram is shown in fig. 4.

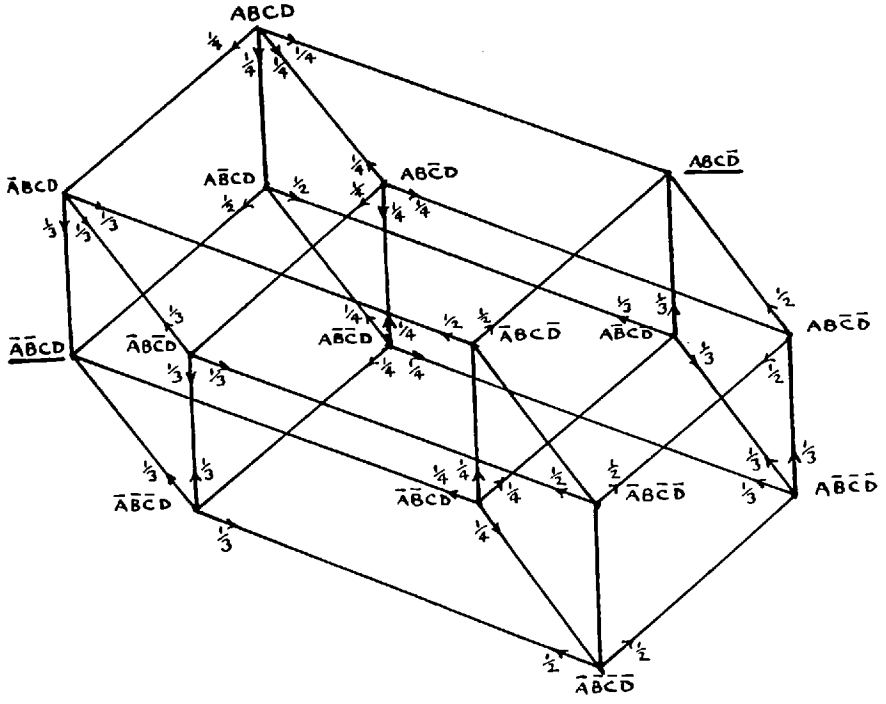
We define the ' space ' occupied by the hypercube as the ' problem-space '. The goal-seeking procedure may be considered in the most elementary form as a sequential scanning process in which each state (state is here identified with a particular combination of variables) is examined to determine whether or not it satisfies the given logical rules. In the scanning process only one state—i.e. point in the problem-space—can be examined at a particular instant.

It is convenient to introduce the concept of ' distance to go ' or ' error-measure function ' in considering the relative ' positions ' of a particular point in the problem-space, corresponding to a given state, and the position of the point corresponding to the desired state; the latter is the ' goal '.

2.4. *Problem-solving Procedures*

In the case of two-valued logical problems involving no more than eight or nine variables it is possible to obtain the required solutions analytically by determining the corresponding truth-table of a particular set of rules or by using a punched-card procedure in which each possible state, i.e. combination of variables, is represented by a punched card and by a straightforward card-sorting procedure the cards not satisfying a particular rule are discarded progressively as each rule is considered in turn. The cards remaining after all rules have been applied correspond to the desired states.

Fig. 4



Equi-probable transitions.

Automatic methods have also been devised for solving two-valued logical problems and the machine built by McCallum and Smith (1950) is particularly noteworthy. In their early machine, which handled seven variables and incorporated a large variety of logical connectives, the machine scanned automatically through all possible combinations of the variables and stopped automatically when a desired state was reached. Whilst applicable to problems in which a limited number of variables are involved, 20 variables involve one million combinations, the scanning method is clearly impracticable if more than 30 variables are involved, even though we utilize high-speed circuits.

In the second machine McCallum and Smith (1951) introduced a new goal-seeking procedure. Assuming the machine starts in an arbitrary state of the variables it carries out automatically a single transition scan in such a way that in the transition from one state to the next the only variables which can be changed are those involved in one or more broken rules. Since only one variable is changed in passing from one state to the next a random selection of the variables involved in the broken rules is made for each transition. The scanning procedure now corresponds to a discrete Markoff process, i.e. a sequence of events in which the probability of a particular outcome of the nth event is fully determined by the outcome of the $(n\text{-}1)$th event. This machine has been described as a logical feedback machine since ' errors ' in an existing state are used to determine the next state in the process of achieving the required goal. It can be shown that this machine is only superior to the ' scanning-machine ' in situations where the rules involved only include a few variables and in which each variable is only involved in a few rules.

We have examined a modified ' feedback process ', also suggested by McCallum and Smith, in which the variable changed in the transition from one state to the next is chosen on the basis of a weighted transition probability. For example, the weight given to a particular variable is made proportional to the number of broken rules in which the variable appears. Again, the goal-seeking procedure is essentially a discrete Markoff process.

Consider a system in which the desired state, or states, is required to satisfy the following rules:

(i) $$A \rightarrow C,$$

(ii) $$(\overline{B} \,.\, C) \equiv D,$$

(iii) $$\overline{(A \equiv D)} \quad \text{or} \quad \sim(A \equiv D),$$

which can be written as a composite rule as follows:

$$(A \rightarrow C) \,.\, [(\overline{B} \,.\, C) \equiv D] \,.\, \sim(A \equiv D).$$

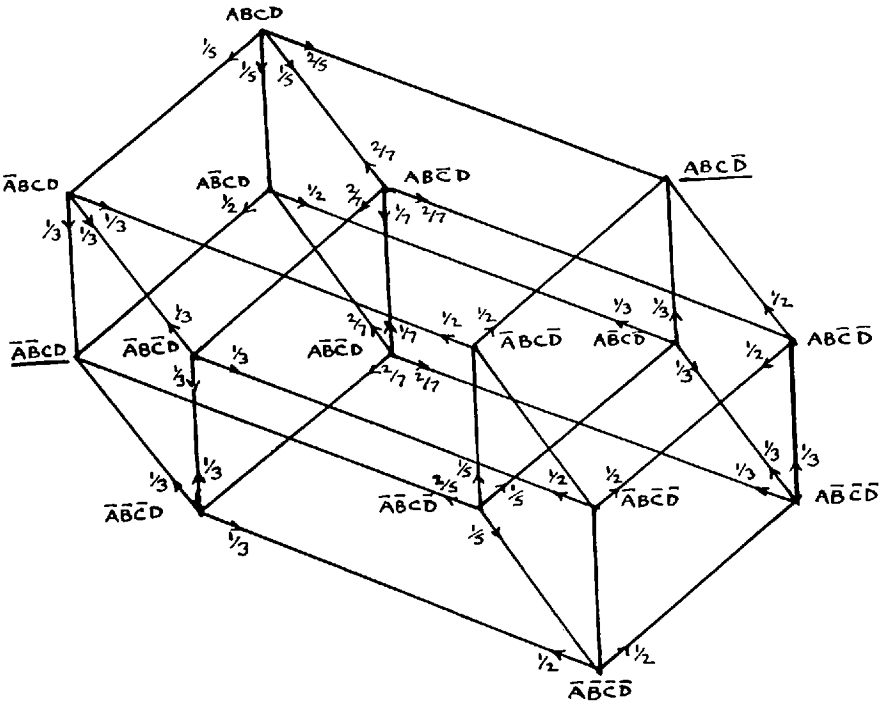
The state transition diagrams corresponding to equi-probable and weighted probability transitions are shown respectively in figs. 4 and 5. The two desired states or goals for this particular problem are shown underlined in the diagram. Note that several transitions, indicated by the absence of an arrow, are not allowed.

We have examined several five-variable problems using both unweighted and weighted probability transitions with inconclusive results†. Although in some problems some advantage is obtained by introducing weighted probability transitions, i.e. the number of transitions required in the goal-seeking process is less than with equi-probability transitions, this is

† The weighted transition process is essentially of the kind described as a Monte Carlo process for which computers have been particularly effective. In our investigations we used a random number generator to determine the transitions on a weighted probability basis.

not always the case and, indeed, in some problems the weighting process actually increased the average number of transitions required. The explanation of such anomalous behaviour in logical goal-seeking processes of this kind is that redundancies in the statement of the rules tend to nullify any advantage which might be expected in the case of weighted transition probabilities. Moreover, in practice, due to interactions between variables, it is impossible to avoid such redundancy and accordingly the weighted-transition probability method must be regarded as of academic interest only.

Fig. 5



Weighted transition probabilities.

We can regard the optimization of goal-seeking processes as means whereby the number of transitions between an arbitrary state or point in the problem-space and the desired state is minimized. And, in order to determine the most effective transitions to make, it would be desirable to have some idea of the 'direction' to go and the 'distance' in terms of transitions from the desired state. In the problem-space there is clearly a minimum 'distance' between any two states and the achievement of a goal-seeking procedure which traces such a path is our major aim.

In the case of automatic regulation systems we can always measure the departure of the state of a system from the desired state and the direction

in which corrections should be applied, and the magnitude of the corrections are known. Unfortunately, in the case of two-valued logical goal-seeking processes, no such criterion is available and it frequently happens that in a single transition a state which does not satisfy any of the required rules is transformed into a state which satisfies all the rules and there is no information available which would indicate *a priori* that a single transition will lead to the desired goal. It is perhaps not surprising that any system of logic based on two values is not amenable to a rationalization of the goal-seeking process. An analogous situation would be the search for a target aircraft using a radar with an infinitely narrow beam.

The fact that certain processes may be described in terms of many-valued logical connectives led us to consider the problem of optimizing goal-seeking procedures in these cases.

§ 3. Statement of Many-valued Logical Problems

3.1. *Many-valued Logical Statements*

In the field of control technology virtually all applications of symbolic logic have been in the synthesis of switching and computing circuits which are inherently made up of two-state devices. From the standpoint of practical application the many-valued logics proposed, for example, by Post (1921) and Lewis and Langford (1950), have not been investigated widely. Perhaps the most useful interpretation of many-valued logics is that they combine the ideas of a two-valued logic (i.e. truth and falsity) and the theory of probability. Thus, the truth-values in a four-valued logic might be regarded as corresponding to truth, plausibility, implausibility, and falsity. Moreover, consistent and complete logics have been developed for many-valued variables, although it should be stressed that all many-valued logics can be reduced to two-valued logic; however, in most cases the reduction process would be impracticable.

We consider, especially in the fields of data-processing, complex plant protection, and production scheduling processes, that there will be an increasing application of the basic concepts of many-valued logics and accordingly the process of establishing solutions to problems stated in such terms is of interest. Initially, we shall restrict the discussion to four-valued logics and assume the values to be 0, 1, 2, and 3. We associate ' truth ' with the truth-value 3, and ' falsity ' with the truth-value 0. In the case of a m-valued logical problem, a solution to the problem will be any assignment of truth-values to the variables for which the complete ' statement ' of the problem takes the truth-value m.

If A, B, etc., are many-valued variables we denote the associated truth-values by a, b, etc. The truth-value of a logical function F_i(A, B . . .) is correspondingly written $f_i(a, b \ldots)$.

The ' basic ' connectives, and the corresponding truth-functions, of the Post Logic are defined in table 2, and the truth-tables for a four-valued logic are given in fig. 6.

Table 2

Connective	Symbolic connective	Truth-function
Conjunction	$F_1=(A \cap B)$	$f_1=\min(a, b)$
Disjunction	$F_2=(A \cup B)$	$f_2=\max(a, b)$
Negation	$F_3=(\sim A)$	$f_3=(m-1-a)$
Implication	$F_4=(A \rightarrow B)$	$f_4=\begin{cases}(m-1) & \text{for } b \geq a \\ 0 & \text{for } b<a\end{cases}$
Cyclic Negation	$F_5=(A')$	$f_5=\begin{cases}(a-1) & \text{for } a>0 \\ (a-1+m) & \text{for } a=0\end{cases}$
J-function	$F_6=J_i(A)$	$f_6=\begin{cases}m-1 \text{ if } a=i \\ 0 \text{ if } a \neq i\end{cases}$
	$i=0, 1, \ldots (m-1)$	

Fig. 6

$F_1(A, B)$
$A \cap B$

∩	3	2	1	0	B
3	3	2	1	0	
2	2	2	1	0	
1	1	1	1	0	
	0	0	0	0	
A					

Conjunction
(a)

$F_2(A, B)$
$A \cup B$

∪	3	2	1	0	B
3	3	3	3	3	
2	3	2	2	2	
1	3	2	1	1	
0	3	2	1	0	
A					

Disjunction
(b)

$F_3(A)$
$\sim A$

A	~A
3	0
2	1
1	2
0	3

Negation
(c)

$F_4(A, B)$
$A \rightarrow B$

→	3	2	1	0	B
3	3	0	0	0	
2	3	3	0	0	
1	3	3	3	0	
0	3	3	3	3	
A					

Implication
(d)

$F_5(A)$
A'

A	A′
3	2
2	1
1	0
0	3

Cyclic-Negation
(e)

POST CONNECTIVES

$f_1=$ Min. (a, b) denotes the value of a or b, whichever is the smaller, and similarly for f_2 which is given by the larger value. Note that F_1, F_2 and F_3 are merely generalizations of the corresponding two-valued logical connectives, AND; OR; NOT. The implication connective in a many-valued logic may be defined in a variety of ways; F_4, as defined above, is one such definition, which is used in the problems considered subsequently.

The connective defined by F_5, the so-called ' cyclic negation ' connective, can be regarded as another way of defining negation in a many-valued logic. Nevertheless, it is still necessary to introduce the J-functions, $F_6 = J_i(A)$, in order to define a logical connective which has a property of two-valued Negation not included in F_3 and F_5.

The corresponding Conjunction, Disjunction and Negation connectives, defined in terms of the Lewis Modal Logic, are shown as truth-tables in fig. 7. The Post and Lewis definitions of Negation are identical. The Lewis connectives are introduced to illustrate the point that more than one form of many-valued logic exists.

Fig. 7

$F_1'(A, B)$
$A \cap B$

∩	3	2	1	0	B
3	3	2	1	0	
2	2	2	0	0	
1	1	0	1	0	
0	0	0	0	0	
A					

Conjunction
(a)

$F_2'(A, B)$
$A \cup B$

∪	3	2	1	0	B
3	3	3	3	3	
2	3	2	3	2	
1	3	3	1	1	
0	3	2	1	0	
A					

Disjunction
(b)

$F_3'(A)$
$\sim A$

A	~A
3	0
2	1
1	2
0	3

Negation
(c)

LEWIS CONNECTIVES

Viewed in the light of the ' problem-space ' concept introduced earlier, the representation of a logical situation expressed in terms of n many-valued (m) variables, can be obtained by dividing each ' edge ' of the corresponding two-valued n-variable hypercube into m equally spaced points. A single transition in this problem-space involves a change in a single variable by only a single step (i.e. x_i to $x_i + 1$). And the m-valued, n variable problem-space consists of m^n ' n-dimensional sub-hypercubes '.

3.2. *Binary-coding of m-valued Variables*

In designing m-valued connective circuits, i.e. decision-elements, (Calderwood and Porter 1958), and in programming a digital computer to handle problems stated in m-valued logics, it is convenient to code the m-valued variables in binary form.

Suppose in a m-valued logic the number of values is given by $m = 2^N$. If X is a m-valued variable it can accordingly be represented by N two-valued variables $X_1, X_2, \ldots X_N$. And if x_j is the truth-value of X_j we can write the truth-value of X as

$$x = \sum_{j=1}^{N} x_j \cdot 2^{j-1} \quad \ldots\ldots\ldots \quad (1)$$

in which x_j is 0 or 1.

A situation describable in terms of n many (m)-valued variables may now be considered as a situation describable in two-valued logics in which there are $n.N$ variables.

3.3. *The Measure Function*

It is convenient to define a function

$$G(x)=\sum_{j=1}^{N} x_j \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (2)$$

which gives the total number of ' 1's ' in the binary representation of the truth-value of X.

If $F(X_i)$ is a logical function of the 2^N-valued variables X_i $(i=1, \ldots n)$, and the truth-value of F is $f(x_i)$ we can generalize (1) above as†

$$x_i=\sum_{j=1}^{N} x_{ij} \,.\, 2^{j-1} \quad (x_{ij}=0 \text{ or } 1), \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (3)$$

$$f=\sum_{j=1}^{N} f_j \,.\, 2^{j-1} \quad (f_j=0 \text{ or } 1), \quad . \quad . \quad . \quad . \quad . \quad . \quad (4)$$

and (2) is written

$$G(F)=\sum_{j=1}^{N} f_j. \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (5)$$

We will assume that $F(X_i)$ describes a situation which is satisfied by a single combination of the 2^N-valued variables $(X_1, X_2, \ldots X_n)$—alternatively we say that $F(X_i)$ defines a problem which has a single solution. The truth-value of F is given by (4), and we consider the value of the function $G_1(F)$, given by (5), as corresponding to a measure of the ' distance from the goal ' for a particular combination of values of $(X_1, X_2, \ldots X_n)$. In terms of the problem-space, for $n.N$ two-valued variables, we can regard this measure as being made up of single transitions in which only one of the two-valued variables (of which the m-valued variables are built up) changes—the corresponding change in the truth-value of the m-valued variable for each transition, in this problem-space is $2^a(a=0, 1, \ldots (N-1))$‡.

The problem-space can be represented as a $n.N$-dimensional hypercube (cf. fig. 4) which is symmetrical with respect to each of the 2^{nN} vertices. This hypercube can be regarded as having $(nN+1)$ levels, or planes, which are readily recognizable in the four-dimensional hypercube shown in fig. 4; it is convenient to call these ' transition levels '. If we assume the hypercube is orientated so that the state corresponding to the solution is at

† In the case of a logical function of two-valued variables f takes the values 0 or 1. If, and only if, the logical function $F(X_i)$ is satisfied by a given combination of variables, say X'_i, can we write $f=1$, otherwise $f=0$. The measure function $G(F)$ in this case can only take the values 0 or 1, and thus provides no additional information of value in seeking the goal.

‡ This ' measure of distance ' was proposed by J. D. Cowan, Department of Electrical Engineering, M.I.T., who is investigating its theoretical validity—this work is as yet unpublished.

the ' top ' of the diagram, the minimum number of single transitions, from a given state to the desired state, corresponds to the number of ' transition levels ' between the two states. For example, the number of transition levels between the ' complement ' of the desired state and the desired stated is nN. In the case of multi-solutions, re-orientation of the hypercube for each solution (desired state) would be necessary.

In order to investigate the nature of $G(F)$ in some specific cases we have used the Pegasus computer to determine the appropriate truth-values and associated functions for several 16-valued, three-variable logical problems. The problem space contains $2^{12}=4096$ points, each of which must be examined; each state can be represented by a 12-binary digit word.

§ 4. Application of a Digital Computer

4.1. *Programming Procedure*

Two machine programmes called I and II, are required; the corresponding programme flow diagrams are given in figs. 8 and 9 respectively. Special sub-routines are required for the generation of the Gray Code (4096 combinations each differing from its predecessor in the sequence by a change in a single digit of the 12-digit word) and the setting-up of the logical rules.

In programme I, each combination of variables Y_i† enters the ' logic sub-routine ' (corresponding to the rules or constraints) and we obtain the appropriate value of $f(Y_i)$, the truth-value of $F(Y_i)$, and the value of $G(F_i)$ is determined by summing the 1's in $f(Y_i)$. The values of $f(Y_i)$ and $G(F_i)$ are punched out on standard five-hole paper tape. The truth-value $f(Y_i)$, associated with each combination of variables, is tested to determine whether or not it corresponds to a solution, i.e. $f(Y_i)\equiv 1111$; if a solution the combination of variables producing it is stored in the magnetic drum store of the computer for use in programme II. On the completion of programme I (after each of the possible combinations of variables has been examined), the list of solutions to the problem is printed-out—the particular problems chosen for detailed study had no more than five solutions.

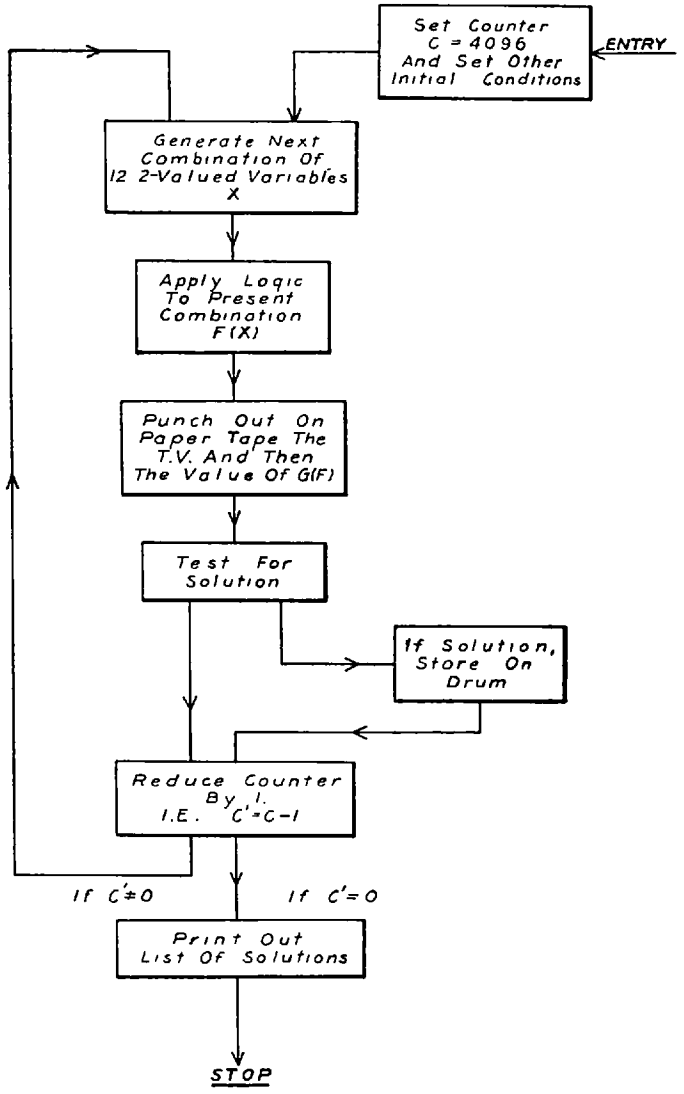
The programme flow diagram of programme II is shown in fig. 9. In addition to the programme tape we feed into the machine the output tape T (giving the values of $f(Y_i)$ and $G(F_i)$) obtained from programme I and the 12-bit word defining one of the solutions; we refer to the latter as S. We arrange for the identical sequence of combinations to be examined as in programme I.

Each combination of variables Y_i is compared with S and a non-equivalence operation is carried out in which a new word P_i is generated having 1's in positions where the digits of Y_i and S are non-equivalent and 0's otherwise. Further, the number of 1's in P_i is summed and this

† The subscript i as in Y_i now refers to a particular ' trial combination '.

value, say p_i, corresponding to the number of transition levels between combinations Y_i and S (note that $0 \leqslant p_i \leqslant 12$), defines a storage location in the drum store. The associated truth-value, $f(Y_i)$, is added into this

Fig. 8

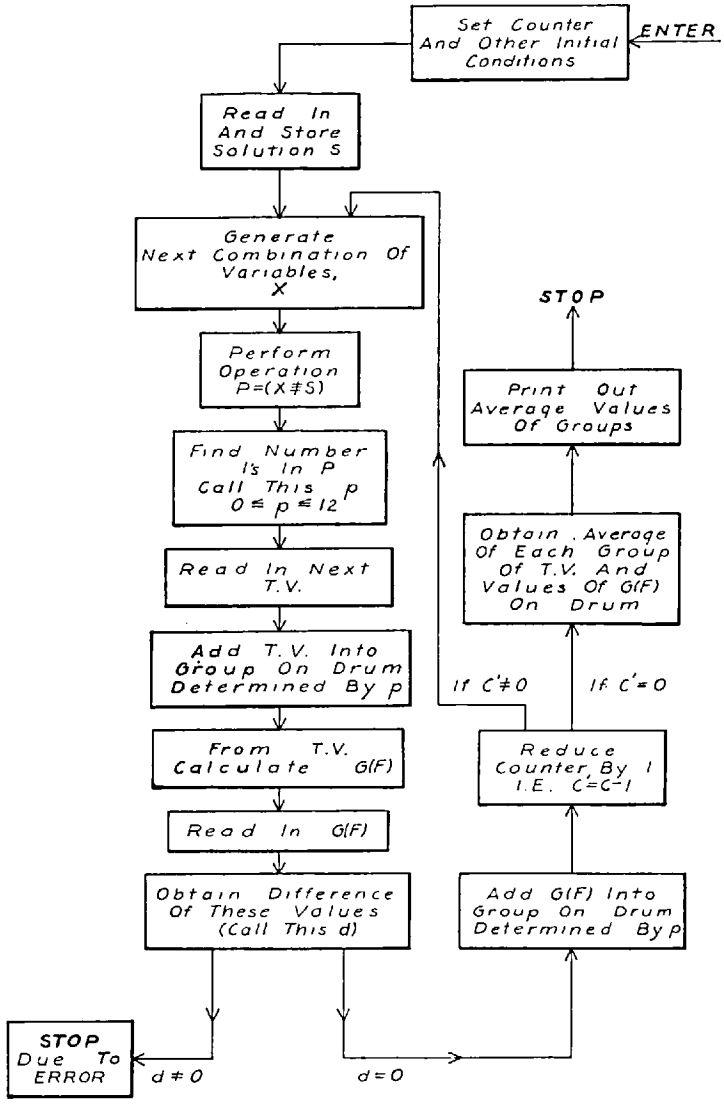


Flow diagram of computer programme I.

storage location p_i. Similarly, the value of $G(F_i)$ is added into another drum storage location also defined by p_i. Upon completion of programme II, after all combinations have been considered, the average values of

$f(Y_i)$ and $G(F_i)$ accumulated in the 'p-storage locations' are obtained—the accumulated sums are divided by the number of entries (points in respective problem-space levels) and these average values are printed-out.

Fig. 9



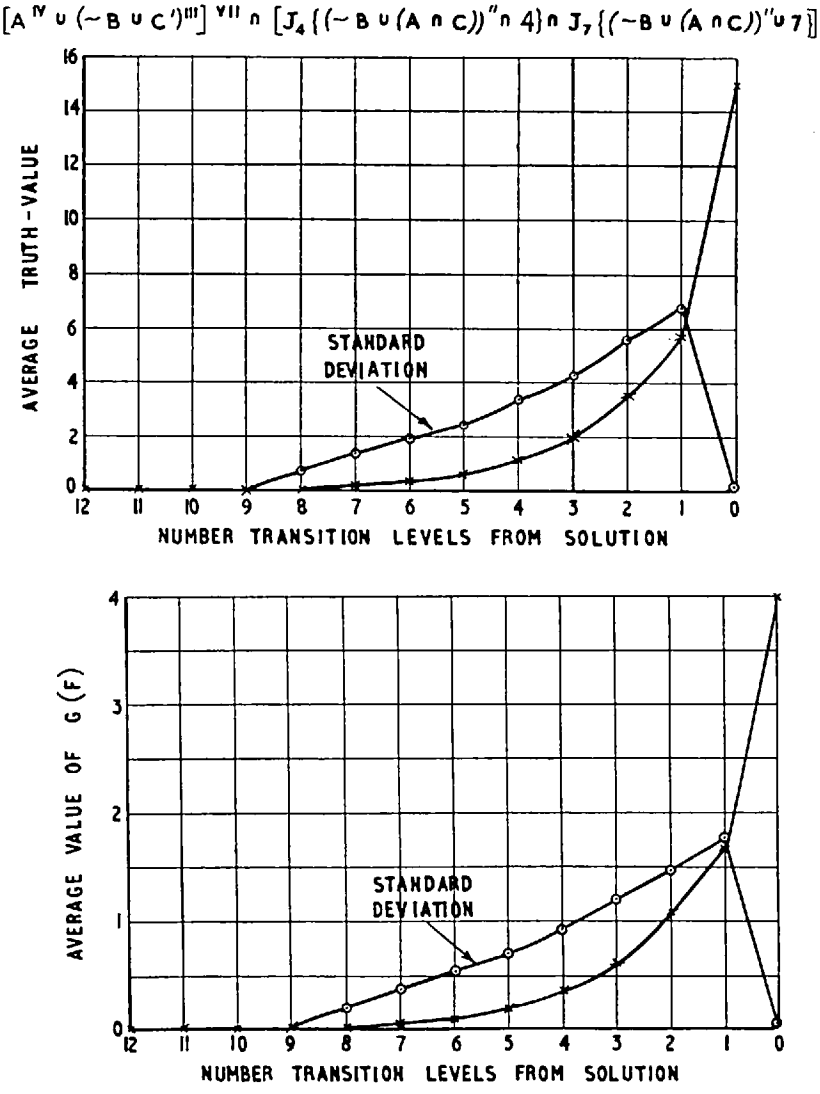
Flow diagram of computer programme II.

Moreover, the standard deviations of the average truth-values and the average values of $G(F_i)$ are determined for each transition level and the results are also printed-out.

4.2. *Results*

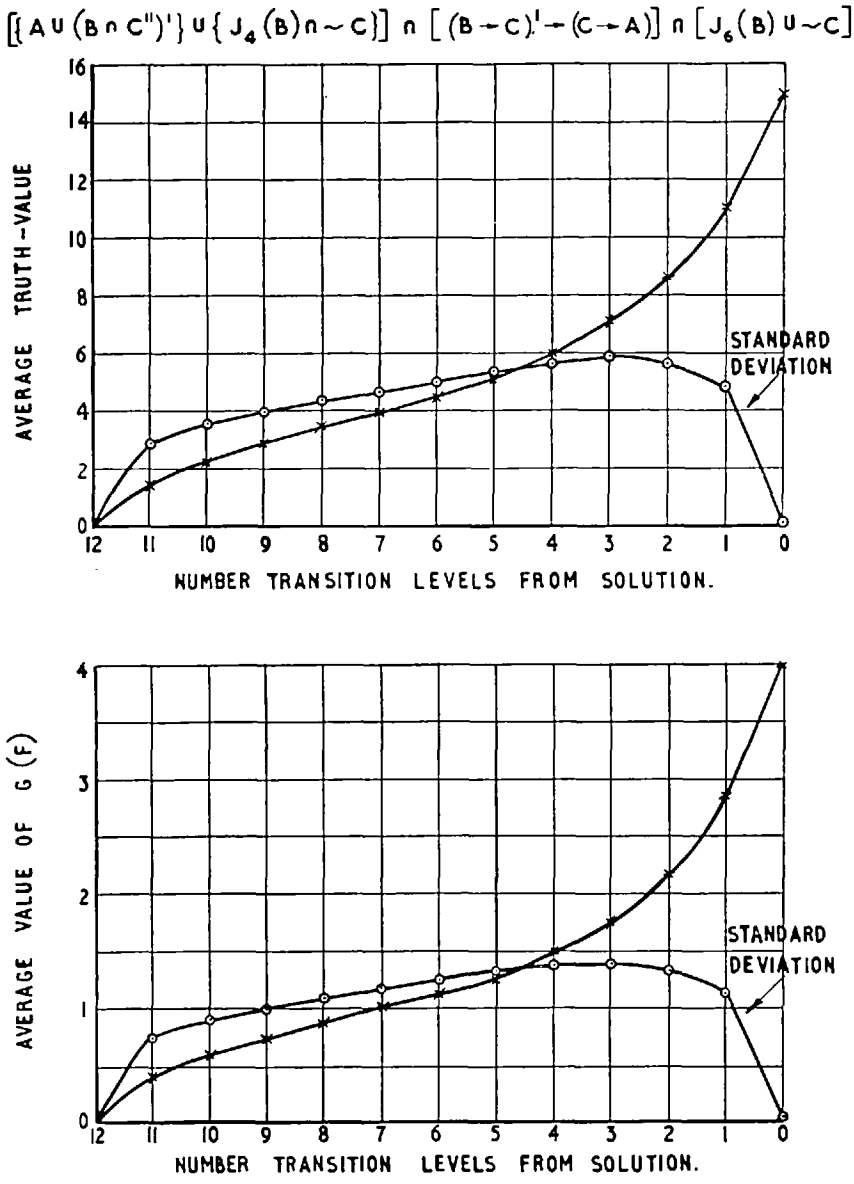
The results for two typical problems are given in figs. 10 and 11. In both cases the average truth-values, the average values of $G(F_i)$, and the

Fig. 10



corresponding standard deviations are plotted as functions of the number of transition levels ' away from a solution '. Although, *a priori*, we did not anticipate a close relationship between $f(Y_i)$ and $G(F_i)$, these functions are almost identical in form, and this matter alone deserves further study. Perhaps the most important finding is the obvious

Fig. 11



existence of a ' trend ', i.e. there is a correlation between the average truth-value, or the average value of $G(F_i)$, and the ' nearness ' to a solution. The nature of the trend function will clearly depend upon the nature of the problem, e.g. the logical situation.

The standard deviations are appreciably greater than we expected and would, in an actual goal-seeking procedure, tend to mask the trend effect,

On the other hand, $m = 16$ corresponds to a comparatively 'low-valued logic' and we believe that for values of $m \geqslant 128$ the trend would be more marked.

In the application of the measure function $G(F)$ to optimize logical goal-seeking processes we would obtain the values of $G(F)$ for points chosen at random in the problem-space and concentrate the search for solutions in those regions where $G(F)$ has high values. The process of goal-seeking would be analogous, in principle, to the evolutionary operation procedure for optimizing the performance of chemical processes (Box 1954). It is intended to extend the present investigations by using the computer to carry out this 'sampling' of the problem-space for particular cases, and to assess quantitatively the efficiency of the search method based on $G(F)$.

ACKNOWLEDGMENTS

We wish to thank the Research Grants Committee of the Department of Scientific and Industrial Research for the award of a grant to pay for time on the Pegasus computer. We acknowledge the many fruitful discussions we have had with Mr. J. D. Cowan.

REFERENCES

BOX, G. E. P., 1954, *Biometrics*, **10**, 14.

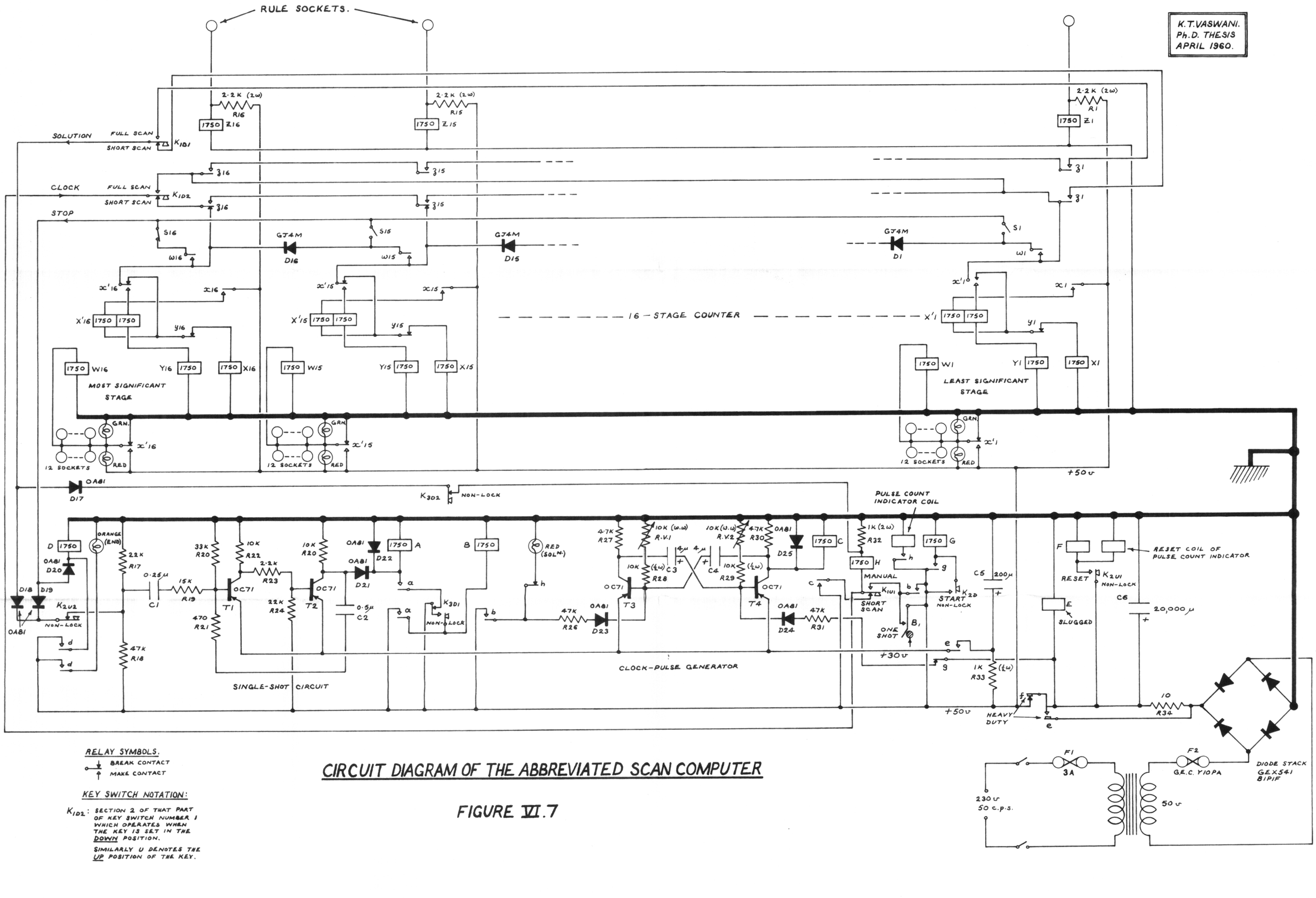
CALDERWOOD, J. H., and PORTER, A., 1958, *J. Electron. Contr.*, **4**, 466.

LEWIS, C. I., and LANGFORD, C. H., 1950, *Symbolic Logic* (New York : Dover Publications Inc.).

MCCALLUM, D. M., and SMITH, J. B., 1951, *Electron. Eng.*, April, 1951, and December, 1951.

POST, E. L., 1921, *Amer. J. Maths.*, **43**, 163.

K.T.VASWANI.
Ph.D. THESIS
APRIL 1960.



RELAY SYMBOLS.

BREAK CONTACT

MAKE CONTACT

KEY SWITCH NOTATION:

K_{ID2} : SECTION 2 OF THAT PART OF KEY SWITCH NUMBER 1 WHICH OPERATES WHEN THE KEY IS SET IN THE DOWN POSITION.

SIMILARLY U DENOTES THE UP POSITION OF THE KEY.

CIRCUIT DIAGRAM OF THE ABBREVIATED SCAN COMPUTER

FIGURE VI.7

U N I V E R S I T Y O F L O N D O N

F A C U L T Y O F E N G I N E E R I N G

On Possibilities for the Automatic Solution of Logical Problems.

A Thesis presented for the Degree of Doctor of Philosophy

by

K.T. VASWANI

Department of Electrical Engineering,
Imperial College of Science & Technology,
City and Guilds College,
LONDON, S.W.7.

April 1960.

ABSTRACT

This thesis is concerned with the solution of the class of problems in two-valued and many-valued logics in which a given set of constraints must be satisfied simultaneously. The solution consists in enumerating all assignments of truth-values to the variables satisfying the constraints.

The concept is introduced of a problem space consisting of a set of discrete points, each point uniquely representing one assignment of truth-values. Any solution procedure may be thought of as a search through a problem space for the solution points. For many-valued problems two measure functions are proposed and investigated which could be used to guide a machine making a search through a problem space towards regions most likely to yield solutions. Using such a measure function, a solution procedure is outlined which should prove effective for solving problems in many-valued logic too large to be solved by an exhaustive scan method.

A method is described of generating an abbreviated scan of the truth-value assignments to any problem so that all the solutions are found although only a fraction of the assignments are considered. Operating on this principle, a machine has been built for solving problems in two-valued logic. The possibility is investigated of programming the machine so as to obtain a scan containing a minimal number of steps In comparison with an exhaustive scan, this machine reduces the length of scan by two orders of magnitude, on average, for 16 variable problems. This factor of improvement is expected to increase with the number of variables in the problem. This technique therefore enables problems to be solved which are too large to be solved by any other existing method.

The abbreviated scan principle is generalized to apply to problems in which the variables may assume any of a finite number of states and in which algebraic as well as logical constraints occur.

ACKNOWLEDGEMENTS

The work described in this thesis was carried out in the Electrical Engineering Department of the Imperical College of Science and Technology, and the author wishes to thank the head of that Department, Professor A. Tustin, for making this possible.

The author is deeply indebted to Professor A. Porter (now Dean of Engineering, University of Saskatchewan) and to Professor E.C. Cherry for their encouragement and for their invaluable supervision of this project.

Thanks are also due to Mr. J.D. Cowan and to Mr. D.G. Watts for many fruitful discussions and contributions at numerous stages of the project.

The author also wishes to express his appreciation to the University of London and to International Computers and Tabulators Ltd., for providing research studentships, to the Department of Scientific and Industrial Research for helping to sponsor the project and to Siemens Edison Swan Ltd. and Ericsson Telephones Ltd. for freely contributing components for building the abbreviated scan computer.

PREFACE

Since many people in an Engineering Faculty may be unfamiliar with symbolic logic more space has been devoted to the introductory Sections of this thesis than is customary.

Throughout this thesis the following are used synonymously:

elementary propositions;
propositional variables;
logical variables;
variables.

TABLE OF CONTENTS

		Page
Introduction		10
Section I	Two-Valued Symbolic Logic	
I.1	Logical Connectives	21
I.2	The Symbolic Expression of a Problem	22
I.3	Tautologies	25
Section II	Some Existing Logical Machines	
II.1	Jevons' Mechanical Machine	28
II.2	The Kalin-Burkhart Machine	29
II.3	The McCallum-Smith Scanning Machine	29
II.4	The McCallum-Smith Feedback Computer	29
II.5	A Feedback Computer with Weighted Transition Probabilities	32
Section III.	Some Ideas on Solving Problems in Many-Valued Logic	
III.1	Generalization from Two-Valued to Many-Valued Logic	34
III.2	The Connectives of Post, Lukasiewiez and Tarski	35
III.3	The Connectives of Lewis	37
III.4	Expression of Problems in Terms of Many-Valued Logic	38
III.5	Reduction of Many-Valued Logic to Two-Valued Logic	42
III.6	Problem Spaces and Measure Functions	46

		Page
Section IV.	Programming the Ferranti "Pegasus" Computer to Investigate the Proposed Measure Functions	
IV.1	The Method of Investigation	58
IV.2	The Problems Used	60
IV.3	Pegasus Programme A	63
IV.4	Pegasus Programme B	66
IV.5	Results	70
IV.6	The Use of Measure Functions in Logical Problem Solving Procedures	78
Section V.	An Improved Technique for Scanning the Truth-Value Assignments to Problems in Logic	
V.1	Use of a Binary Counter for Generating an Exhaustive Scan of the Truth-Value Assignments to Problems in Two-Valued Logic	81
V.2	Generation of an Abbreviated Scan of the Truth-Value Assignments to Two-Valued Problems.	82
V.3	On the Ordering of Constraints and Variables	87
V.4	Generalization of the Abbreviated Scan Technique to Many-Valued Combinational Problems	91
Section VI.	Mechanization of the Abbreviated Scan for Problems in Two-Valued Logic	
VI.1	The Purpose of the Machine	94
VI.2	The General Requirements of the Machine	95
VI.3	Block Diagram of the Complete Machine	96
VI.4	A Binary Counter with Simultaneous Carry Propagation Between All Stages	98
VI.5	The Operation Cycle of the Machine	102
VI.6	The Logical Connective Units	104
VI.7	Operation of the Complete Machine	108
VI.8	A Fuse Indicator	113

		Page
Section VII.	An Investigation of Methods of Programming the Abbreviated Scan Machine	
VII.1	The Ideas to be Tested	116
VII.2	Calculation of the Proportion of Truth-Value Assignments Satisfying a Constraint	118
VII.3	The Method of Investigation	121
VII.4	Problem One	122
VII.5	Problem Two	125
VII.6	Problem Three	127
VII.7	Summary	133
Section VIII.	Application of the Abbreviated Scan Technique to the Academic Timetable Problem	
VIII.1	The Problem	135
VIII.2	Expression of the Problem in Terms of State Variables	136
VIII.3	The Method of Scanning the Variables Within a Group	137
VIII.4	The Complete Scanning Procedure	139
Section IX.	Conclusion	
IX.1	Measure Functions for Many-Valued Logics	144
IX.2	The Abbreviated Scan Machine	146
IX.3	On the Application of Logic	148
Appendix 1.	Solution of a Set of Logical Constraints by Systematic Reduction	148
Appendix 2.	The Abbreviated Scan for the Timetable Problem in Section VIII.	150
References		158

INTRODUCTION

The word, logic, is derived from the Greek noun λόγος, which signifies a thought or the expression of a thought.

Logic was invented by Aristotle 2,300 years ago. Aristotle was the first to attempt to put reasoning on a firm basis when he thought of the syllogism. The four basic kinds of proposition used by Aristotle were :

Universal	Affirmative	-	ALL S IS P .
	Negative	-	NO S IS P .
Particular	Affirmative	-	SOME S IS P .
	Negative	-	SOME S IS NOT P .

In these propositions S and P may be replaced by any nouns. A syllogism is constructed by taking two propositions as premises and arriving at a third proposition as a conclusion.

Example:

ALL MEN ARE MORTAL,
SOCRATES IS A MAN,
THEREFORE SOCRATES IS MORTAL .

SOME COMPUTERS ARE RELAY-OPERATED DEVICES,
ALL RELAY-OPERATED DEVICES ARE OLD FASHIONED,
THEREFORE SOME COMPUTERS ARE OLD FASHIONED.

Since each of the three lines of a syllogism may be chosen from one of the four basic kinds of proposition there are 64 possible arrangements or moods of the syllogism. However, not all the moods are valid. An example of an invalid mood is :

SOME ELEMENTS ARE GASES,
SOME GASES ARE LIGHTER THAN AIR,
THEREFORE SOME ELEMENTS ARE LIGHTER THAN AIR.

Rules exist for deciding the validity of the different forms of syllogism and, of the 64 possible moods, 19 prove to be valid.

For many centuries people thought that all valid forms of reasoning and inference were reducible to the valid moods of the syllogism. During the 16th to the 19th centuries many people realized the futility of the Aristotelian logic but no significant advance was made until 1847 when G. Boole published his book "The Mathematical Analysis of Logic". This was followed in 1854 by Boole's revolutionary book "An Investigation of the Laws of Thought". Referring to this treatise, Boole says (Chapter I) :

> "It is designed, in the first place, to investigate the fundamental laws of those operations of the mind by which reasoning is performed.
>
> "It is designed, in the next place, to give expression in this treatise to the fundamental laws of reasoning in the symbolic language of a calculus."

Boole's three fundamental laws of thought were :

1. A law of identity; stating that a thing is identical to itself.
2. The Law of Contradiction; stating that no thing can have two contradictory attributes.
   e.g. no thing can be both green and not green.
3. The Law of Excluded Middle; stating that every proposition must be either true or false.

The subject of this thesis consists in the study of solution procedures for a certain class of problem in the calculus of symbolic logic, originated by Boole, and in its modern extension, many-valued logic. The mechanization of these solution procedures involves the use of electrical computing techniques.

In the modern presentation of two-valued symbolic logic capital letters A, B, C, etc., may be used to represent <u>elementary propositions</u>.

There is no restriction upon the propositions which may be of concern and the following are examples:

(i) The cost of living is rising.
(ii) "2 + 2 = 5"
(iii) The railway signal is set at danger.
(iv) Silence is golden
(v) An increase in wages is being considered.

These elementary propositions may be combined to form complex propositions by the use of such words as AND, OR, NOT, IF - - - THEN, IS IDENTICAL TO. Examples of such complex propositions are :

(a) The cost of living is rising AND an increase in wages is being considered.
(b) "2 + 2 = 5" OR silence is golden.

The words AND, OR, etc., used to combine elementary propositions are termed logical connectives and, in the calculus of symbolic logic they are represented by signs.

Every proposition has associated with it a truth-value having the value unity for a true proposition and the value zero for a false proposition.

Any complex proposition may be represented as a string of symbols consisting of capital letters denoting elementary propositions and signs denoting logical connectives. Parentheses might also be used to obviate ambiguity. It is possible, by using the rules and definitions of symbolic logic (see Section I) to find the truth-value of a complex proposition as a function of the truth-values of the elementary propositions involved. As a trivial example of this consider the truth of the complex proposition, "The cost of living is rising AND silence is golden." This is defined as being true if, and only if, both the elementary propositions are true. Let these elementary propositions be denoted by A and B respectively. There are four

combinations of truth and falsity of these elementary propositions and the following table (Fig. 1) shows the status of the complex proposition in these cases.

A	B	A AND B
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

Figure 1.

A	B	A AND B
1	1	1
1	0	0
0	1	0
0	0	0

Figure 2.

The truth-table, obtained by denoting truth by unity and falsity by zero, is shown in Fig. 2. The truth-table for a complex proposition involving n elementary propositions has 2^n rows corresponding to the 2^n possible combinations of truth and falsity to the elementary propositions.

As an example of the type of thing expressible in terms of symbolic logic consider the following problem :

A manufacturer must decide what items to produce in order to maximize his profit. Suppose these items are designated by the letters A, B, C, D and E . He must produce both items A and B or produce item C in order to satisfy certain market requirements. However, it is uneconomical to manufacture items B and C simultaneously. Further, if item D is chosen, then it is convenient to produce item B. To ensure efficient use of machinery it is necessary to produce item D or item E .

Determine the combinations of items from which the manufacturer is free to choose.

Let A represent "item A is manufactured."
Let B represent "item B is manufactured."
etc.

The problem then consists of the following four rules which must be satisfied simultaneously :

1. (A AND B) OR C.
2. NOT (B AND C).
3. IF D, THEN B.
4. D OR E.

The solution of this problem by construction of its truth-table appears in Section I, (p. 24 , Fig. I.5). It is found that the manufacturer has five combinations of products from which to choose.

Many people have suggested that symbolic logic might be applicable to the solution of certain optimization problems where some complex process, such as a chemical plant, must be controlled and kept working at optimum efficiency in an environment which is continually changing due to fluctuations of the system input parameters. (e.g. cost of raw materials, customers' demands, etc.). D.G. Prinz and J.B. Smith (ref: 1) have suggested as possible applications the checking of legal documents for consistency and for loopholes; checking signal operations on complex railway layouts; air traffic control at airports where high landing and take-off frequencies exist and certain aircraft must be given priority. Another possible application is to the problem of constructing a timetable for a school or for a railway service. A. Rose (ref: 2) has published a paper on this and the problem is also considered here in Section VIII.

Most of the problems arising, in addition to containing constraints of a logical nature, involve variables which are continuous. Variables such as time, voltage, velocity and position can be quantized and expressed in binary form. In this way all constraints can be expressed in terms of logic. However, this is a very artificial way of dealing with a problem and the number of propositional variables (as the elementary propositions in a logical problem are called), brought

into the problem is usually great.

The logic so far discussed is called two-valued since, in it, any proposition is either true or false, having a truth-value of unity or zero. B. Rosser (ref: 3) and a few others believe that two-valued logic may well be inadequate for dealing with many of the problems and situations occurring about us. In modern mathematics and physics situations repeatedly occur where an element of uncertainty exists and certain things cannot be thought of as being simply true or false. In the branch of mathematics known as set theory the relationships are studied between sets of things having certain defined attributes in common, e.g. in making a statistical survey one might be interested in the set of people born in the British Isles between 1950 and 1959 inclusive. A proper set is defined as a set which does not contain itself as a member. An example of an improper set is the set of all conceivable ideas. Since this is, itself, a conceivable idea, this set is a member of itself. Consider the set, U , of all proper sets. Now pose the question: Is U a proper set? If U is a proper set then, since U is the set of all proper sets, it contains itself. However, this implies that U is an improper set. If U is an improper set it contains itself as a member. Since all members of U are proper sets this means that U is a proper set. D.A. Bočvar (ref: 4) has shown that the use of a three-valued logic can clear up this ridiculous situation. Another suggestion for the adoption of a three-valued logic is put forward by H. Reichenbach in his book "Philosophic Foundations of Quantum Mechanics". He shows how certain anomalies existing in quantum mechanical theory may be suppressed by using a three-valued logic.

Many people think that problems involving logical constraints as well as continuous variables might be formulated in terms of many-valued logics. Continuous variables could be quantized and a logic with a suitable number of values chosen. A many-valued system could be

interpreted by saying that the truth-value of a proposition indicates the plausibility of the proposition, or the probability of the proposition being true. A. Rose (ref: 2) has attempted to formulate the problem of timetable construction and also a problem regarding the most economic use of machines in a production line in terms of many-valued logics.

A very large class of problems both in two-valued and many-valued logics can be expressed in the following form:

Given a set of logical rules or constraints involving propositional variables and defined logical connectives, find all assignments of truth-values to the variables which satisfy the constraints.

This thesis is concerned with the solution of this class of problem.

One way of solving such problems is to construct the complete truth-table for the combined set of problem constraints. It has already been noted that the truth-table for a problem involving n two-valued variables has 2^n rows. The corresponding number for a problem in m-valued logic is m^n. The truth-table could be constructed by hand for two-valued problems in up to about seven variables. Large problems would have truth-tables with at least 256 rows and the manual construction of these would be extremely laborious and the likelihood of making mistakes would be quite high. Since the rules for constructing truth-tables are of a very mechanical nature the whole process can be easily mechanized. Indeed, this has been done, and one mechanical machine and several electrically operated machines for automatically constructing the rows of truth-tables are described in Section II. However, if an electronic computer were used to scan through the 2^{50} assignments of truth-values to a 50 variable problem at a rate of 1,000,000 per second it would take about 30 years to complete the task! Clearly then, however optimistic one may be regarding improvements in electronic techniques and consequent increases in computer operating speeds, the largest two-valued logical problems which can be tackled by these methods are not likely to involve

more than 50 propositionalvariables.

Should it prove desirable to formulate practical problems in terms of logic it is very likely that problems would be encountered involving considerably more than 50 variables. It would therefore be essential to find another way of solving these problems not necessitating the investigation of all possible assignments of truth-values to the propositional variables.

All the machines so far proposed, with one exception, have operated on the principle of making an exhaustive scan of all possible assignments of truth-values and thereby working through the complete truth-table of a problem. Consequently, these machines are strictly limited in the number of variables that they can handle. Far too much equipment would be needed if a machine were to consider all assignments of truth-values of a problem simultaneously; for this reason all existing logical computers of any significance adopt a time-sequential scan of the truth-value assignments.

The machine mentioned above as being an exception, which does not make an exhaustive scan of the truth-value assignments, was built by D.M. McCallum and J.B. Smith (refs: 5 and 6) in 1951. They call this machine a feedback logical computer. It makes a sequential scan of truth-value assignments in such a way that the truth-values of the propositional variables change in a semi-random manner according to a Markoff process. This is a non-deterministic machine, i.e. for a given problem and the machine being in a given state, its next "move" is not uniquely determined. The machines moves are determined partly by an element of chance. McCallum and Smith showed that this machine is particularly well suited to a certain class of problems for which it might find one or a few solutions very rapidly. On the other hand, some problems are ill-suited to solution by this machine, and for these problems the machine might take longer to find one solution than a machine making an exhaustive scan of truth-value assignments. The

great disadvantage of this machine is that, however long it has been left to work on a problem, it is impossible to determine whether all the solutions to the problem have been found. In fact the machine might never find all the solutions to a problem. The operation of this machine is discussed in detail in Section II.

A concept often useful when thinking about problems of this nature is that of a problem space or phase space. The 2^n assignments of truth-values to the n variables in a problem in two-valued logic may be associated with the 2^n vertices of an n-dimensional hypercube. The problem space consists of 2^n discrete points and has n orthogonal dimensions. The solutions to a given problem constitute a subset of these points. Any solution procedure may be thought of as a search through the problem space for all those points representing assignments of truth-values satisfying all the logical constraints of a problem. If one point in the space represents a solution then some points in the space will be "near" to this solution point while others will be "remote". However, this is a space without a metric. Nevertheless, it is very tempting to ask "how far" any selected point is from a given solution point. Is it possible to define a measure function whose value at any point in the space would convey this information? Alternatively, could a measure function be defined such that, after evaluating it at numerous points in one region of the space, a probability could be assessed of finding a solution point in this neighbourhood in the space? If such a possibility should exist, machines could be imagined which would grope their way through a problem space in an intelligent manner and "home" on points representing solutions. After much thought along these lines it was felt that it would not be possible to find a measure function of this kind for problems in two-valued logic because the only information known about any logical rule in a problem is that it is either satisfied or not satisfied. No intermediate state is possible and no information appears to be available to indicate whether successive guesses at a solution are getting "hotter" or "colder". However, many-valued logics

present a more promising picture. The truth-value of a logical rule can vary over a range of integral values and a sense of "direction" is introduced. Ideas on two possible measure functions were tested by programming the Ferranti Pegasus computer to solve problems in 16-valued logic and to investigate the behaviour of the proposed measure functions. The results appeared quite hopeful. Since this work was carried out J.D. Cowan has submitted a D.I.C. thesis (ref: 7) at Imperial College in which he proves analytically that the kind of measure function discussed here may be defined for certain classes of many-valued logics but not for two-valued logic.

Later, working on the idea of solving logical problems in a rule-by-rule manner, a scanning technique was devised whereby all the solutions to a problem can be found without making an exhaustive scan of all possible truth-value assignments. A machine has been constructed that will solve problems involving up to 16 two-valued propositional variables and up to 100 logical connectives. The machine performs a systematic scan of the truth-value assignments by first scanning only those variables involved in constraint number one. (The constraints having been ordered in some way, for the time being, supposed arbitrary.) This continues until constraint 1 is satisfied. Next, all those variables are scanned which appear in constraint 2 and have not already been scanned. This continues until constraint 2 is satisfied. The machine then proceeds to scan those variables in constraint 3 that have not already been scanned. In this way the rules are satisfied one by one. If, at any stage, it proves to be impossible to satisfy a rule, then the machine returns to the previous rule and continues scanning its variables to find another satisfactory assignment of truth-values. In the case of the 16 variable problems put on the machine the number of trial steps required to find all the solutions has been reduced by a factor of the order of 100 in comparison with the 2^{16} trial steps in an exhaustive scan. However, this factor of improvement varies from one

problem to another and it also varies for a given problem according to the order in which the logical constraints of the problem are put on to the machine. By putting a selection of problems on the machine and trying various orderings of the logical rules and variables ideas have been tested regarding the possibility of programming this machine so as to minimize the number of trial steps in the scan.

One great advantage of this machine is that it reaches a stage where it is known that all the solutions to a problem have been found. As with the feedback computer described earlier, this machine is better suited to some problems than to others. However, in the worst possible case this machine will take no more trial steps to find all the solutions to a problem than a machine making an exhaustive scan of the truth-table. With a problem having no solutions a very good factor of improvement may still be obtained.

I. TWO-VALUED SYMBOLIC LOGIC

I.1 Logical Connectives

No attempt is made to present a rigorous, axiomatic treatment of symbolic logic. Rather, an intuitive approach is adopted and sufficient detail is included only to enable an understanding of the following chapters in this thesis.

Elementary propositions are represented by capital letters A, B, C, etc.

All propositions are categorical and must be either true or false.

Every proposition, elementary or complex, has a truth-value associated with it. This truth-value is unity for a true proposition and zero for a false proposition.

Logical connectives are used for combining elementary propositions to form complex propositions. A list of the more important connectives together with the signs used to denote them appears in Figure I.1 .

CONNECTIVE	WRITTEN	SIGN(S)
NOT. NEGATION.	NOT - A .	$\sim A$; $\overline{A}$
AND. CONJUNCTION.	A AND B.	A . B
OR (INCLUSIVE). DISJUNCTION.	A OR B.	$A \vee B$
IF,.....THEN. IMPLICATION.	IF A, THEN B. A IMPLIES B	$A \supset B$
IDENTITY. EQUIVALENCE. IF, AND ONLY IF.	A IDENTICAL TO B. A EQUIVALENT TO B. A IF, AND ONLY IF, B.	$A \equiv B$
NON-IDENTITY. NON-EQUIVALENCE. OR ELSE EXCLUSIVE OR.	A NOT IDENTICAL TO B. A NOT EQUIVALENT TO B. A OR ELSE B.	$A \not\equiv B$

FIGURE I.1

The connectives listed in Figure I.1 are defined in truth-table form below.

A	~A
1	0
0	1

FIGURE I.2.

A	B	A.B	A ∨ B	A ⊃ B	A ≡ B	A ≢ B
1	1	1	1	1	1	0
1	0	0	1	0	0	1
0	1	0	1	1	0	1
0	0	0	0	1	1	0

FIGURE I.3.

These definitions are compatible with intuitive ideas and with the normal usage in conversation of the words AND, OR, etc.. However, some explanation is necessary regarding the definition of the IMPLICATION connective.

Let A represent the proposition, "John takes arsenic".

Let B represent the proposition, "John shall die".

The proposition represented by $A \supset B$ is, "If John takes arsenic, then John shall die". The upper two rows of the truth-table for $A \supset B$ are intuitively satisfactory, but what of the lower two rows? It must be noted that the proposition, "If John takes arsenic, then John shall die" says nothing of what shall happen to John if he does not take arsenic. In this case John may not die. Yet it is quite conceivable that John may die due to some cause other than poisoning by arsenic. In other words, if the antecedent proposition of an IMPLICATION is false, then the consequent proposition may be either true or false without the IMPLICATION being violated. This fits in with the lower two rows in the truth-table for $A \supset B$.

I.2. The Symbolic Expression of a Problem

A problem was given in the Introduction of a manufacturer having to decide what items to produce in order to maximize his profit. The

four logical rules (or constraints) of this problem were listed on p. 14 . These rules are expressed symbolically as follows :

1. $(A.B) \vee C$
2. $\sim(B.C)$
3. $D \supset B$
4. $D \vee E$

The truth-table for rule 1 is :

(A	.	B)	$\vee$	C
1	1	1	1	1
1	1	1	1	0
1	0	0	1	1
1	0	0	0	0
0	0	1	1	1
0	0	1	0	0
0	0	0	1	1
0	0	0	0	0

TRUTH-TABLE FOR RULE 1.

FIGURE I.4.

This table is constructed by first writing down the columns beneath the logical variables A, B and C . These give the eight assignments of truth-values. Next, the column for A.B is entered beneath the "." sign. Finally, from the A.B column and the C column, the column is obtained giving the truth-value of the complete rule. This column is entered beneath the "$\vee$" sign.

The truth-table for any logical expression, however complex, can be constructed in this way. The truth-table for the four rules in the above mentioned problem appears in Figure I.5. Five out of the 32 assignments of truth-values to the logical variables satisfy all the four logical constraints, i.e. the problem has five solutions. The corresponding rows of the truth-table are marked "X" and the solutions are :

A	B	C	D	E
1	1	0	1	1
1	1	0	1	0
1	1	0	0	1
1	0	1	0	1
0	0	1	0	1

The manufacturer therefore has five combinations of items from which to choose.

	[(A . B) ∨ C]					.	∼	[B . C]			.	[D ⊃ B]			.	[D ∨ E]		
	1	1	1	1	1	0	0	1	1	1		1	1	1		1	1	1
	1	1	1	1	1	0	0	1	1	1		1	1	1		1	1	0
	1	1	1	1	1	0	0	1	1	1		0	1	1		0	1	1
	1	1	1	1	1	0	0	1	1	1		0	1	1		0	0	0
X	1	1	1	1	0	1	1	1	0	0		1	1	1		1	1	1
X	1	1	1	1	0	1	1	1	0	0		1	1	1		1	1	0
X	1	1	1	1	0	1	1	1	0	0		0	1	1		0	1	1
	1	1	1	1	0	0	1	1	0	0		0	1	1		0	0	0
	1	0	0	1	1	0	1	0	0	1		1	0	0		1	1	1
	1	0	0	1	1	0	1	0	0	1		1	0	0		1	1	0
X	1	0	0	1	1	1	1	0	0	1		0	1	0		0	1	1
	1	0	0	1	1	0	1	0	0	1		0	1	0		0	0	0
	1	0	0	0	0	0	1	0	0	0		1	0	0		1	1	1
	1	0	0	0	0	0	1	0	0	0		1	0	0		1	1	0
	1	0	0	0	0	0	1	0	0	0		0	1	0		0	1	1
	1	0	0	0	0	0	1	0	0	0		0	1	0		0	0	0
	0	0	1	1	1	0	0	1	1	1		1	1	1		1	1	1
	0	0	1	1	1	0	0	1	1	1		1	1	1		1	1	0
	0	0	1	1	1	0	0	1	1	1		0	1	1		0	1	1
	0	0	1	1	1	0	0	1	1	1		0	1	1		0	0	0
	0	0	1	0	0	0	1	1	0	0		1	1	1		1	1	1
	0	0	1	0	0	0	1	1	0	0		1	1	1		1	1	0
	0	0	1	0	0	0	1	1	0	0		0	1	1		0	1	1
	0	0	1	0	0	0	1	1	0	0		0	1	1		0	0	0
	0	0	0	1	1	0	1	0	0	1		1	0	0		1	1	1
	0	0	0	1	1	0	1	0	0	1		1	0	0		1	1	0
X	0	0	0	1	1	1	1	0	0	1		0	1	0		0	1	1
	0	0	0	1	1	0	1	0	0	1		0	1	0		0	0	0
	0	0	0	0	0	0	1	0	0	0		1	0	0		1	1	1
	0	0	0	0	0	0	1	0	0	0		1	0	0		1	1	0
	0	0	0	0	0	0	1	0	0	0		0	1	0		0	1	1
	0	0	0	0	0	0	1	0	0	0		0	1	0		0	0	0

TRUTH-TABLE FOR MANUFACTURER PROBLEM

FIGURE I.5.

I. I.3 Tautologies

A logical expression which has the final column of its truth-table consisting only of 1's is said to be a logical truth or a tautology. A tautology does not convey any new information since it is true irrespective of the truth or falsity of its component propositions. An example of a tautology is :

$$(A \supset B) \equiv \left[(\sim A) \vee B \right]$$

1	1	1	1	0	1	1	1
1	0	0	1	0	1	0	0
0	1	1	1	1	0	1	1
0	1	0	1	1	0	1	0

TRUTH-TABLE OF A TAUTOLOGY

FIGURE I.6

The following tautologies are all important laws in symbolic logic:

$A \equiv A$	Law of identity	(1)
$\sim (A.\sim A)$	Law of contradiction	(2)
$A \vee \sim A$	Law of excluded middle	(3)
$\sim \sim A \equiv A$	Law of double negation	(4)
$(A.B) \equiv (B.A)$	Associative laws	(5)
$(A \vee B) \equiv (B \vee A)$		(6)
$A.(B.C) \equiv (A.B).C$	Commutative laws	(7)
$A \vee (B \vee C) \equiv (A \vee B) \vee C$		(8)
$A.(B \vee C) \equiv (A.B) \vee (A.C)$	Distributive laws	(9)
$A \vee (B.C) \equiv (A \vee B).(A \vee C)$		(10)
$\sim (A.B) \equiv \sim A \vee \sim B$	De Morgan's relations	(11)
$\sim (A \vee B) \equiv \sim A. \sim B$		(12)
$(A \supset B) \equiv (\sim A) \vee B$		(13)

Bearing in mind the linguistic interpretation of symbolic logic these laws are self-evident. However, disregarding any interpretation, their validity can be checked within the system of logic by constructing their truth-tables and observing that they are all tautologies.

Laws (1),(2) and (3) are what Boole called the three "Fundamental Laws of Thought".

De Morgan's relations are interesting, since, using them together with (4), it is found that AND is expressible in terms of NOT and OR while OR is expressible in terms of NOT and AND.
i.e.

$$A.B \equiv \sim(\sim A \vee \sim B)$$
$$A \vee B \equiv \sim(\sim A. \sim B)$$

The truth-table for a logical function of n propositional variables has 2^n rows. Since the final column of the truth-table may have either unity or zero in each position there are $2^{(2^n)}$ possible truth-tables for n variables. This means that there are $2^{(2^n)}$ possible logical functions of n propositional variables. In the case of two variables there are 16 possible functions, of which only five were given on p.22 The truth-tables for all the 16 functions of two logical variables are given in Figure I.7 . Some of the functions are trivial.

A	B	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

THE 16 FUNCTIONS OF TWO LOGICAL VARIABLES

FIGURE I.7

The tautologies listed on p. 25 may be used to simplify logical expressions. To illustrate the method of doing this the manufacturer problem given earlier is solved by this method in Appendix 1. The solution is achieved by continued application of various of the tautologies (1) to (13) . Although this method of solution is not too difficult for this simple problem it is quite unsuitable for solving more complicated problems involving large numbers of variables.

II. SOME EXISTING LOGICAL MACHINES

II.1 Jevons' Mechanical Machine

In 1870 a paper by W.S. Jevons entitled "On the Mechanical Performance of Logical Inference" appeared in the Philosophical Transactions of the Royal Society (ref: 8). The paper described an ingenious mechanical machine constructed rather like a small piano. The machine could solve logical problems expressed in the form of an IDENTITY and involving not more than four propositional variables. On the face of the machine were 16 wooden slats each marked with a different assignment of truth and falsity of four logical propositions. A large number of pins protruded from the back of the slats. The machine had a keyboard and when a key was depressed levers engaged with certain of the pins and caused certain slats to be moved so that their faces disappeared from view. The pins on the back of the slats were cleverly arranged so that all the slats marked with assignments for which a chosen proposition was true (or false if desired) were moved by depressing the appropriate key. The faces of all those slats then disappeared from view. Other keys corresponded to the logical connectives OR and IDENTITY. To solve a problem it was first necessary to express the problem in the required form of an IDENTITY with the right hand and left hand sides containing only the connectives AND, OR and NOT. The corresponding keys were then pressed in the order in which the propositional variables and connectives occurred in the IDENTITY.

This was probably the first machine capable of solving a problem faster than it could be solved manually. The machine constructed all the rows of the truth-table of any problem <u>simultaneously</u>. If machines were built on this principle to handle larger problems the amount of equipment required would be proportional to the number of rows in the truth-tables of the problems. Such machines can therefore only be built to solve simple problems involving few variables.

II.2 The Kalin-Burkhart Machine

In 1947 T.A. Kalin and W. Burkhart constructed the first electrically operated logical computer at Harvard University. This was a relay operated machine capable of solving problems involving up to twelve variables. The logical constraints were set up within the machine by setting the position of numerous rotary switches. When set into operation the machine scanned exhaustively through the assignments of truth-values of the variables at a rate of about two per second. The machine stopped at any assignment of truth-values satisfying all the logical constraints. By pressing a button the machine was set into operation again, continuing its scan until it found any other solutions.

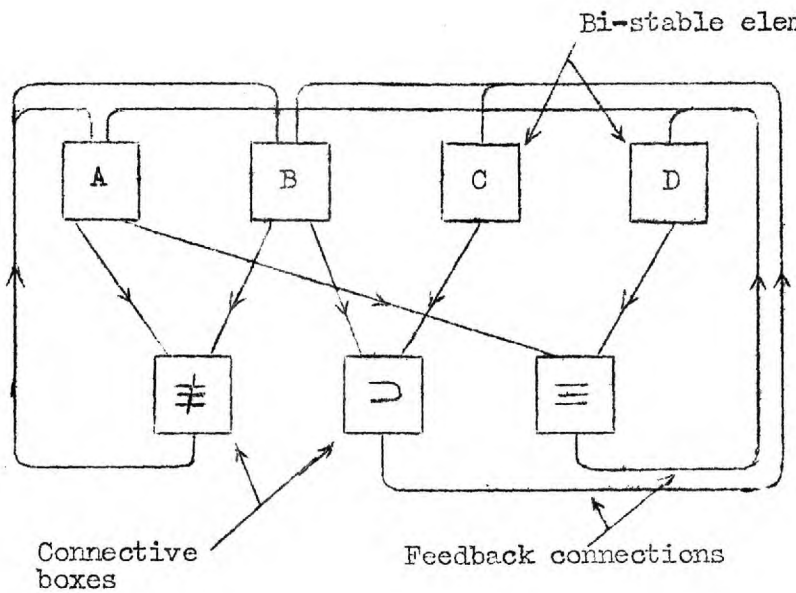
II.3 The McCallum-Smith Scanning Machine

In 1950 D. M. McCallum and J.B. Smith constructed independently, a scanning machine at the Edinburgh laboratories of Ferranti Ltd.. This machine handled problems in up to seven logical variables. The logical constraints were set up by interconnecting logical connective boxes (relay operated), arranged on a plug board. This machine was constructed of relays and uniselectors and, like the Kalin-Burkhart machine, it made an exhaustive scan of the truth-value assignments to the variables. A description of this machine may be obtained from ref: 5.

II.4 The McCallum-Smith Feedback Computer

Realizing the limitations of a machine working on the exhaustive scan principle McCallum and Smith constructed what they called a feedback logical computer. This machine could find one solution extremely quickly for a certain class of problems. The feedback machine had a plugboard arrangement, similar to that of the exhaustive scan machine, on which connective boxes were connected in accordance with the logical constraints. Problems involving up to four propositional variables could be put on this machine. The truth status of each variable was represented by the state

of a bi-stable element to which feedback connections were taken from the output of all rules (connective boxes) containing that variable. This is illustrated in Figure II.1 in which the boxes containing the letters A, B, C and D represent the bi-stable elements whose output wires are held at a positive potential to indicate truth and at earth potential to indicate falsity.



CONNECTION OF THE FEEDBACK COMPUTER TO SOLVE A SIMPLE PROBLEM WITH THREE RULES

FIGURE II.1.

The output of a logical connective box is similarly held at a positive potential when the connective is satisfied and at earth potential otherwise.

The feedback machine performed a non-deterministic scan according to the following rules :

(i) If a constraint was satisfied by the existing assignment of truth-values signals were fed back to all the variables involved in it,

(ii) Any variable receiving signals on all its feedback connections was held in its existing state and was not allowed to change its state in the following transition (from the existing state of the machine to the next),

(iii) All variables with at least one feedback connection without a signal were given equal probabilities of changing state in the next transition of the machine, and

(iv) Only one variable was allowed to change state between the existing state of the machine and the next state of the machine, i.e. only single transitions were allowed.

The only states in which the machine was stable under these conditions were those in which the assignment of truth-values corresponded to a solution. The non-deterministic scan of this machine corresponded to a discrete Markoff process which is a form of stochastic process. When the machine was set into operating starting with an arbitrary assignment of truth-values it tended to find a solution differing only little from this (if such a solution existed). However, in the process of finding the solution the machine often considered some assignments of truth-values several times. In terms of the problem space concept this means that, while searching in the problem space for a solution, the machine tried some points on several occasions. In their paper describing the feedback machine (ref: 6) McCallum and Smith suggest that, in order to find as many solutions to a problem as possible, the machine be allowed to make a series of searches in the problem space from a set of "uniformly spaced" starting points. The serious disadvantage of this machine is that, even after allowing the machine to make many searches, it is impossible to tell whether many solutions remain to be found. If the machine were given a problem having no solutions and were allowed to search the problem space for a long time it would be impossible to decide whether the problem had no solutions or whether it had, but the machine had not yet found any.

Consider the following three constraints :

1. $(A \vee B) \equiv C \supset A.F$
2. $(A \supset C.F) \vee E$
3. $(D \equiv E) \supset G$

Constraints 1 and 2 have three variables in common. These constraints are said to overlap. Constraints having a high proportion of their variables in common are said to have a high degree of overlap. Thus, constraints 1 and 2 have a high degree of overlap while constraints 2 and 3 have a low degree of overlap. Constraints 1 and 3 do not overlap at all. Constraints not overlapping are independent.

McCallum and Smith found that the feedback logical computer was very advantageous in comparison with an exhaustive scan machine only for the class of problemsin which the logical constraints had a low degree of overlap and in which each constraint did not involve many variables. The reason for this is easily seen. Suppose that, in the course of its search, the machine has arrived at a state where it is necessary to change the state of only one logical variable to reach a solution, i.e. the machine is one single transition away from a solution. The one variable requiring to be changed might be involved in, say, one or two unsatisfied logical constraints together with numerous other logical variables. All these variables are then given equal probabilities of changing state in the next transition. The machine might therefore have a higher probability of moving further away from the solution than moving towards it.

McCallum and Smith suggested that this weakness in the machine's strategy might be overcome by altering the operating principles so that the transition probabilities assigned to the logical variables for a given state of the machine are weighted in proportion to the numbers of unsatisfied constraints that the respective variables are involved in.

II.5 A Feedback Computer with Weighted Transition Probabilities

In a sense this is out of place in a Section headed, "Some Existing Logical Machines" since no machine has been built to work according to the ideas discussed here.

The first part of the work carried out here, and forming the subject of this thesis, consisted in investigating the idea of McCallum and Smith of having a feedback computer with weighted transition probabilities. An arbitrary problem with five propositional variables was considered and an assignment of truth-values was chosen randomly as a starting point. These truth-values were substituted in the logical constraints to find which of these were satisfied and which were broken. Each variable was then assigned a probability proportional to the number of broken constraints in which it appeared. The periphery of a roulette wheel was subdivided to give the resulting set of probabilities and the wheel was spun. When the wheel had stopped its position indicated the logical variable whose state was to be changed in the next transition. The state of the nominated variable was changed, the new assignment of truth-values was substituted in the logical constraints and the next set of transition probabilities was obtained. This was repeated until a solution to the problem was found. This procedure was repeated from many arbitrarily chosen starting points and the average number of transitions required to reach a solution was calculated. The experiment was then performed using other problems.

Using the same problems the experiments were repeated using equal transition probabilities instead of weighted probabilities. The average numbers of transitions required in this casewere compared with figures obtained using weighted probabilities. In some cases the weighted probabilities were found to give an advantage but for other problems the average number of transitions required to reach a solution was greater than that when using equi-probable transitions.

It was soon realized that weighting the probabilities in this way could not be expected to have the desired effect because very often the logical expression of a problem contains redundant information. Suppose that a given problem has a set, S , of solutions. Suppose that the set, S , is a subset of the set, S' , of solutions to another logical constraint, C . If the constraint, C , were taken as an additional rule to the given problem it would be completely redundant. The problem with the additional constraint, C , would have the same set, S , of solutions as the given problem.

If a problem containing a redundant rule were solved by the weighted transition probability technique completely different weighting of probabilities would result if the redundant rule were removed from the problem. Yet the problem would be essentially unaltered. For this reason the proposed method of weighting the transition probabilities was a poor one.

Some time was spent trying to think of some measure function (say, for example, the number of satisfied rules), that could be used with a machine making a search through a problem space to convey information to the machine regarding the "distance" of the point in the space being considered by the machine from a solution. No measure function of this kind could be found and it was felt very strongly, though not proved, that no such measure function was possible for problems formulated in terms of two-valued logic.

III. SOME IDEAS ON SOLVING PROBLEMS IN MANY-VALUED LOGIC

III.1 Generalization from Two-Valued to Many-Valued Logic.

In Section I two-valued logic was introduced by defining, by means of truth-tables, a number of logical connectives which could be used for combining elementary propositions to obtain complex propositions. The truth-value of a proposition was defined to be unity if the proposition were true and zero if it were false.

This may be quite easily extended to define an m-valued logic in which the truth-values attached to propositions may have any integral value in the range 0 to m-1 . An m-valued logical connective relating two variables can then be defined in the form of an m x m matrix. The truth-value of one of the operand variables determines the relevant column of the matrix while the truth-value of the other operand variable determines the relevant row. The entry common to the relevant row and column gives the resultant truth-value of the complex proposition consisting of the two elementary propositions joined by the connective defined by the matrix.

Whereas, in two-valued logic, there are 16 possible logical functions of two variables, in an m-valued system there are $m^{(m^2)}$; each of the m^2 entries in the defining matrix having any value in the range 0 to m-1. There are, therefore, a vast number of connectives in many-valued logic of which relatively few are used or have any useful interpretation. An interesting paper on this subject was published by B. Rosser in the "American Journal of Physics", (ref: 3).

Various interpretations for many-valued logics have been proposed involving probability and plausibility. For example, in a four-valued system, the truth-values 0, 1, 2 and 3 might represent falsity.

improbability, probability and truth respectively.

In a system of m-valued logic a value, S , is chosen such that $0 < S \leqslant m$. Truth-values 0 to S-1 are said to be undesignated while the truth-values S to m are said to be designated. A many-valued proposition is then assertable if its truth-value is designated under all possible assignments of truth-values to the elementary propositions involved in it. This is equivalent to the definition of a tautology in two-valued logic.

III.2 The Connectives of Post, Lukasiewicz and Tarski

The systems proposed by E.L. Post, J. Lukasiewicz and A. Tarski all have the same AND and OR connectives. The truth-tables for these are shown in Figure III.1 . For economy of space they are shown for a four-valued system.

$A \cap B$	3	2	1	0	B
3	3	2	1	0	
2	2	2	1	0	
1	1	1	1	0	
0	0	0	0	0	
A					

AND

$A \cup B$	3	2	1	0	B
3	3	3	3	3	
2	3	2	2	2	
1	3	2	1	1	
0	3	2	1	0	
A					

OR

POST CONNECTIVES

FIGURE III.1.

If a and b are the truth-values of propositional variables A and B , then the truth-value of $A \cap B$ is defined as MIN.(a,b) , while the truth-value of $A \cup B$ is defined as MAX.(a,b).

When a many-valued connective is sought that is equivalent to the NOT of two-valued logic it is found that no single many-valued connective

will suffice. Several many-valued connectives are used for this purpose, each doing part of the work done by NOT in two-valued logic. The connectives used by Post, Lukasiewicz and Tarski in this connection appear in Figure III.2 (for four-valued logic).

A	$-A$	A'	A''	A'''	A^{IV}	$J_0(A)$	$J_1(A)$	$J_2(A)$	$J_3(A)$
3	0	2	1	0	3	0	0	0	3
2	1	1	0	3	2	0	0	3	0
1	2	0	3	2	1	0	3	0	0
0	3	3	2	1	0	3	0	0	0

FIGURE III.2.

The first of these connectives is called NEGATION. The NEGATION of A is denoted by $-A$. If "a" is the truth-value of A, then in an m-valued system, the truth-value of $-A$ is m-1-a (m=4 for Fig. III.2). This connective plays the role played by NOT in two-valued logic in such principles as:

$$A.B \equiv \sim(\sim A \vee \sim B)$$
$$\sim\sim A \equiv A$$
$$A \supset B \equiv (\sim B \supset \sim A).$$

In two-valued logic a double negative yields a positive. In Post's system use is made of an equivalent connective in m-valued logic such that, when applied to a proposition m times in succession, the truth-value of the proposition remains unchanged. Post calls this connective CYCLIC NEGATION since it "cycles" the values in the truth-table, as seen in Figure III.2 . The result of operating on A with the CYCLIC NEGATION connective is denoted by A' and the result of repeating the operation is denoted by A" . It is seen that m successive applications (four in the case of Figure III.2) of this connective leave the truth-value of a proposition unchanged.

There is yet another role played by NOT in two-valued logic and that is, NOT-A is assertable when, and only when, A takes the status "false" . In order to have many-valued connectives that will play this role the following J-FUNCTIONS are introduced:

$J_{m-1}(A)$, having truth-value m when, and only when, A has truth-value m-1.

$J_{m-2}(A)$, having truth-value m when, and only when, A has truth-value m-2 .

⋮

$J_0(A)$, having truth-value m when, and only when, A has truth-value zero.

The truth-tables for $J_0(A)$, $J_1(A)$, $J_2(A)$ and $J_3(A)$ for four-valued logic are given in Figure III.2 .

$J_0(A) \cup J_1(A) \cup J_2(A) \cup J_3(A)$ has the truth-value 3 for all truth-values of A . This is a generalization of the two-valued principle, $A \vee \sim A$.

III.3 The Connectives of Lewis.

The system proposed by C.I. Lewis uses the same NEGATION (–) , as defined in Figure III.2. However, the AND and OR connectives used by Lewis are different and these appear in Figure III.3 . For reasons similar to those given above, Lewis also introduces a set of J-FUNCTIONS which, for four-valued logic, also appear in Figure III.3 .

A ∩ B	3	2	1	0	B
3	3	2	1	0	
2	2	2	0	0	
1	1	0	1	0	
0	0	0	0	0	
A					

AND

A ∪ B	3	2	1	0	B
3	3	3	3	3	
2	3	2	3	2	
1	3	3	1	1	
0	3	2	1	0	
A					

OR

A	$J_3(A)$	$J_2(A)$	$J_1(A)$	$J_0(A)$
3	3	0	0	0
2	0	2	1	0
1	0	1	2	0
0	0	0	0	3

J-FUNCTIONS

LEWIS CONNECTIVES

FIGURE III.3.

III.4 Expression of Problems in Terms of Many-Valued Logic.

In 1955 the following railway timetable problem appeared in an issue of the Manchester Guardian:

The down line from Denver to Milwaukee West is shown (Fig. III.4) accompanied by the timetable for the afternoon. The names which are underlined are names of stations, at which passenger trains stop, while the remainder are names of signal boxes. All times shown are departure times except those at Milwaukee West which are times of arrival.

Three extra trains must be fitted into the timetable:

(1) A fast passenger train leaving Denver at 3.55, travelling at 60 m.p.h. over the $11\frac{1}{2}$ miles to Milwaukee West.

(2) A stopping train, stopping at each station for one minute. This must leave Denver ten minutes after the express and must never exceed 40 m.p.h..

(3) A goods train, leaving Denver at 3.30, which must not exceed a speed of 18 m.p.h..

The express train must not be interfered with and the stopping train has priority over the goods train. However, the goods train must get to Milwaukee West as early as possible. To facilitate working of the line there are three loop lines on to which a slow train may be diverted to let a faster train pass. The goods train should be allowed two minutes for starting and for stopping and allowance should also be made for goods trains being as much as two minutes off time at any point.

Distance in $\frac{1}{4}$-mile units	Place	Trains 1	2	3	4	5
		G	G	G	G	P
0	DENVER	3.13		3.28	4.06	4.19
4	ALBANY					
7	CHARLESTON		3.28		4.27	
9	SEATTLE					
13	MONTPELIER					4.28
15	MONTR.SIDINGS			3.41		
19	LITTLE ROCK					
22	TALLAHASSEE	3.36	3.42		4.42	4.33
26	CLEVELAND					
33	DETROIT					
37	OKLAHOMA CITY					
41	NEW ORLEANS					4.41
45	MILWAUKEE					
46	MILWAUKEE WEST	4.01				4.46



G - GOODS; P - PASSENGER.

FIGURE III.4.

The following ideas on stating this problem in terms of many-valued logic are based on ideas expressed by A. Rose in a letter to Professor A. Porter.

Since the time in this problem ranges from 3.13 p.m. to 4.46 p.m., i.e. a range of 93 minutes, the problem can be expressed in terms of a 94-valued logic. Truth-values 0, 1, 2, ---93 correspond to times 3.13, 3.14, 3.15, --- 4.46 respectively.

If x, y are the truth-values of the many-valued propositions X, Y, then the following connectives are defined:

(i) $X \cap Y$, having truth-value MIN.(x,y) .
This is Post's AND connective.

(ii) $X * Y$, having truth-value MIN.(93, x + y) .

(iii) $X \rightarrow Y$, having truth-value $\begin{cases} 93 \text{ when } x \leqslant y \ , \\ 0 \text{ when } x > y \ . \end{cases}$

(iv) $\neg X$, having truth-value $\begin{cases} 0 \text{ when } x = 93, \\ 93 \text{ when } x \neq 93. \end{cases}$

This connective could be expressed in terms of Post connectives as $-J_{93}(X)$.

The truth-value 93 will be the only designated one in this system.

The information given in the problem is now translated into logical formulae. The express train fits into the timetable without any trouble.

Let the trains be numbered from 1 to 5 in the given order and let the express train be train 6, the stopping train be train 7 and the goods train be train 8 . Let the stations and signal boxes be numbered from 1 to 14 in the given order. Let the time at which train i leaves station j be represented by the truth-value of a proposition $T_{i,j}(i=1,---,6;\ j=1,---,14)$, and let the time at which train 7 leaves station j be represented by the truth-value of a proposition $P_j(j=1,---,14)$. The values of $T_{i,j}$ are all known, e.g., train 2

leaves signal box 3 at 3.28 p.m., which, in truth-value notation, is 15. Hence, $T_{2,3}$ has truth-value 15 .

Let the truth-value of proposition $X_i (i=1,---,13)$, be the time taken by train 7 to get from station i to station i+1 . Propositions $P_1,---,P_{14}$ may then be expressed as follows:

$$\left.\begin{aligned}
P_1 &\equiv 52 \\
P_2 &\equiv 52 * X_1 \\
P_3 &\equiv 52 * X_1 * X_2 \\
P_4 &\equiv 52 * X_1 * X_2 * X_3 \\
P_5 &\equiv 53 * X_1 * X_2 * X_3 * X_4 \quad \text{(allowing 1m. stop.)} \\
P_6 &\equiv 53 * X_1 * X_2 * X_3 * X_4 * X_5 \\
&\vdots \\
P_{14} &\equiv 55 * X_1 * ---------- * X_{13}
\end{aligned}\right\} \quad (1)$$

Since train 7 must not exceed 40 m.p.h., minimum values may be calculated for $X_1 ,---,X_{13}$.

Expressions must now be constructed which will ensure that overtaking shall occur only where a loop line is available. Consider the expression $(T_{i,j} \rightarrow P_j)$. This is true (has truth-value 93) when train i leaves station j before train 7 . Similarly, the proposition $(P_{j+1} \rightarrow T_{i,j+1})$ is true when train 7 leaves station j+1 before train i . If both these expressions are true, then train 7 overtakes train i between stations j and j+1 . This is only permitted for j=2,6,9 . By examining the timetable given it is evident that the only trains that might be overtaken by train 7 are trains 4 and 5 . Hence, the expressions required to ensure that trains do not have to overtake at the wrong places are:

$$\left.\begin{aligned}
&\neg\{(T_{i,j} \rightarrow P_j) \cap (P_{j+1} \rightarrow T_{i,j+1})\} \\
&\neg\{(P_j \rightarrow T_{i,j}) \cap (T_{i,j+1} \rightarrow P_{j+1})\}
\end{aligned}\right\} \quad \begin{array}{l} i= 4,5 \ ; \\ j= 1,3,4,5, \\ \quad 7,8,10,11,12,13) \end{array} \quad (2)$$

In these expressions the $\neg$ connective acts as a NOT, since $\neg X$ has truth-value zero when X has truth-value 93 .

Additional expressions are required to allow for the fact that goods trains are liable to be as much as two minutes off time at any point. Assignments of truth-values to the propositions $P_1,---, P_{14}$ must then be found which satisfy all the constraint expressions and for which $\sum_{i=1}^{13} X_i$ is as small as possible consistent with the minimum values previously calculated for $X_1,---,X_{13}$.

The whole procedure is then repeated to fit train 8 into the timetable.

This does not appear to be a practicable way of solving a problem of this nature since it is not an easy matter to find an assignment of truth-values to $P_1,---,P_{14}$ which satisfy all the constraints and for which $\sum_{i=1}^{13} X_i$ is minimized. However, this does indicate one way in which problems involving relationships between continuous variables and also logical constraints can be expressed in terms of many-valued logical propositions.

In a paper published in the "Proceedings of the Cambridge Philosophical Society", Rose (ref: 2 .) shows how the problem of constructing an academic timetable can be expressed in terms of many-valued logic. In this formulation the many-valued logical variable $X_{i,j,k}$ represents the proposition, "class i is taken by lecturer j , and class i has been taken previously by lecturer j during k-1 periods in the week". If class i is taken by lecturer j for the k^{th} time during the period numbered p , then the truth-value of the proposition $X_{i,j,k}$ is taken to be p .

III.5 Reduction of Many-Valued Logic to Two-Valued Logic

Let $F(A,B,---,P)$ be a logical function of the propositional variables $A,B,---,P$ in an m-valued system of logic. Consider the case where m is an integral power of two,

$$\text{i.e.} \quad m = 2^N \qquad (1)$$

Let the truth-values of the propositions A, B, ----,P be represented by a,b, ----, p respectively, where these may have integral values in the range 0 to m-1 . Let f(a,b,----,p) be the truth-value function giving the truth-value of F(A,B,----,P) for any given assignment of values a,b,----,p .

e.g. if $F(A,B) \equiv A \cap B$

then, $f(a,b) \equiv \text{MIN.}(a,b)$.

Now, let "a" $(0 \leqslant a \leqslant m-1)$. be expressed as a binary number in the form

$$a = a_1 + a_2 . 2 + a_3 . 2^2 + \text{--------} + a_N . 2^{N-1}$$

i.e.
$$a = \sum_{i=1}^{N} a_i . 2^{i-1} \qquad (2)$$

The truth-value, a, may therefore be represented by the ordered set of binary digits $\langle a_N \text{ ----- } a_2 a_1 \rangle$, the binary number formed by this ordered set giving the truth-value of the many-valued proposition A . Similarly, b,----,p , the truth-values of B,----,P , may be represented by the ordered sets of binary digits $\langle b_N \text{----} b_2 b_1 \rangle$,----- , $\langle p_N \text{----} p_2 p_1 \rangle$ respectively.

The truth-value of F(A,B,----,P) may also be represented by an ordered set of binary digits $\langle f_N \text{----} f_2 f_1 \rangle$ so that

$$f = \sum_{i=1}^{N} f_i . 2^{i-1} \qquad (3)$$

It is then possible to express the $f_i (i = 1,2,\text{----},N)$ in the form

$$f_i = f_i(a_1, a_2, \text{---}, a_N, b_1, b_2, \text{---}, b_N, \text{-----}, p_1, p_2, \text{---}, p_N) \qquad (4)$$

Let the $a_i, b_i, \text{---}, p_i$ and $f_i (i = 1,2,\text{---},N)$, be thought of as being the truth-values (0,1) of two-valued propositions $A_i, B_i, \text{---}, P_i$

and $F_i (i = 1,2,---,N)$, respectively. The m-valued system of logic may now be replaced by two-valued logic by expressing all the many-valued truth-value variables in binary form and operating upon the binary truth-value variables with two-valued logical functions.

This is demonstrated by expressing some of the four-valued connectives given in Sections III.2 and III.3 in terms of two-valued connectives. Consider first the four-valued NEGATION connective (–) defined by the truth-table in Figure IV.2 (p. 36). This is redrawn below with the four-valued proposition A replaced by two-valued propositions A_1 and A_2. The four-valued connective F(A) is replaced by two two-valued logical functions F_1 and F_2.

A		F(A)	
A_2	A_1	F_2	F_1
1	1	0	0
1	0	0	1
0	1	1	0
0	0	1	1

$$\left.\begin{aligned} F_1 &\equiv \sim A_1 \\ F_2 &\equiv \sim A_2 \end{aligned}\right\} \qquad (5)$$

FIGURE III.5

In Figure IV.5 the binary digits under A_2 and A_1 form the binary expression of the truth-value of A while the binary digits under F_2 and F_1 form the binary expression of the truth-value of F(A).

The AND connective of Post is defined in Figure III.1 (p. 35) and the AND connective used by Lewis is defined in Figure III.3 (p. 38).

These are given below in binary form.

F_2F_1 \ A_2A_1	11	10	01	00	B_2B_1
1 1	11	10	01	00	
1 0	10	10	01	00	
0 1	01	01	01	00	
0 0	00	00	00	00	

$F(A,B) \equiv A \cap B$.

POST AND CONNECTIVE

F_2F_1 \ A_2A_1	11	10	01	00	B_2B_1
1 1	11	10	01	00	
1 0	10	10	00	00	
0 1	01	00	01	00	
0 0	00	00	00	00	

$F(A,B) \equiv A \cap B$.

LEWIS AND CONNECTIVE

FIGURE III.6

The pairs of digits forming the entries of these matrices are the truth-values of two-valued propositions F_2, F_1 . These propositions are expressed below in terms of the two valued propositions A_2, A_1, B_2, B_1 and two-valued logical connectives.

The expressions for the Post AND connective are:

$$\left.\begin{aligned} F_1 &\equiv A_1 \cdot B_1 \vee (\sim A_1 \cdot A_2 \cdot B_1 \cdot \sim B_2) \vee (A_1 \cdot \sim A_2 \cdot \sim B_1 \cdot B_2) \\ F_2 &\equiv A_2 \cdot B_2 \end{aligned}\right\} \quad (6)$$

The expressions for the Lewis AND connective are:

$$\left.\begin{aligned} F_1 &\equiv A_1 \cdot B_1 \\ F_2 &\equiv A_2 \cdot B_2 \end{aligned}\right\} \quad (7)$$

The Lewis AND connective is therefore obtained simply by operating upon the first digits of the binary representation of the four-valued truth-values with the two-valued AND connective and operating upon the second digits of the binary representation of the four-valued truth-values in the same way.

Any logical functions in any system of many-valued logic can be expressed in terms of two-valued propositions and connectives in this way.

III.6 Problem Spaces and Measure Functions

Before considering what type of problem space might be applicable to problems in many-valued logic it is worth examining more closely the structure of the hypercubic spaces already discussed in connection with two-valued logic. Figure III.7 illustrates how two n-dimensional spaces are combined to form a space of n+1 dimensions. Two points are connected to form a line. Corresponding points of two lines are connected to form a square. Corresponding points of two squares are connected to form a cube. Similarly, corresponding points of two n-dimensional hypercubes are connected to form a hypercube of n+1 dimensions.

These spaces have been used as models when thinking about methods of solving problems in two-valued logic. If a logical computer for solving two-valued problems is restricted to making single transitions in which the truth-value of only one logical variable is changed, then the edges of the hypercube represent all the transitions that it is possible for the machine to make. Still restricting attention to problems in two-valued logic, suppose that a given problem has only one solution and that the hypercube problem space is orientated so as to bring the point corresponding to the solution to the top of the hypercube. From Figure III.7 it is seen that the minimum number of single transitions required to get from any point in an n-dimensional space to the solution (at the uppermost point) lies in the range 0 to n . Further, it is seen that, when drawing a two-dimensional representation of the space, the points may be arranged to occupy n+1 levels with all the points at any level being the same minimum number of transitions away from the

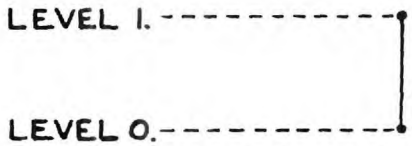
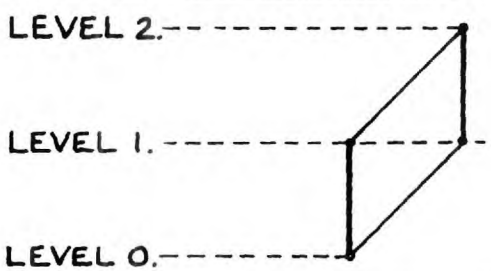
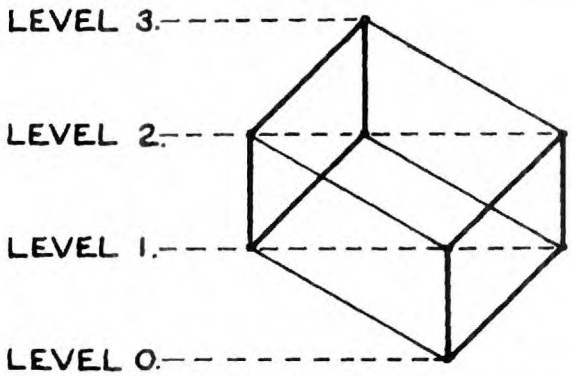
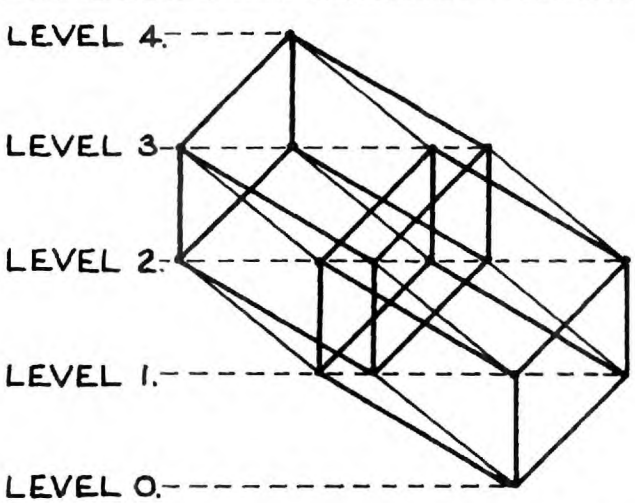
n.	AN (n+1)-DIMENSIONAL SPACE IS OBTAINED BY JOINING ALL CORRESPONDING POINTS OF TWO n-DIMENSIONAL SPACES.
1	
2	
3	
4	

FIGURE III.7.

solution, e.g., at least n transitions are required from level 0, at least n-1 transitions are required from level 1, and so on.

As mentioned in the Introduction, it would be extremely useful if a measure function could be found which could be evaluated at points in the problem space through which a computer was searching and which would convey information to the computer regarding the likelihood of finding a solution in the neighbourhood in the space in which the computer was searching. Information might also be conveyed to the computer as to the best "direction" in which to continue the search for a solution. What properties should such a measure function possess?

When the problem space for a two-valued problem with a single solution is orientated to bring the solution to the upper-most point, all the points at any level in the space are equi-distant from the solution in the sense that they are symmetrically placed relative to the solution and are all the same minimum number of single transitions away from the solution. In addition, a given point in the space is nearer to the solution than any point at a lower level in the space. An ideal measure function would therefore have the following two properties, the solution being at the upper-most point in the problem space:

(i) the measure function has the same value at all points at a given level in the space, and

(ii) the value of the measure function at any given level in the space is higher than the value of the function at any lower level, i.e., the measure function has a minimum value at the lowest point in the problem space and a maximum value at the highest point (the solution point).

There is one obvious function possessing these properties. This function is obtained by collating (or comparing) the assignment of truth-values to the variables at a given point in the space with

the assignment of truth-values constituting the solution to the problem and counting the number of places in which they agree.

Example: suppose that it is required that this function be evaluated at a point in the problem space at which the assignment of truth-values to the variables, say, A,B,C,D, is 1011 while the solution to the problem is the assignment 1100. The collation of these two assignments of truth-values will be a four digit number containing unity in all positions in which the two assignments have the same value and zero in all positions in which the two assignments differ in value. The collation of the assignments 1011 and 1100 is therefore 1000. The value of the function at the given point is given by the number of ones in this collation and is therefore unity.

This measure function will have the value zero at the lowest point in the space and the value n , for an n variable problem, at the solution point, at the top of the space. This function also satisfies the requirement of having the same value at all points at any given level in the problem space. However, since the evaluation of this function requires a prior knowledge of the solution being sought it is of no practical use.

So far only problems having a single solution have been considered. In general, a problem will have a whole set of solutions. The hypercube problem space can be orientated to bring only one solution to the upper-most point and the level of any given point in the space will depend upon which of the solutions has been brought to the top. How then, can a measure function be found? It might be possible to find some measure function that would have a high value at points in the space inclose proximity to a solution and a low value at points in the space remote from solutions.

It appears unlikely that any ideas will result from a study of the properties of problem spaces. It would be more useful if another space could be found that would be a homomorphism of the problem space such that all the solutions to a problem are mapped on

to a single point. One could then speak of the distance of a point in this new space from the solution point even though the problem considered may have numerous solutions. Such a space does exist. It is the truth-value space in which each different truth-value in the system of logic in question is represented by a unique point. The truth-value of a problem corresponding to any given assignment of truth-values to the logical variables can be found by substituting the assignment of truth-values in the logical constraints constituting the problem. The truth-value of a problem can be found in this way at all points in the problem space. All the points in the problem space can then be mapped on to the points in the truth-value space.

In the case of two-valued logic the result is rather disappointing since the truth-value space consists merely of two points corresponding to the truth-values unity and zero.



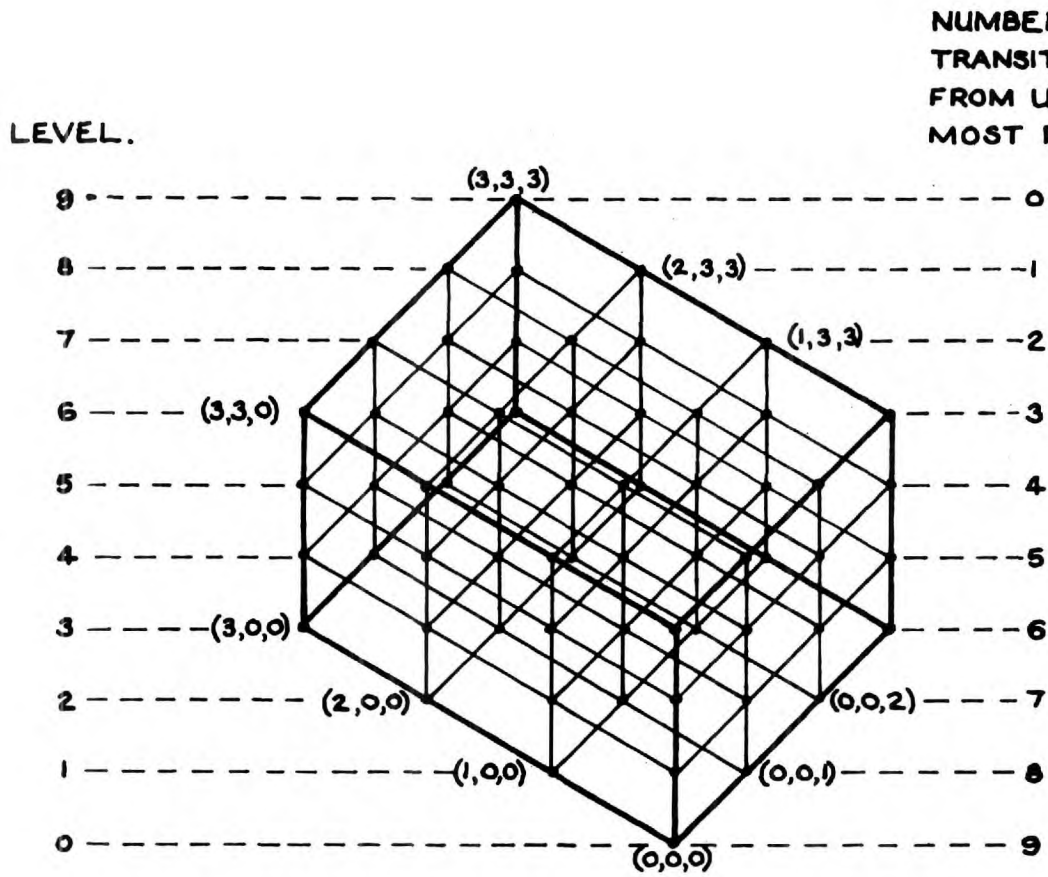
TRUTH-VALUE SPACE
FOR 2-VALUED LOGIC.

FIGURE III.8

All the solutions to a problem are mapped on to the upper point in this truth-value space while all the non-solutions are mapped on to the lower point. There is no sense of "distance" in this space; a given point in the space either is the solution point or is not.

However, consider now what kind of structure might be suitable as a problem space for a system of many-valued logic.

One possible space for n variable problems in m-valued logic would be that obtained by introducing a metric and dividing the edges of an n-dimensional hypercube into $m-1$ equal parts. Each edge would then contain m equally spaced points corresponding to the m truth-values of one of the logical variables. The hypercube has then been divided into $(m-1)^n$ sub-hypercubes. A problem space of this



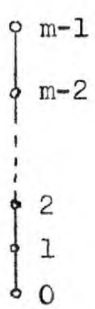
PROBLEM SPACE FOR A PROBLEM IN 4-VALUED LOGIC WITH 3 VARIABLES.

FIGURE III.9.

type for problems in four-valued logic and three variables appears in Figure III.9. This space consists of $(4-1)^3$ subcubes and contains 4^3 points representing all possible assignments of the truth-values 0,1,2,3 to the three variables. The truth-value assignments are indicated for a few of the points in the diagram. As with problem spaces for two-valued logic, it has been possible to draw this space with all the points arranged in levels such that all the points at any level are equi-distant from the upper-most point in terms of the minimum number of single transitions required to reach the upper-most point. A single transition in this space is one in which the truth-value of only one logical variable is changed and in which the truth-value of this variable changes by unity.

Again a homomorphic space is required in which all the solutions to any problem are mapped on to a single point. The truth-value space is the only obvious space possessing this property and, in this case, it consists in a simple ordering of the truth-values (Figure III.10).

As this space contains m points (for m-valued logic) it is meaningful to talk of the "distance" of any point from the upper-most point on to which all the solutions are mapped. The truth-value itself provides a perfect measure in this space, having a maximum value at the solution point and decreasing in value at points further away from the solution point



TRUTH-VALUE SPACE
FOR M-VALUED LOGIC.

FIGURE III.10

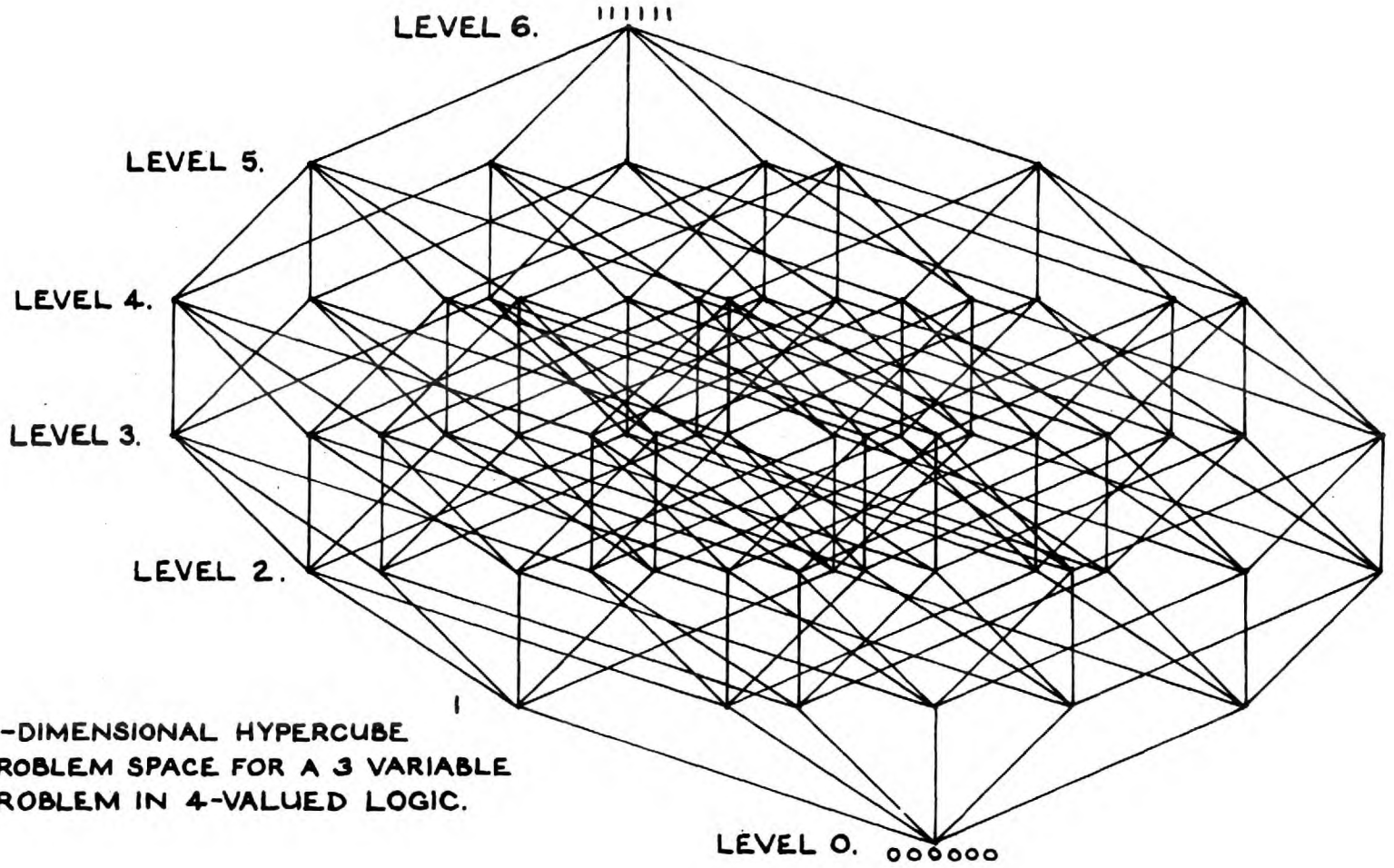
Now, suppose that a logical machine is searching for solutions to a many-valued logical problem and that the machine's transition from any existing state to the next state is constrained to incur a change

in the truth-value of only one variable, by unity. The machine will then be searching through a space having a structure similar to that in Figure III.9. The machine considers a succession of truth-value assignments to the logical variables occurring in the problem and tests them to see whether they are solutions. This can be represented by imagining a "marker point" moving about in the problem space in correspondence with the machine's transitions. A given point in the problem space corresponds to a particular assignment of truth-values which, when substituted in the logical rules of the problem, yields a resultant truth-value for the whole problem. This resultant truth-value is represented by a point in the truth-value space (Figure III.10). This is the way in which points in the problem space are mapped on to those in the truth-value space.

As the marker point in the problem space moves about in accordance with the machine's moves a corresponding marker point can now be imagined moving about in the truth-value space. The position of this marker point in the truth-value space will be given by a mapping of the position of the marker point in the problem space.

Now, if the problem space is orientated so as to bring one solution to the problem to the upper-most point, and if the successive transitions of the machine are such that the marker point in the problem space moves steadily in an upward direction towards the solution point, will the marker point in the truth-value space also move steadily upward towards the solution point? If this is the case, then the truth-value of the problem will be an effective measure function. This depends on whether the many-valued logical connectives used in expressing the problem are order preserving functions in the truth-value space.

In Section III.5 it was shown that any problem in an m-valued logic can be reduced to an equivalent problem in two-valued logic. If the m-valued problem involves n variables it will reduce to a two-valued problem involving n.N variables, where N is the least integer such that $2^n \geqslant m$. If $m=2^n$ there is a one to one correspondence between the $2^{n.N}$ assignments of truth-values to the two-valued variables



6-DIMENSIONAL HYPERCUBE
PROBLEM SPACE FOR A 3 VARIABLE
PROBLEM IN 4-VALUED LOGIC.

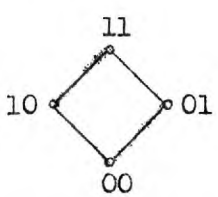
FIGURE III.11.

and the m^n assignments of truth-values to the m-valued variables. If m is not an integral power of two, then some of the assignments of truth-values to the two-valued variables are redundant and are not used.

To avoid unnecessary obsurity consider only logics where the value, m , is an integral power of two. Everything that follows applies equally well to logics of any value, however.

Another possible problem space for a many-valued problem is the hypercube space for the equivalent two-valued problem. The three-dimensional space with a metric in Figure III.9 was for a three-variable problem in four-valued logic. The equivalent problem in two-valued logic requires six variables (two to represent each four-valued variable). An alternative problem space for this problem is the six-dimensional hypercube shown in Figure III.11. This problem space is only applicable when the problem is represented within the logical computer in terms of the equivalent two-valued problem and the computer is restricted to making single transitions in which only one of the two-valued variables changes state.

The truth-value space for the problem in four-valued logic is shown in Figure III.12 in which the truth-values are partially ordered.



TRUTH-VALUE SPACE
FOR 4-VALUED LOGIC.

FIGURE III.12.

If the existing truth-value of one of the four-valued variables is 01 (being the binary representation of 1), then in the next transition the truth-value of this variable could change to 11 or 00 (representing

3 or 0). A single transition could not change the truth-value from 01 to 10. This leads to the partial ordering of the truth-values.

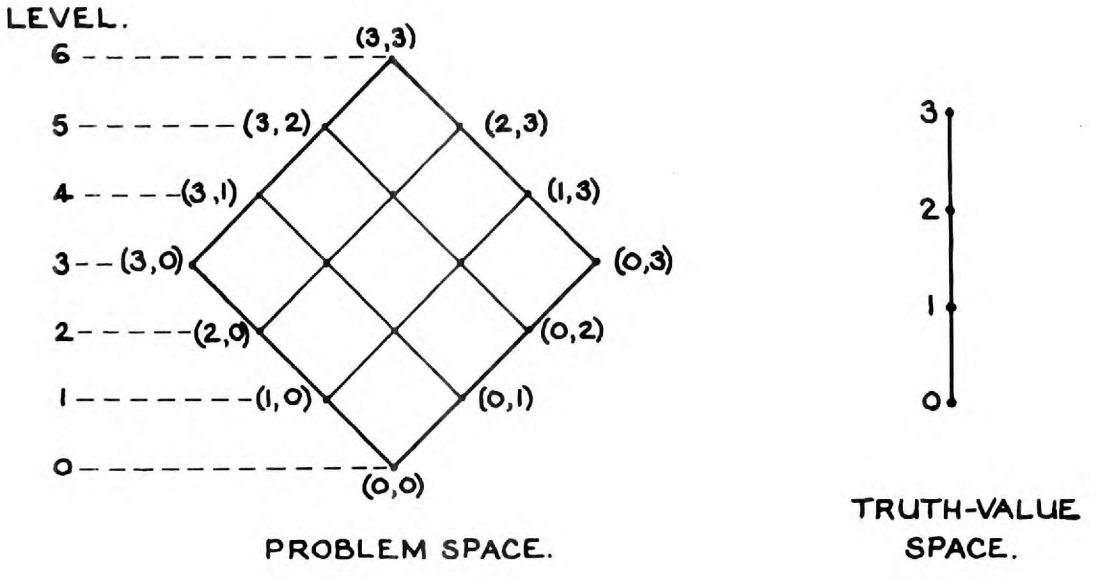
The truth-value space for a problem in m-valued logic, where $m=2^N$, is an N-dimensional hypercube.

Once again all the solutions to a problem are mapped on to the upper-most point in the truth-value space. An interesting property of this truth-value space is that, for all the points at any given level in the space, the number of ones in the binary expression of the truth-value is the same. The number of ones in the binary representation of a truth-value will be called the PARITY of the truth-value. In Figure III.12 the parity is zero at the bottom level, unity at the middle level and two at the top level.

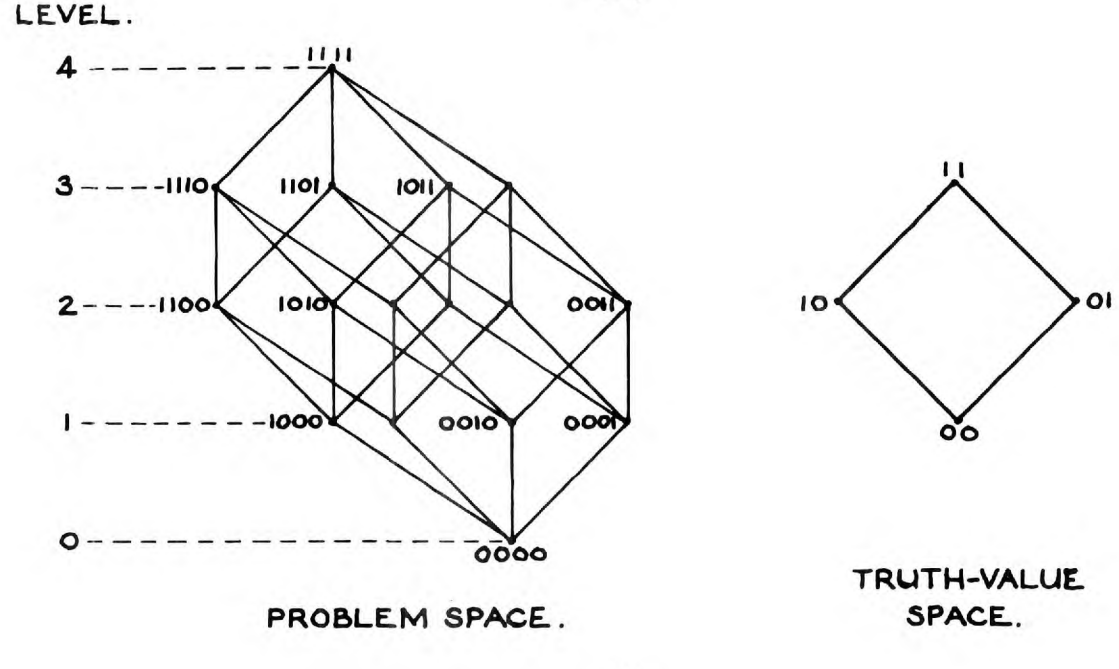
The parity has the properties required of a measure function and can be used as such providing that the many-valued logical connectives used in formulating problems are order preserving functions in the hypercube truth-value space.

For means of comparison the two alternative problem space structures together with the corresponding truth-value spaces for a two-variable problem in four-valued logic are shown in Figure III.13.

In summary then, two possible measure functions have been suggested but their effectiveness requires that the many-valued logical connectives used in formulating problems be order preserving in the appropriate truth-value space. The next Section describes an investigation to see whether this requirement is satisfied.



(a)



(b)

TWO PROBLEM SPACE STRUCTURES FOR A 2 VARIABLE PROBLEM IN 4-VALUED LOGIC.

FIGURE III.13.

IV. PROGRAMMING THE FERRANTI "PEGASUS" COMPUTER TO INVESTIGATE THE PROPOSED MEASURE FUNCTIONS

IV.1 The Method of Investigation

The class of many-valued logical problems of interest consists of those problems in which a given set of logical constraints involving many-valued propositional variables and connectives must all be satisfied simultaneously. A solution to such a problem is any assignment of truth-values to the propositional variables under which all the constraints are satisfied and, consequently, the truth-value of the conjunction of the set of problem constraints has the maximum value possible in the system of logic being used.

In the case of a problem expressed in an m-valued logic the truth-value of the conjunction of the problem constraints at all points in the problem space corresponding to solutions will be m. At all other points in the problem space the conjunction of the problem constraints will have a truth-value in the range 0 to m-1.

The proposed measure functions are :

(i) the truth-value of the conjunction of the problem constraints at any given point in the problem space, and

(ii) the number of ones in the binary expression of the truth-value of the conjunction of the problem constraints at any given point in the problem space, this being referred to as the PARITY of the truth-value.

It was decided to investigate the proposed measure functions by programming the PEGASUS computer to evaluate the functions at all the points in the problem space for a selection of problems in 16-valued logic. For the purpose of programming the computer the problems in 16-valued logic were expressed in terms of two-valued logic in the

manner described in Section III.5. The investigation for each problem considered was divided into two parts and the two separate computer programmes constructed for this purpose will be referred to as Programme A and Programme B respectively.

The first part of the investigation was to make an exhaustive scan through all the points in the problem space of the given problem, evaluating at each point the truth-value of the set of problem constraints and the parity of this truth-value. At the same time all the solutions to the problem were enumerated. This much was done by the orders of Programme A .

After considering the available storage space in the Pegasus computer and the fact that the amount of computer time required, and hence the cost, increases rapidly with the number of propositional variables in the problem, it was decided to limit the investigation to problems involving three 16-valued variables. These were represented for the purpose of programming by 12 two-valued variables, i.e. four two-valued variables per 16-valued variable. The relevant problem space therefore had 12 dimensions and 13 levels.

After making an exhaustive scan of a problem and enumerating the solutions by using Programme A one solution was selected arbitrarily and the hypercube problem space was imagined to be orientated so as to bring this solution to the upper-most point. All the points at any one of the 12 levels below this solution would then have been equi-distant from the solution at the top of the hypercube. It was the function of Programme B to compute the truth-value and parity of the conjunction of the set of problem constraints at each level in the hypercube averaged over all the points at that level. Hence, 13 pairs of average values were obtained corresponding to the 13 levels in the hypercube. In addition to this the standard deviation of these values was computed for all the points at each level in the space. This procedure was repeated several times for each problem bringing a different solution to the upper-most point of the hypercube on each occasion.

These results were expected to yield information regarding the behaviour of the proposed measure functions throughout the problem space and, in particular, in the proximity of solutions.

IV.2 The Problems Used

A selection of arbitrarily constructed problems were used in 16-valued logic involving Lewis connectives. In Section III.5 it was found that Lewis connectives can be expressed very simply in terms of two-valued logical variables and connectives (see p.45). The problems put on Pegasus were expressed in terms of two-valued logic, each 16-valued variable, V , being represented by four two-valued variables, V_4 , V_3 , V_2 , V_1 . The following connectives were used (the same notation is used below as was used in Section III.5) :

(i) NEGATION, (–) .

$$\text{If } F(A) \equiv -A ,$$
$$\text{then } F_1 \equiv \sim A_1$$
$$F_2 \equiv \sim A_2$$
$$F_3 \equiv \sim A_3$$
$$F_4 \equiv \sim A_4$$

(ii) Lewis AND, ($\cap$) .

$$\text{If } F(A,B) \equiv A \cap B,$$
$$\text{then } F_1 \equiv A_1 \cdot B_1$$
$$F_2 \equiv A_2 \cdot B_2$$
$$F_3 \equiv A_3 \cdot B_3$$
$$F_4 \equiv A_4 \cdot B_4$$

(iii) Lewis OR, ($\cup$) .

$$\text{If } F(A,B) \equiv A \cup B ,$$
$$\text{then } F_1 \equiv A_1 \vee B_1$$
$$F_2 \equiv A_2 \vee B_2$$
$$F_3 \equiv A_3 \vee B_3$$
$$F_4 \equiv A_4 \vee B_4$$

(iv) IMPLICATION, $(\rightarrow)$.

$A \rightarrow B$ is defined as $-(A \cap -B)$.

(v) IDENTITY, $(\equiv)$.

$A \equiv B$ is defined as $(A \cap B) \cup (-A \cap -B)$.

(vi) Lewis J-FUNCTIONS, $\lceil J_i(\ ) \rfloor$.

The 16 J-FUNCTIONS for 16-valued logic were defined by the following Table. These functions have the required property that $\lceil J_0(A) \cup J_1(A) \cup \text{------} \cup J_7(A) \rfloor$ has the same truth-value as $-A$, while $\lceil J_8(A) \cup J_9(A) \cup \text{------} \cup J_{15}(A) \rfloor$ has the same truth-value as A .

A	J_{15}	J_{14}	J_{13}	J_{12}	J_{11}	J_{10}	J_9	J_8	J_7	J_6	J_5	J_4	J_3	J_2	J_1	J_0
15	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	14	0	0	0	0	0	0	1	1	1	1	1	1	1	0
13	0	13	13	0	0	0	0	0	2	2	2	2	2	2	2	0
12	0	12	12	12	0	0	0	0	3	3	3	3	3	3	3	0
11	0	11	11	11	11	0	0	0	4	4	4	4	4	4	4	0
10	0	10	10	10	10	10	0	0	5	5	5	5	5	5	5	0
9	0	9	9	9	9	9	9	0	6	6	6	6	6	6	6	0
8	0	8	8	8	8	8	8	8	7	7	7	7	7	7	7	0
7	0	7	7	7	7	7	7	7	8	8	8	8	8	8	8	0
6	0	6	6	6	6	6	6	6	0	9	9	9	9	9	9	0
5	0	5	5	5	5	5	5	5	0	0	10	10	10	10	10	0
4	0	4	4	4	4	4	4	4	0	0	0	11	11	11	11	0
3	0	3	3	3	3	3	3	3	0	0	0	0	12	12	12	0
2	0	2	2	2	2	2	2	2	0	0	0	0	0	13	13	0
1	0	1	1	1	1	1	1	1	0	0	0	0	0	0	14	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15

LEWIS J-CONNECTIVES FOR 16-VALUED LOGIC

FIGURE IV.1.

Other problems were used incorporating 16-valued Post functions. These problems were also represented within the computer by two-valued logical variables. Although it is possible to synthesize Post connectives in terms of two-valued logical connectives it is much simpler to obtain the Post connectives on a computer by using ordinary arithmetic operations. The following connectives were used:

Let "a" and $f(a)$ be the truth-values of A and $F(A)$, a logical function of A .

(i) NEGATION, $(-)$.

This is the same as the NEGATION connective in the Lewis system, but as the other Post connectives were obtained using arithmetic operations the NEGATION was also obtained in this way -

$$\text{If } F(A) \equiv -A ,$$
$$\text{then, } f(a) = 15 - a .$$

(ii) Post AND, $(\cap)$.

$$\text{If } F(A,B) \equiv A \cap B ,$$
$$\text{then, } f(a,b) = \text{MIN.}(a,b).$$

(iii) Post OR, $(\cup)$.

$$\text{If } F(A,B) \equiv A \cup B ,$$
$$\text{then } f(a,b) = \text{MAX.}(a,b).$$

(iv) CYCLIC NEGATION $(')$.

$$\text{If } F(A) \equiv A'$$
$$\text{then, } f(a) = \begin{cases} a-1 & \text{if } a \neq 0 , \\ 15 & \text{if } a = 0 . \end{cases}$$

(v) Post J-FUNCTIONS, $\lfloor J_i( ) \rfloor$.

$$\text{If } F(A) \equiv J_i(A) ,$$
$$\text{then, } f(a) = \begin{cases} 15 & \text{if } a = i , \\ 0 & \text{if } a \neq i . \end{cases}$$

$$\text{for } i=0,1,2 \text{ ----, } 15.$$

(vi) IMPLICATION, $(\rightarrow)$.

This connective is not in the system defined by Post. It is, in fact, the IMPLICATION connective used in formulating the railway timetable problem in Section III.4, (p. 40).

$$\text{If } F(A,B) \equiv A \rightarrow B,$$

$$\text{then, } f(a,b) = \begin{cases} 15 & \text{if } a \leqslant b , \\ 0 & \text{if } a > b . \end{cases}$$

Problems were constructed arbitrarily using, first, the Lewis connectives and then the Post connectives. Programme A was put on to the computer for each problem and any problem having no solutions or far too many solutions was rejected. Problems having fewer than 100 solutions out of the 4,096 possible truth-value assignments were selected for further investigation.

IV.3 Pegasus Programme A .

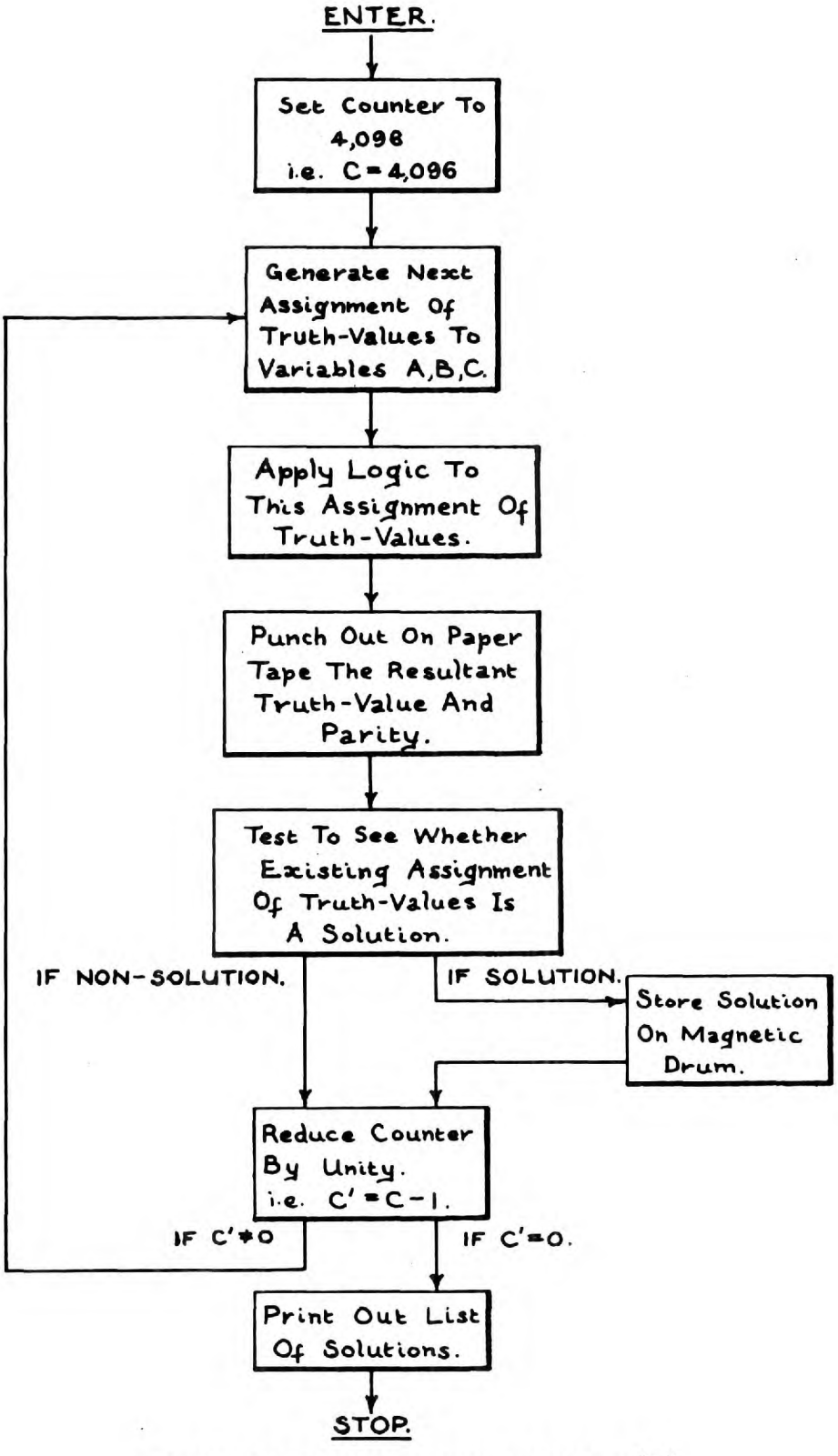
The flow diagram for Programme A is shown in Figure IV.2. The main part of the programme consists of a loop of orders which have to be obeyed 4,096 times, i.e. once for each assignment of truth-values in the exhaustive scan of the 12 two-valued variables in terms of which the three variables in 16-valued logic are being expressed. When the programme is "entered" a counter is set to the value 4,096. This consists in placing the number 4,096 in a known storage location within the computer. Each time the main loop of orders in the programme is obeyed the value of this counter is reduced by unity. The counter therefore changes in value as the programme progresses and its value will be denoted by C .

All numbers are represented inside the computer in binary form. The counter which initially has the value 4,096 therefore consists of a 12 digit binary number in which all the digits are initially ones. As the value of the counter is reduced in unit steps the 12 binary digits composing the counter are stepped through 2^{12} combinations from 111----1

to 000----0. Hence, in addition to functioning as a counter, these 12 binary digits may also be used to represent the assignment of truth-values to the 12 two-valued logical variables.

Let (a_4,a_3,a_2,a_1), (b_4,b_3,b_2,b_1) and (c_4,c_3,c_2,c_1) be the sets of binary digits used to represent the truth-values assigned to the 16-valued variables A,B and C respectively. Each storage location in Pegasus has a capacity of 38 bits. In writing these programmes one storage location was used to hold each set of four binary digits such as (a_4,a_3,a_2,a_1); the remaining 34 digit positions of each storage location were not used. Let S_A, S_B and S_C be the storage locations in the computer used to hold (a_4,a_3,a_2,a_1), (b_4,b_3,b_2,b_1) and (c_4,c_3,c_2,c_1) respectively.

The first few orders in the main loop of orders cause the most significant four digits of the counter, C, to be copied and stored in S_A, the middle four digits of C to be copied and stored in S_B and the least significant four digits of C to be copied and stored in S_C. The next part of the programme substitutes the existing truth-value assignment in the logical constraints constituting the problem and determines the truth-value of each individual constraint. Since all the constraints must be satisfied conjunctively the individual constraint truth-values are combined using a 16-valued AND connective to give the resultant truth-value of the whole problem. This truth-value is punched out in binary form on a paper tape. The number of ones in the truth-value is counted to find the parity and this is punched on the paper tape, also in binary form, immediately after the truth-value. The computer then tests the truth-value of the problem to see if the existing truth-value assignment is a solution. If it is a solution it is stored on the magnetic drum so that it may be printed out later. At this stage the value of the counter, C, (initially set to 4,096), is reduced by unity. The resulting value is then tested and, if it is non-zero, the computer repeats the main loop of orders, applying the logic to the next assignment of truth-values, and so on.



FLOW DIAGRAM OF PROGRAMME "A".

FIGURE IV.2.

When the value of the counter has been reduced to zero the computer leaves the main loop of orders, prints out a complete list of solutions to the problem and stops. The computer output tape then contains the truth-value and the parity for the problem corresponding to each of the 4,096 assignments of truth-values to the variables.

IV.4 Pegasus Programme B.

It was the function of Programme B to compute the average and the standard deviation (S.D.) of the truth-values and parities of the problem over all the points at each of the 13 levels in the problem space when a chosen solution was situated at the opper-most point.

One solution to the problem being considered was chosen arbitrarily from the list of solutions printed out by Programme A. This solution was punched out on paper tape, as a series of 12 binary characters. This piece of paper tape was joined to the beginning of the output tape from Programme A containing the truth-value and parity data. This tape was fed to the computer as an input tape for Programme B. Of the 12 binary digits of the solution the most significant four formed the binary representation of the truth-value of the 16-valued proposition A, the middle four formed the binary representation of the truth-value of B and the least significant four formed the binary representation of C . This 12 binary digit solution will be denoted by S .

The flow diagram for Programme B appears in Figure IV.3. On entering the programme a counter, C, is set to the value 4,096 as in Programme A. Once again this counter serves as a check on the number of times the main loop of orders has been obeyed and also as an indication of the existing assignment of truth-values to the variables at any instant during the course of the programme. As the counter goes through the same sequence of values as did the counter in Programme A, the order of the truth-value/parity pairs on the input tape to Programme B corresponds to the order in which the truth-value assignments are generated by the counter.

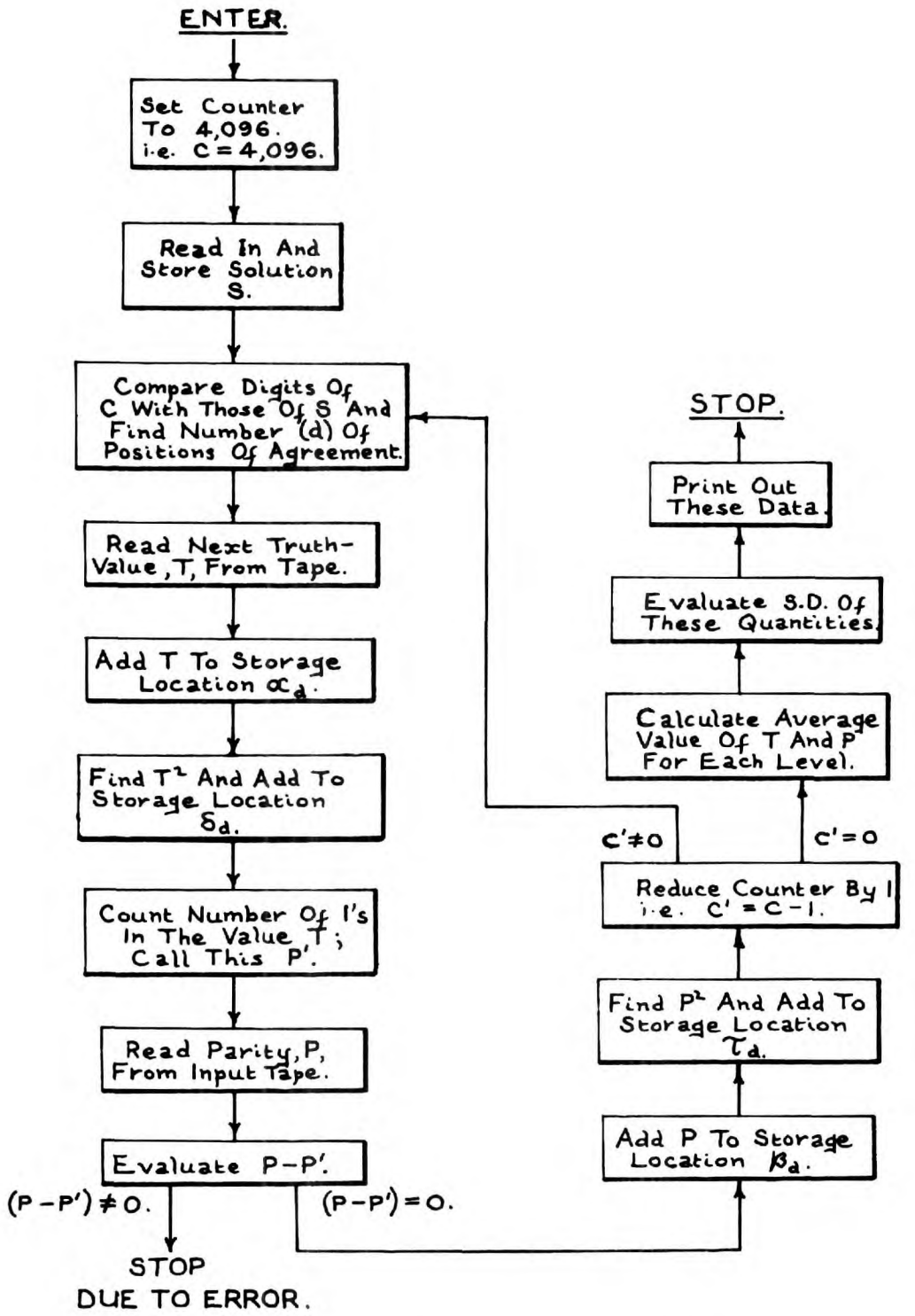
When the machine starts reading the input tape it comes first to the chosen solution, S . This it stores in a storage location where it will remain, undisturbed, throughout Programme B.

There are 13 levels in the hypercube at which the average values of truth-value and parity have to be calculated. For this purpose 13 storage locations are used to accumulate the sums of the truth-values at the 13 levels. These storage locations will be denoted by $\alpha_0, \alpha_1, \ldots, \alpha_{12}$, so that the sum of the truth-values of the problem at all points at level i in the problem space is accumulated in location α_i . Similarly storage locations, which will be denoted by $\beta_0, \beta_1, \ldots, \beta_{12}$, are used to accumulate the sums of the parities, the sum of the parities at all points at level i being accumulated in location β_i .

In order to compute the standard deviation of these quantities the sums of squares of truth-values are accumulated in storage locations denoted by $\delta_0, \delta_1, \ldots, \delta_{12}$ and the sums of squares of parities are accumulated in locations denoted by $\tau_0, \tau_1, \ldots, \tau_{12}$.

The solution, S, having been read from the tape, the next character to pass under the tape reader is the first truth-value on the tape, having a value T , such that $0 \leqslant T \leqslant 15$. This truth-value corresponds to the first assignment of truth-values to the variables which exists within the computer as the initial combination of the 12 binary digits of the counter, C. The 12 digits of the counter are compared with those of the chosen solution, S, and the number of places is computed in which the digits of C and S are identical (i.e. both unity or both zero.) Let this number be d ($0 \leqslant d \leqslant 12$). The value of d indicates the level in the hypercube to which the existing assignment of truth-values belongs.

E.g. If the digits of the chosen solution, S, were 000100000110 while the initial combination of digits of C was 111111111111 , then C and S would be identical in three digit positions and d would have the value three. The assignment of truth-values under consideration



FLOW DIAGRAM OF PROGRAMME "B".

FIGURE IV.3.

would then belong to level three in the hypercube, the solution, S being at level 12.

The first truth-value, T, read from the tape is arithmetically added to the contents of storage location α_d in which is being accumulated the sum of the truth-values of the problem at all points at level d in the hypercube. (At the start of Programme B the contents of all the storage locations α, β, δ, τ are set to zero.) The truth-value is then squared and the value T^2 is added to the contents of storage location δ_d. The next few orders in the programme instruct the machine to count the number of ones in the truth-value, T. This gives the value of the parity and it will be denoted by P'. The tape reader reads the next character on the input tape which is also the parity corresponding to the truth-value, T. The parity value read from the tape will be denoted by P. The values of P and P' are compared. If they are <u>not</u> identical the computer comes to a halt indicating that an error has been detected. This method of detecting errors proved to be a useful facility and on several occasions errors were found, some being due to faulty operation of the tape readers and tape punches used as input and output organs of the computer.

Assuming that the values of P and P' prove to be identical, the programme continues by adding the value P to the contents of storage location β_d. The value of the parity is then squared and P^2 is added to the contents of storage location τ_d. The value of the counter C, is then reduced by unity and if the resulting value is non-zero the main loop of orders of Programme B is repeated.

When the value of the counter becomes zero the machine leaves the main loop of orders. Knowing the number of points at each of the 13 levels in the problem space, the average values of truth-value and parity at each level are obtained by dividing the contents of the corresponding storage locations α and β by the appropriate number. Both the sums and the sums of squares of the truth-values and parities are available for each level in the hypercube. By performing some simple arithmetic

operations upon these the standard deviations of the truth-values and parities are calculated for each level. The average values and the standard deviations are then printed out and the machine stops.

IV.5 Results.

The results are best presented graphically by plotting the average truth-value at a given level and the average parity at a given level against the distance, in levels, from the chosen solution. Too much space would be required to present all the results in this way so the results for three representative problems are given.

The following is a statement of the logical constraints of the three problems for which results are given together with the number of solutions to each problem.

Problem 1. Lewis Logic

Rule 1 : $J_6(A) \cup C$
Rule 2 : $A \rightarrow B$
Rule 3 : $J_4(C) \equiv B$
Number of solutions : 1 .

Problem 2. Lewis Logic

Rule 1 : $\lceil \neg J_7(A) \rfloor \rightarrow J_6(C)$
Rule 2 : $B \rightarrow \neg C$
Rule 3 : $J_{14}(A) \rightarrow J_8(C)$
Rule 4 : $B \rightarrow A$
Rule 5 : $C \rightarrow A$
Number of solutions : 2 .

Problem 3. Post Logic

Rule 1 : $\lceil A \cup (B \cap C'')' \rfloor \cup \lceil J_4(B) \cap \neg C \rfloor$
Rule 2 : $(B \rightarrow C)' \rightarrow (C \rightarrow A)$
Rule 3 : $J_6(B) \cup \neg C$
Number of solutions : 74 .

The results for Problem 1 appear in Figures IV.4a and IV.4b. In Figure IV.4a the full line shows the variation of the average truth-value at different levels in the problem space. The broken line shows the variation over the levels of the standard deviation of the truth-values. Strictly, these "graphs" should consist of isolated points since the problem space is not a continuous space. However, the results are far easier to follow when the plotted points are connected by straight lines. Figure IV.4b shows the variation of the parity (full line) and the standard deviation of the parities (broken line) for Problem 1.

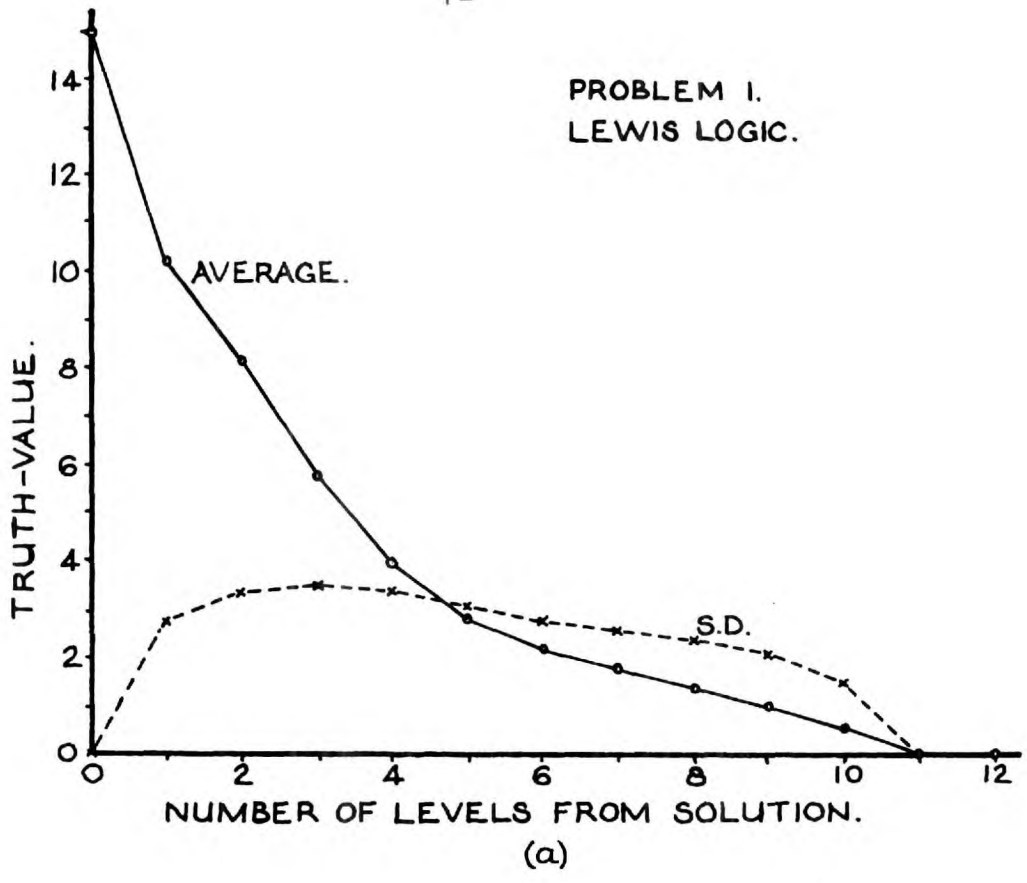
Problem 2 had two solutions and the two sets of graphs corresponding to these are given in Figures IV.5 and IV.6. Similarly, Figures IV.7 and IV.8 are for Problem 3, corresponding to orientations of the problem space to bring two out of the 74 solutions to this problem to the highest level.

The most remarkable thing about these results is the similarity in the behaviour of the average truth-value and the average parity of a problem.

The average values of the truth-value and the parity for Problem 1 are zero at the lowest level in the problem space and increase monotonically towards the solution, at the highest level in the space.

The results for Problem 2 show almost identical behaviour to those of Problem 1.

Consider now the results of Problem 3 when solution 1 is at the highest level of the problem space. Again the average truth-value and the average parity vary in a similar way, but this time the variation over the levels is not so smooth. The graphs are in the form of a step and the average values of truth-value and parity are more or less constant throughout most of the space. Even two levels below the solution the average values show no appreciable increase in value. The functions plotted would make very poor measure functions in this case. However, these results were amongst the worst of those obtained. Problem 3 had 74 solutions and the graphs plotted in Figure IV.8 correspond to an


PROBLEM I.
LEWIS LOGIC.
AVERAGE.
S.D.
TRUTH-VALUE.
0
2
4
6
8
10
12
14
NUMBER OF LEVELS FROM SOLUTION.
(a)

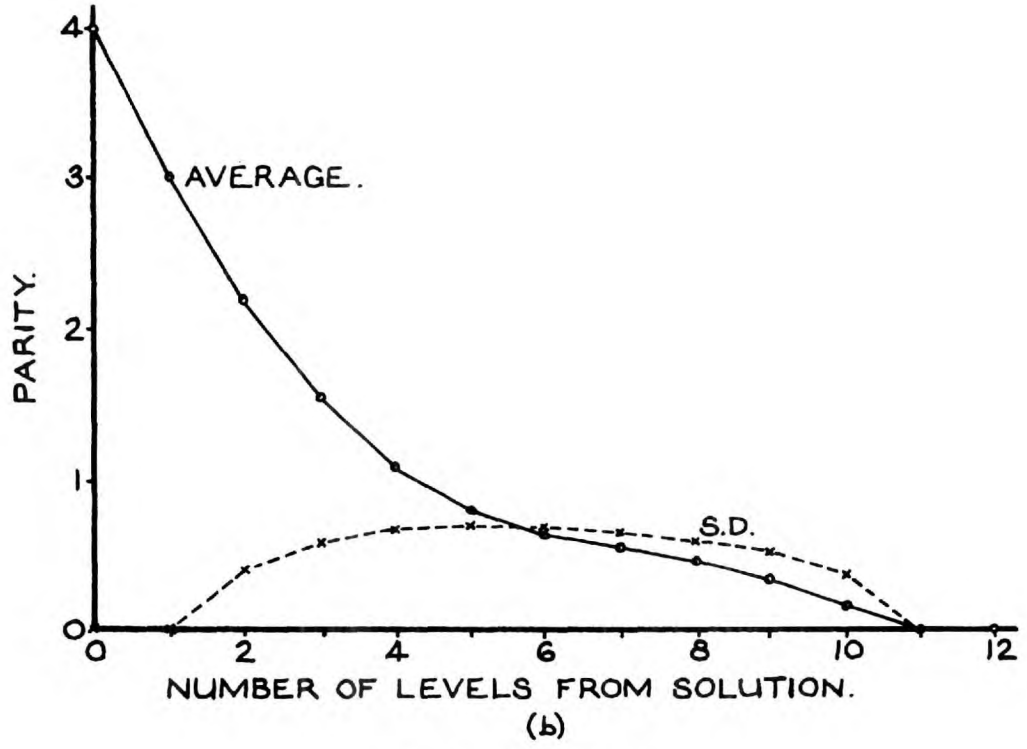
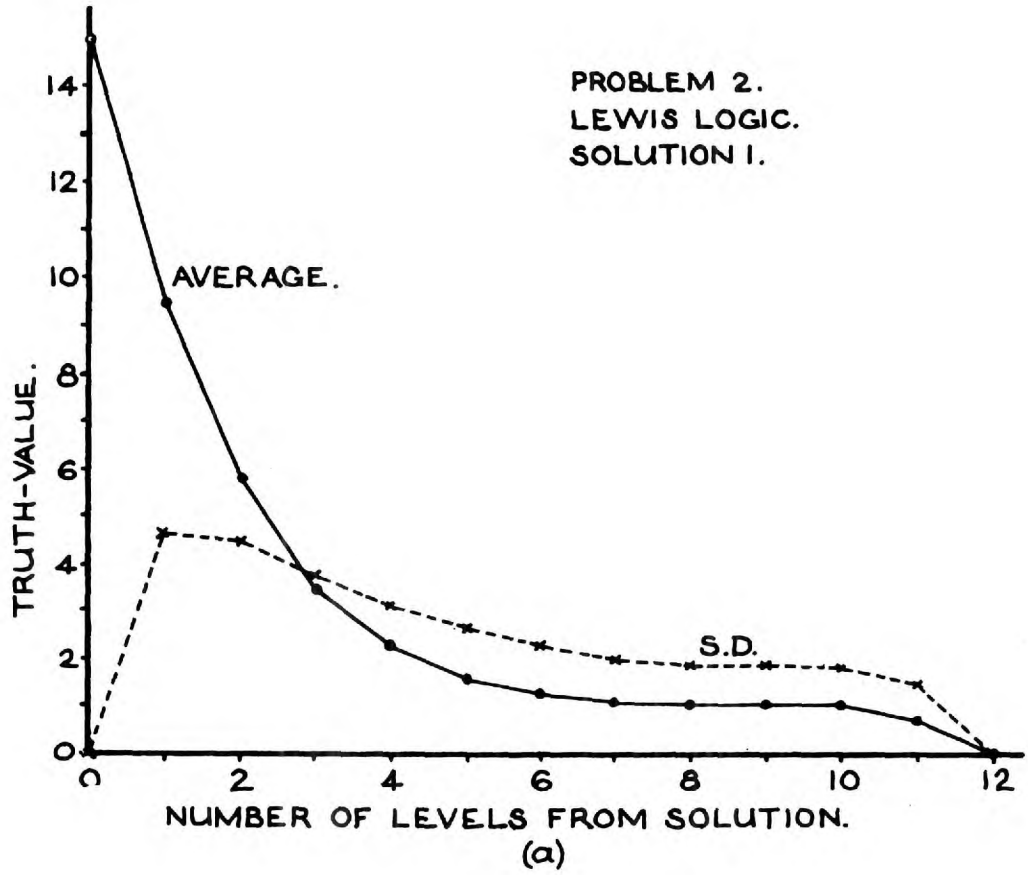

AVERAGE.
S.D.
PARITY.
0
1
2
3
4
0
2
4
6
8
10
12
NUMBER OF LEVELS FROM SOLUTION.
(b)

FIGURE IV.4.


PROBLEM 2.
LEWIS LOGIC.
SOLUTION 1.
AVERAGE.
S.D.
TRUTH-VALUE.
0
2
4
6
8
10
12
14
NUMBER OF LEVELS FROM SOLUTION.
(a)

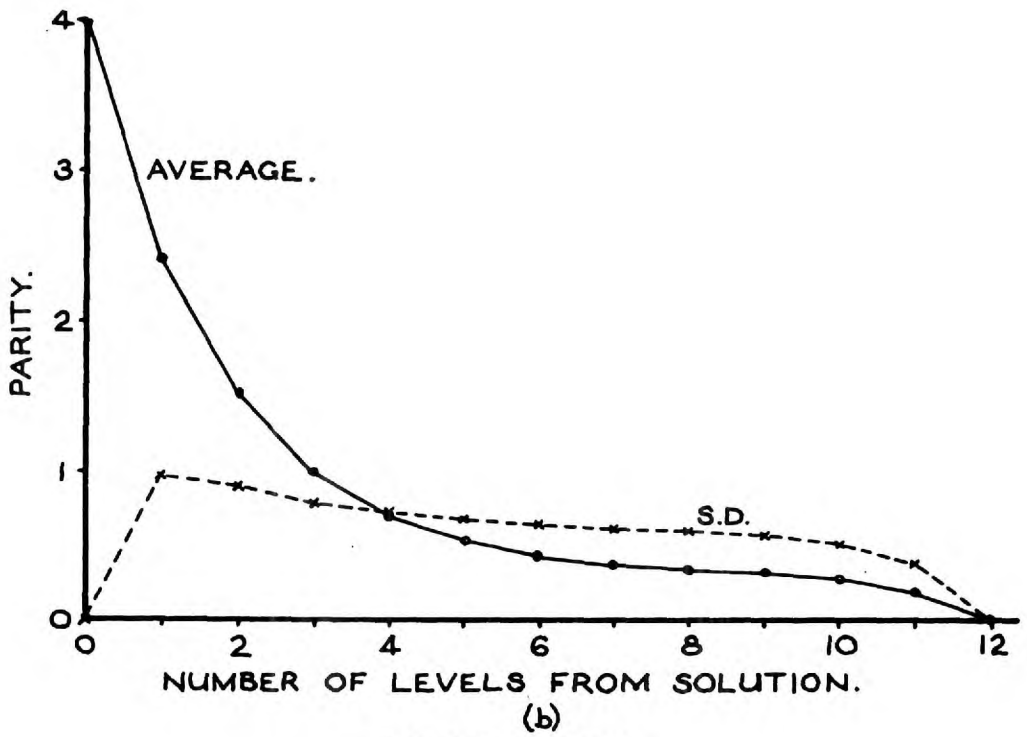
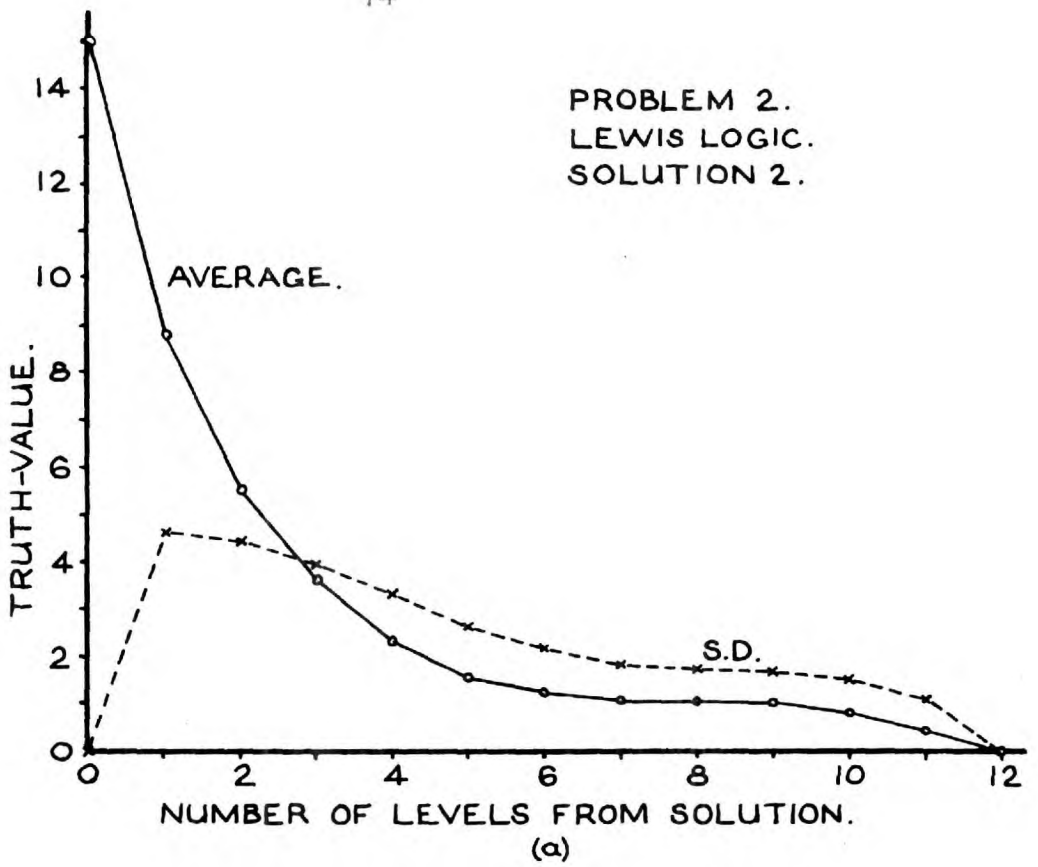
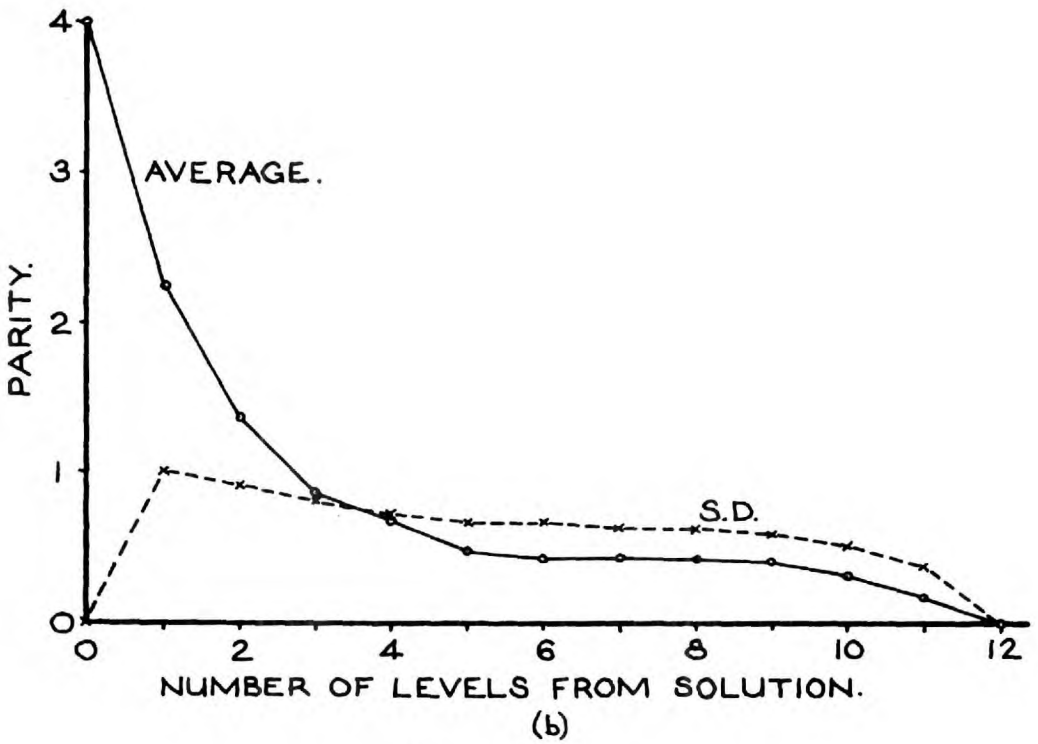

AVERAGE.
S.D.
PARITY.
0
1
2
3
4
0
2
4
6
8
10
12
NUMBER OF LEVELS FROM SOLUTION.
(b)

FIGURE IV.5.

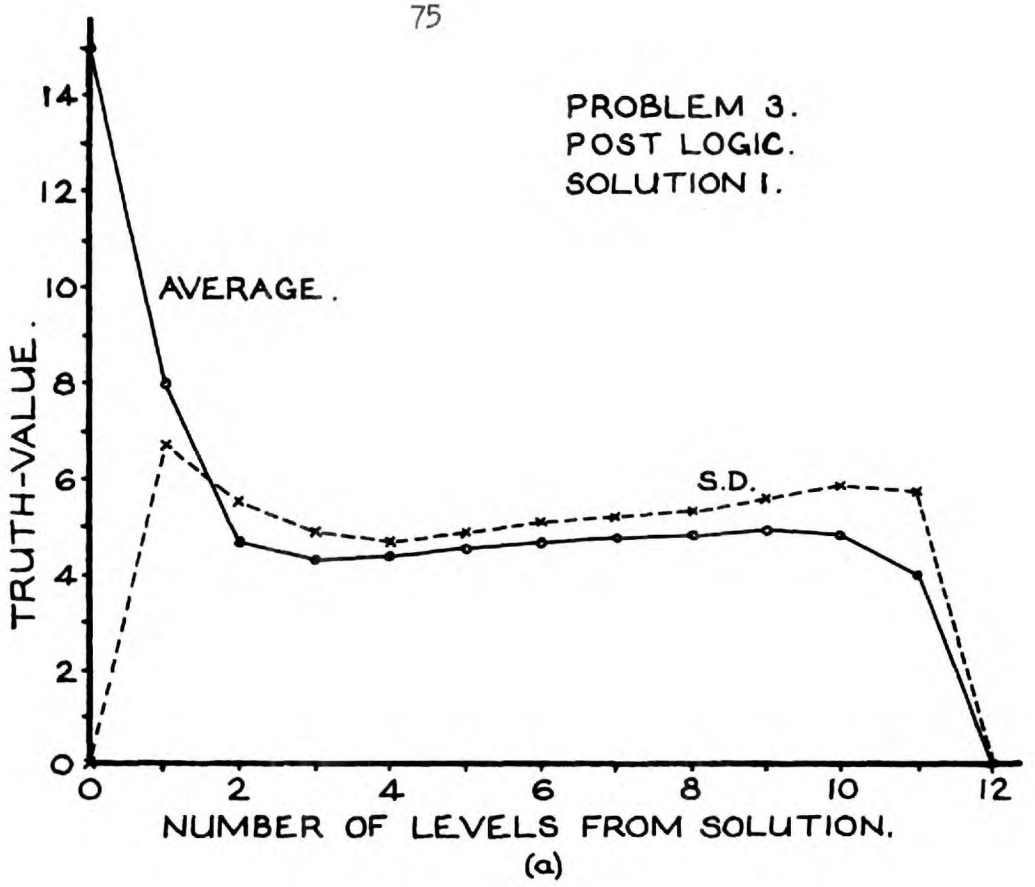


(a)



(b)

FIGURE IV.6.


PROBLEM 3.
POST LOGIC.
SOLUTION 1.
AVERAGE.
S.D.
TRUTH-VALUE.
0
2
4
6
8
10
12
14
NUMBER OF LEVELS FROM SOLUTION.
(a)

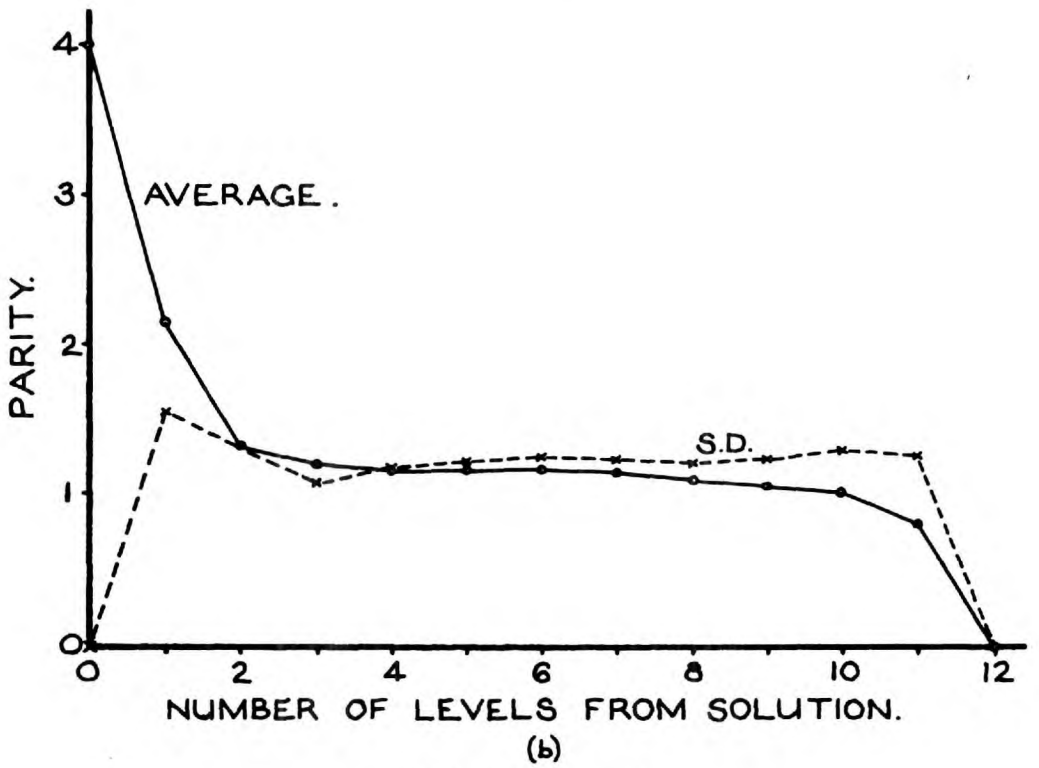
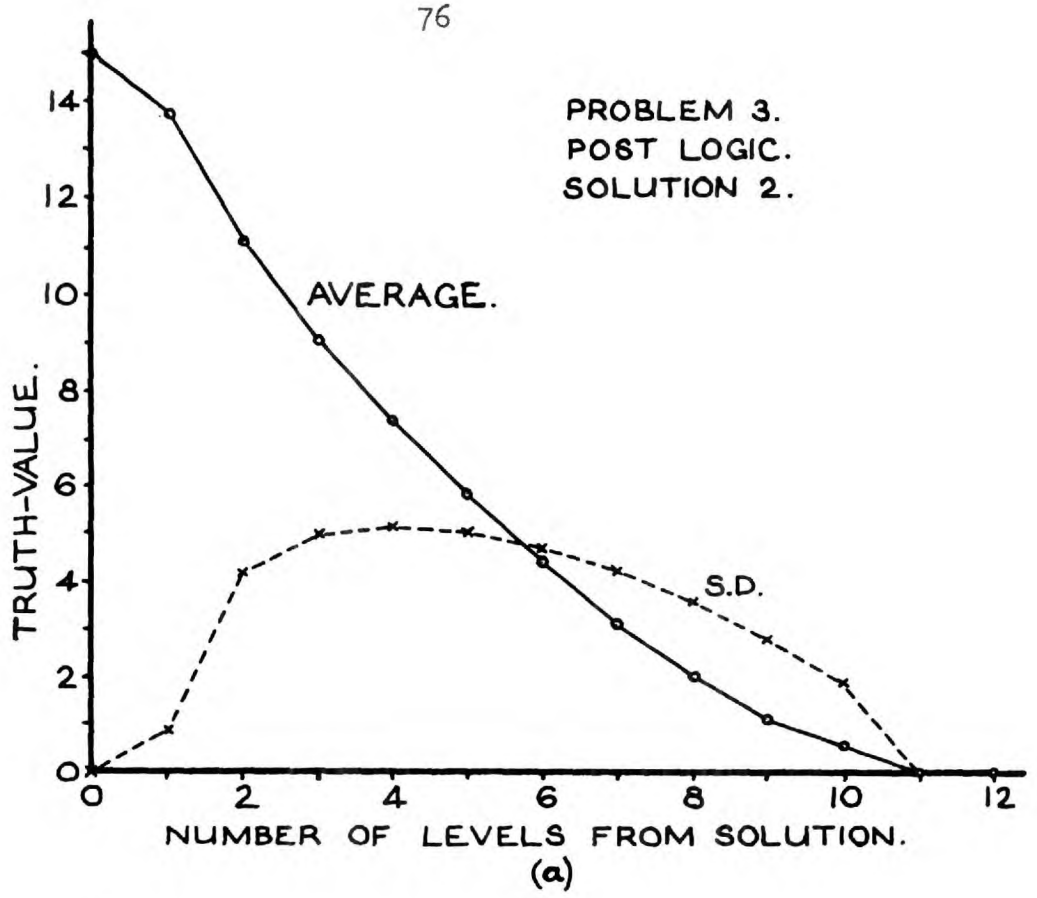

AVERAGE.
S.D.
PARITY.
0
1
2
3
4
NUMBER OF LEVELS FROM SOLUTION.
(b)

FIGURE IV. 7.


PROBLEM 3.
POST LOGIC.
SOLUTION 2.
AVERAGE.
S.D.
TRUTH-VALUE.
0
2
4
6
8
10
12
14
NUMBER OF LEVELS FROM SOLUTION.
(a)

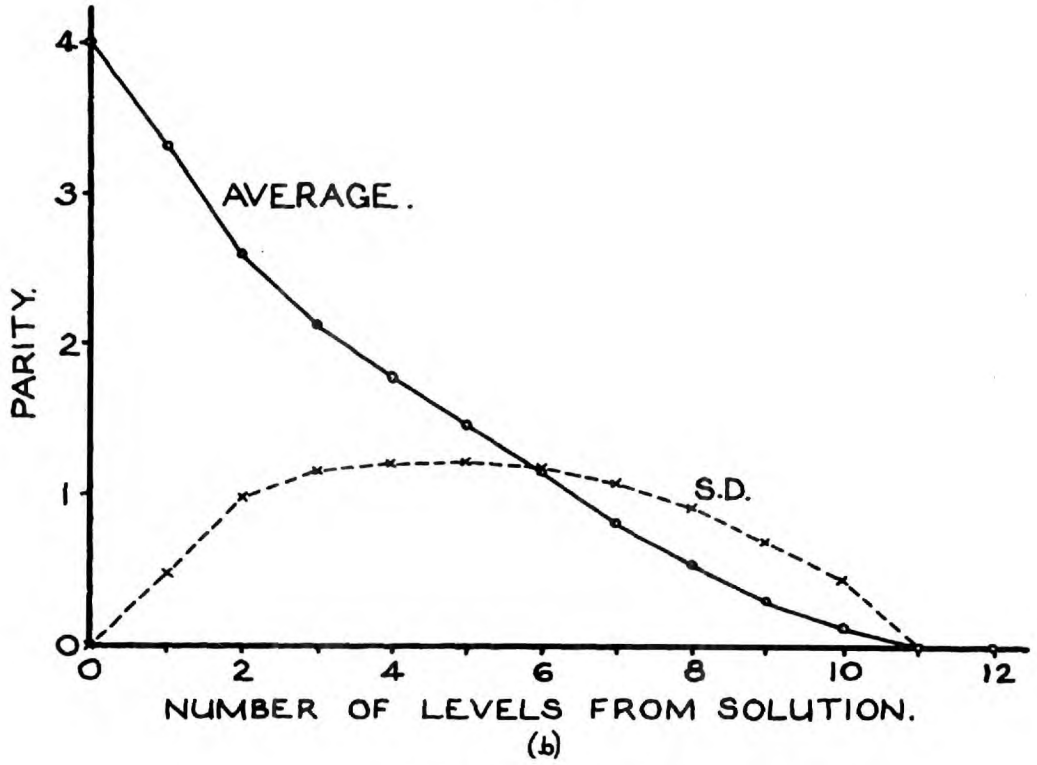

AVERAGE.
S.D.
PARITY.
0
1
2
3
4
0
2
4
6
8
10
12
NUMBER OF LEVELS FROM SOLUTION.
(b)

FIGURE IV.8.

orientation of the problem space to bring a different solution to the top. These results are seen to be similar to those obtained for the first two problems; the average truth-value and the average parity being zero at the lowest level in the space and increasing monotonically towards a maximum at the highest level in the space.

Examination of all the results obtained showed that, in most cases, the proposed measure functions, averaged over all the points at each level in the problem space, had a minimum value (not necessarily zero) at the lowest level in the space and increased monotonically to a maximum value at the highest level in the space (at which the chosen solution was situated).

In all cases the standard deviation of the proposed measure functions is extremely high. This is put down to the following reasons:

(1) In a 16-dimensional hypercube the numbers of points at the various levels of the hypercube are fairly small. Small samples tend to exhibit high standard deviations.

(2) The truth-value and parity functions can only assume integral values. Therefore, if there is any variation of the value of these functions over the points of a level, the minimum difference between the values at two points is unity. A change of unity in truth-value is quite appreciable in a 16-valued logic while a change of unity in the parity (restricted to the range 0 to 4 for a 16-valued logic) will have a very pronounced effect on the standard deviation.

Both these effects should be less noticeable if a higher valued logic were used.

In continuous variable theory a function of two variables can be represented by a contour map. Various techniques exist for exploring the surface of such contour maps and finding the peaks and troughs.

The results of the experiments on problems in 16-valued logic suggest that the proposed measure functions might be represented by

multi-dimensional contour maps, the solutions to a problem occurring at the peaks of the map. It is thought that, if high valued logics are used, (much higher than the 16-valued logics used in the experiments) the "surface" of the multi-dimensional contour maps would be quite smooth. The truth-value and the parity would then be useful measure functions when searching for solutions to these problems.

IV.6 The Use of Measure Functions in Logical Problem Solving Procedures.

Solution procedures for problems in many-valued logic could easily be devised (and machines could easily be designed to perform them) using measure functions, similar to those described, as a guide when searching through a problem space for solutions.

If the truth-value were used as a measure function the most appropriate problem space would be that described in Section III.6 having one dimension for each variable in the problem. The space for an n variable problem would be an n-dimensional hypercube. For problems in m-valued logic each edge of the hypercube would have m equally spaced points so that the space would consist of $(m-1)^n$ sub-hypercubes. A single transition in this space would involve a change of unity in the truth-value of only one logical variable.

The machine could start the solution procedure by considering a randomly chosen point in the space. It could then find the average truth-value in the neighbourhood of this point. The neighbourhood could be defined as consisting of all those points within, say, four single transitions of the given point. The machine could scan through all these points and compute the average truth-value very easily. If this average value were above a certain specified level the machine could be made to intensify the search in this part of the space. If movement of the search in a particular direction in the space tended to increase

the mean truth-value level the machine could be made to move its centre of concentration in that direction. In this way there would be a good chance of finding any solution existing in that part of the problem space.

If, after searching for some time, the machine had been unsuccessful, it could be made to take a random jump to another point in the space and to start again from there. Alternatively, it might prove better to use a set of starting points uniformly distributed throughout the problem space.

If a problem in many-valued logic were expressed in terms of two-valued logical variables and connectives the parity could be used as a measure function (the parity being the number of ones in the binary expression of the truth-value of the problem). The appropriate problem space for a problem in m-valued logic involving n variables would then be an n.N-dimensional hypercube, where N is the least integer such that $2^N \geqslant m$. A single transition in this space would correspond to a change in truth-value of one of the two-valued variables in terms of which the problem was being expressed. The same sort of solution procedure as that described above could be adopted using the parity as a measure function.

Although only two measure functions have been investigated others might exist. The most appropriate problem space and measure function might depend upon the system of logic being used and upon the particular connectives defined in that system. This, in turn, would depend upon the physical problem being solved.

The next part of the investigation would have consisted in programming a computer to carry out the type of problem solving procedure described using many-variable problems. However, to be of any use such an investigation would have to be quite long, being repeated for many different problems.

Since the development of many-valued logics and their application is very much in its infancy it was felt that such a continuation of the

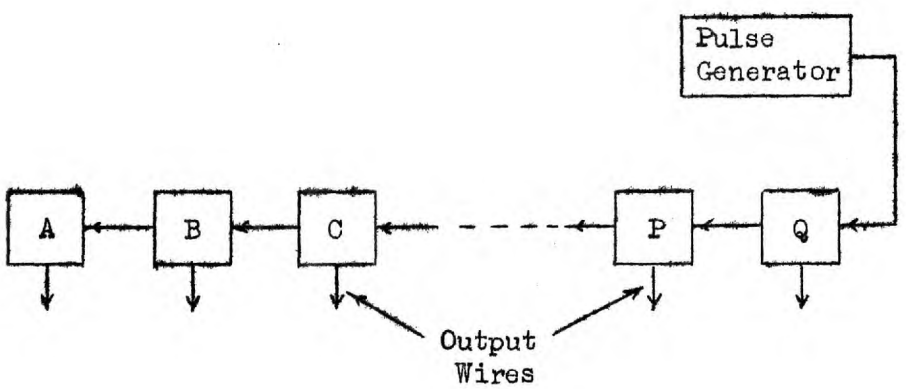
investigation would be rather premature.

Nevertheless, should many-valued logics be developed to the stage where it becomes practicable to formulate problems in this way, an investigation along the lines outlined above might prove very fruitful.

V. AN IMPROVED TECHNIQUE FOR SCANNING THE TRUTH-VALUE ASSIGNMENTS TO PROBLEMS IN LOGIC

V.1. Use of a Binary Counter for Generating an Exhaustive Scan of the Truth-Value Assignments to Problems in Two-Valued Logic

In Section II two machines were described that solved problems in logic by constructing, one at a time, all the rows of the truth-table. It is possible to use an electrical binary counter as the scanning mechanism for a machine of this type, as illustrated in Figure V.1.



USING A BINARY COUNTER TO GENERATE AN EXHAUSTIVE SCAN OF TRUTH-VALUE ASSIGNMENTS

FIGURE V.1.

The logical variables in the problem, A,B,C,----P,Q, are each represented by one stage of a binary counter. The arrows connecting the stages represent the carry lines of the counter. Each stage of the counter is a bistable circuit having an output wire, held at earth potential to indicate a truth-value of zero and held at a positive potential to indicate a truth-value of unity. Logical connective boxes

are connected to the output wires in accordance with the logical constraints. The output from the main connective box of each constraint is connected to one input of a final AND gate. When the output wire of this AND gate becomes energized a solution is indicated.

Before the truth-value scanning starts all the stages of the counter are set to the zero state. The pulse generator is then switched on and a train of pulses is fed to the input of the least significant stage of the binary counter. The counter is thus made to scan exhaustively through all its states from 000---00 to 111---11. In this way the truth-value assignments, represented by the states of the output wires of the counter, are scanned exhaustively according to the order of the binary code.

V.2. Generation of an Abbreviated Scan of the Truth-Value Assignments to Two-Valued Problems.

Consider the problem consisting of the two rules:

1. $A \not\equiv B$
2. $(A.C) \equiv D$

If the variables are arranged in alphabetical order the exhaustive list of truth-value assignments as generated by a binary counter is given below.

Row Number.	A	B	C	D	
1	0	0	0	0	Constraint 1 NOT satisfied
2	0	0	0	1	
3	0	0	1	0	
4	0	0	1	1	
5	0	1	0	0	Constraint 1 satisfied
6	0	1	0	1	
7	0	1	1	0	
8	0	1	1	1	
9	1	0	0	0	
10	1	0	0	1	
11	1	0	1	0	
12	1	0	1	1	
13	1	1	0	0	Constraint 1 NOT satisfied
14	1	1	0	1	
15	1	1	1	0	
16	1	1	1	1	

FIGURE V.2.

When the truth-values of row 1 are assigned to the variables it is found that constraint 1 is not satisfied. Since constraint 1 involves only the variables A and B the next assignment of truth-values that need be considered is the next one in the binary code bringing about a change in truth-value of at least one of the variables A, B. It is therefore possible to omit rows 2,3 and 4 and to jump straight to row 5 (as indicated by the arrow). Constraint 1 is now satisfied, and remains satisfied for the next seven assignments. It is therefore necessary to consider each of the eight assignments from row 5 to row 12. On considering row 13 it is found that constraint 1 is again not satisfied. However, all possible assignments of truth-values to the variables A and B have been tried. Hence, without trying rows 14, 15 or 16, it is known that no more solutions exist. All the solutions to this simple problem can therefore be found by considering only ten truth-value assignments; whereas the exhaustive scan contains 16 assignments.

It was seen that the exhaustive scan can be generated by adding in ones at the least significant end of a binary counter which, in this case, corresponds to the variable D. The abbreviated scan described above can be generated by adding in ones at the least significant end of the counter when constraint 1 is satisfied and adding in ones to the stage of the counter corresponding to the variable B when constraint 1 is not satisfied. It should also be observed that, in the transition from row 8 to row 9, a one is added in to the stage corresponding to B by virtue of a carry from the stage of lower significance corresponding to C. This occurs when all possible assignments of truth-values to C and D have been tried. The same thing occurs in the transition from row 12 to row 13.

This scanning system can be extended to problems involving any number of variables and constraints.

This is demonstrated by means of another example:

Constraint 1. $C \supset G$

Constraint 2. $\sim A \not\equiv (F \supset \sim C)$

Constraint 3. $B.F \equiv A. \sim G$

Constraint 4. $D \equiv (B \not\equiv F)$

Constraint 5. $(A \vee D \vee F) \supset \sim C.E$

Suppose that, in a logical machine for solving this problem, the seven variables are represented by the stages of a seven digit binary counter. It is assumed that input digits can be fed to any stage of the counter.

The above ordering of the constraints is quite arbitrary. Let the order in which the variables are represented by the stages of the binary counter be decided in the following way. Let the most significant two stages of the counter correspond to the variables C and G of constraint 1. The additional variables (i.e. additional to C and G) appearing in constraint 2 are A and F. Let these correspond to the most significant two of the remaining stages of the counter. Constraint 3 has only one additional variable (additional to C,G,A and F). That is the variable B. Let B correspond to the most significant of the remaining stages of the counter. The ordering is continued in this way and the result is given below, the left-hand position being the position of highest significance.

Groups	1	2	3	4	5
Variables	C G ,	A F ,	B ,	D ,	E
Constraints	1	2	3	4	5

The variables have been arranged in groups in such a way that constraint 1 involves only variables in group 1, constraint 2 involves only variables in groups 1 and 2, constraint 3 involves only variables in groups 1, 2 and 3, and so on.

The scanning commences with all the stages of the counter set to zero so that the initial assignment of truth-values is

CG,	AF,	B,	D,	E
00	00	0	0	0

(The commas indicate the grouping of the variables). It is found that, with this assignment of truth-values to the variables, constraint 2 is the only one not satisfied. The transition from any existing truth-value assignment to the next is obtained by adding unity to one stage of the counter. Thus, although not all truth-value assignments are tested, those that are tested are taken in the order in which they occur in the binary code. The machine cannot, therefore, test any particular truth-value assignment once its position in the binary code has been passed. Since constraint 2 involves only variables in groups 1 and 2, the addition of unity to any of the stages corresponding to B,D or E would leave constraint 2 unsatisfied. Constraint 2 might be satisfied by adding unity to one of the stages corresponding to C,G,A and F. However, any solutions that might subsequently be found if unity were added to the stage corresponding to F would be missed if, instead, unity were added to a stage of higher significance. Hence, the next transition must consist in adding unity to the least significant stage of group 2. The next assignment of truth-values becomes

CG,	AF,	B,	D,	E
00	01	0	0	0

With this assignment it is found that constraints 2,4 and 5 are not satisfied.

Consider the next transistion. Although constraints 4 and 5 might possibly be satisfied by adding unity to a stage of lower significance, constraint 2 will certainly remain unsatisfied unless unity is again added to the stage corresponding to F, or to a stage of even higher significance. Once again, to ensure that no solutions are missed, unity is added to the stage corresponding to F. The next truth-value assignment is then

CG,	AF,	B,	D,	E
00	10	0	0	0

Constraints 3 and 5 are now unsatisfied. In the following transition, therefore, unity is added to the least significant stage of

group 3, - i.e. to the stage corresponding to B. The resultant truth-value assignment is

CG,	AF,	B,	D,	E
00	10	1	0	0

which does not satisfy constraints 3,4 or 5. In the next transition unity is again added to the stage corresponding to B. This causes a carry to occur to the stage in group 2 corresponding to F. The next assignment is thus

CG,	AF,	B,	D,	E
00	11	0	0	0

Under this assignment of truth-values constraints 3,4 and 5 are still unsatisfied. Unity is therefore added to the stage corresponding to B making the next truth-value assignment

CG,	AF,	B,	D,	E
00	11	1	0	0

Only constraint 5 is now unsatisfied. In the next transition unity is added to the stage corresponding to E bringing the next truth-value assignment to

CG,	AF,	B,	D,	E
00	11	1	0	1

which is found to be a solution.

Since all the constraints are satisfied, the scan is continued by adding unity to the least significant stage of the counter, i.e. to the stage corresponding to E. The next assignment, being

CG,	AF,	B,	D,	E
00	11	1	1	0

leaves constraints 4 and 5 unsatisfied. In the next transition unity is added to the stage corresponding to D making the next truth-value assignment

CG,	AF,	B,	D,	E
01	00	0	0	0

The scan is continued in this way until a point is reached when it is known that no solutions remain to be found. In Figure V.3 the full binary code is shown for the seven variables in this problem. The abbreviated scan is indicated by arrows. The problem has four solutions and these are indicated in the diagram by boxes.

To see how the abbreviated scan terminates consider the penultimate truth-value assignment in this scan. This is

CG,	AF,	B,	D,	E
11	10	1	1	1

Constraint 5 is the only one not satisfied. In the next transition unity is added to the least significant position of the counter bringing the next truth-value assignment to

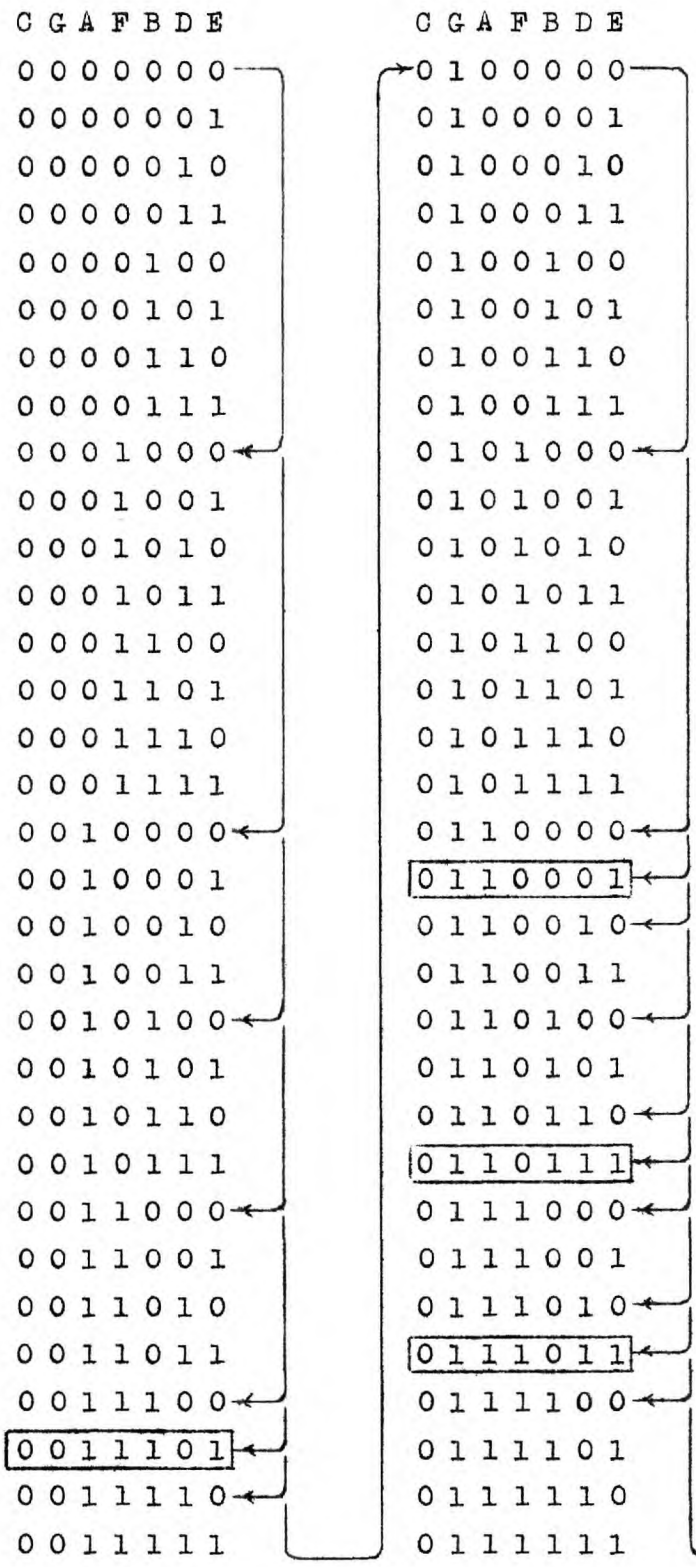
CG,	AF,	B,	D,	E
11	11	0	0	0

This does not satisfy constraints 2,4 or 5. The next transition would consist in adding unity to the least significant stage of group 2 . However, all possible assignments to the variables in groups 1 and 2 have already been tried. Thus, there can be no more solutions.

The exhaustive scan for this problem contains 128 truth-values of which only 33 are considered in the abbreviated scan.

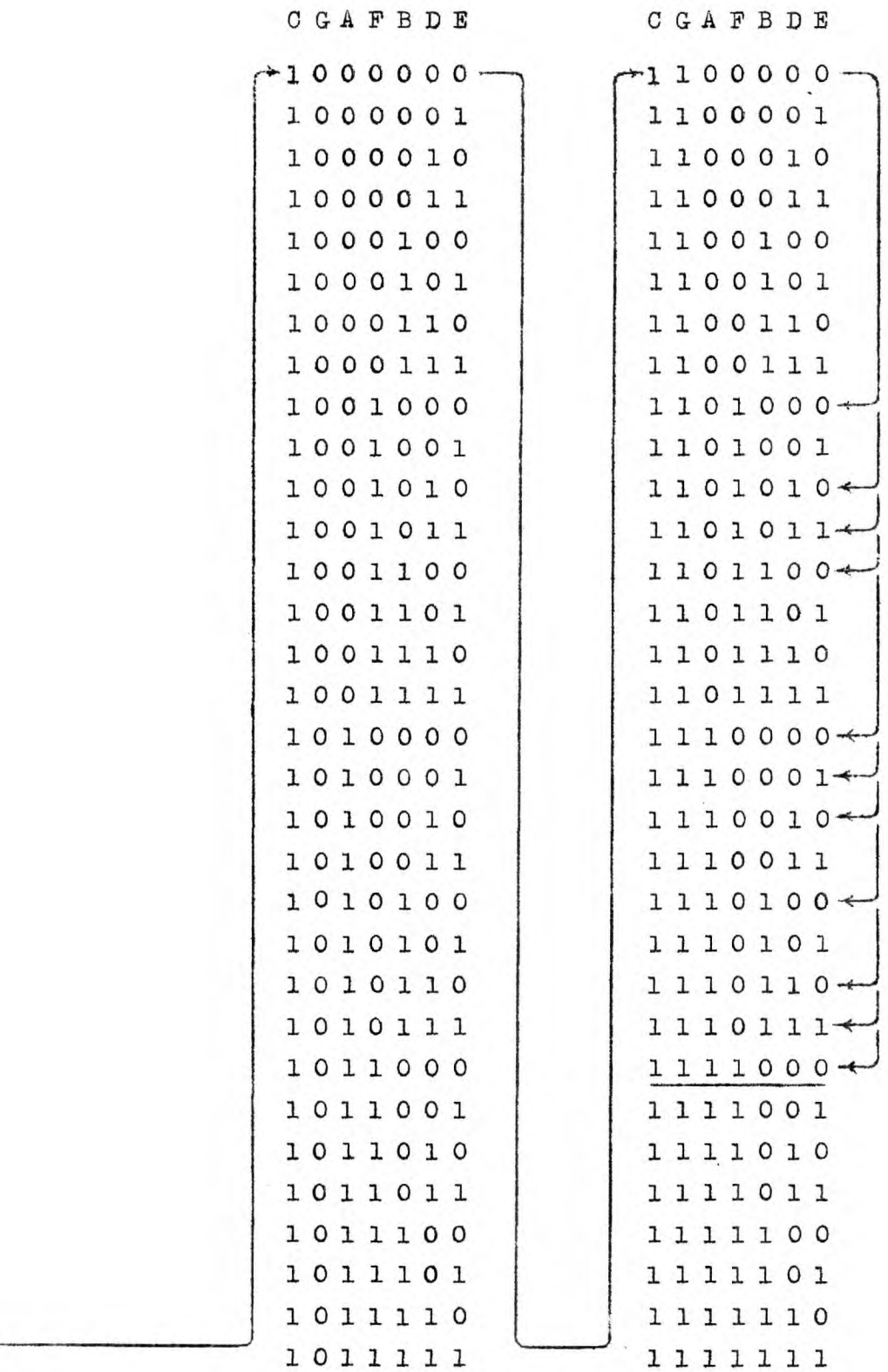
V.3 On the Ordering of Constraints and Variables.

The five constraints in the example just considered were arranged in an arbitrary order. With this ordering of constraints the abbreviated scan considered 33 truth-value assignments. However, if the constraints were arranged in a different order there is no reason to expect that the abbreviated scan would still consider the same number of truth-value assignments. In fact, in general, this would not be the case. In the example in Section V.2, not only were the constraints taken in an arbitrary



COMPARISON OF ABBREVIATED SCAN

FIGURE V.3.



WITH EXHAUSTIVE SCAN

order, but the ordering of variables within the groups into which they were arranged was also arbitrary. For example, maintaining the same ordering of the constraints, the order of the variables of group 1 could have been GC instead of CG.

In what way should the constraints and variables of a problem be ordered if it is desired that the truth-value assignments considered in the abbreviated scan be low in number? Can rules be found regarding the ordering of the constraints and variables which will produce an optimal ordering for which the number of assignments considered in the abbreviated scan is a minimum?

Some light can be thrown upon the first question by noticing that the abbreviated scan always considers all the assignments of truth-values to the variables in the most significant group (which are the variables occurring in the constraint chosen as number one). It would therefore be a bad policy when ordering the constraints to select as number one a constraint involving nearly every variable in the problem. If this were done every possible assignment of truth-values to the first constraint would be considered and the abbreviated scan would certainly contain many trial moves. It would therefore appear advisable when ordering the constraints to select first those in which only a few variables occur.

From the description of the method of generating the abbreviated scan (given in Section V.2) it is evident that an attempt is made to satisfy the constraints one by one, starting with constraint number one and considering the others in numerical order. Only when a given constraint has been satisfied are the less significant variables in higher numbered constraints scanned. Thus, when deciding the constraint ordering it would appear to be advantageous to select first (i.e. allocate low numbers to) any constraints known to have a low <u>proportion</u> of satisifying truth-value assignments. This should tend to produce an abbreviated scan in which large leaps through the binary code occur.

These ideas are discussed in greater detail in Section VII which describes experiments performed on a machine which generates the abbreviated scan to see whether any optimum programming techniques can be found.

V.4. Generalization of the Abbreviated Scan Technique to Many-Valued Combinational Problems.

The abbreviated scan technique can be generalized to deal with combinational problems involving state variables. In this context a state variable is taken to mean anything which may exist in or assume any of a finite number of mutually exclusive states. A state variable may be:

(i) an algebraic variable restricted to assuming any of a finite number of values,
(ii) a propositional variable in two-valued logic,
(iii) a propositional variable in many-valued logic, or
(iv) any other thing capable of existing in a finite number of states, but not included in categories (i), (ii) or (iii).

An example of a state variable in category (iv) is, "the set of tasks to be allocated to an individual working on a factory assembly line".

A state variable can be represented by a symbol that can have assigned to it any one of a finite number of integral values, e.g. a state variable could be represented by the letter V given that V can have assigned to it integral values in the range 0 to R-1. Each value in this range would correspond to a different state of the state variable.

It is not necessary that all the state variables occurring in a particular problem should belong to the same one of the above four categories.

Let $\{V_i\}$, $i = 1,2,----, n$, be the set of state variables in a problem and let $\{v_i\}$, $i = 1,2,----, n$, be the set of values assigned to these state variables. If V_i can exist in R_i different states, then

$$\left.\begin{array}{l} v_i \text{ is integral} \\ \text{and} \quad 0 \leqslant v_i \leqslant R_i - 1 \end{array}\right\} \qquad (1)$$

Let C_1 , C_2 , ---, C_p be the set of constraints that must be satisfied, or, in the case of a constraint in many-valued logic, must have a designated truth-value. Some of these constraints may be arithmetic.

As in Section V.2, the variables are ordered from left to right starting with all the variables occurring in constraint 1. The next variables in the ordering are all those involved in constraint 2 that have not appeared in the ordering so far. As before the variables, thus ordered, are arranged into groups, one group corresponding to each constraint.

Groups	1		p
Variables.	$\underbrace{V_1V_2\text{---}}_{1}$,	--- $\underbrace{\quad}$,---	$\underbrace{\text{---}V_{n-1}V_n}_{p}$
Constraints			

Let N be the <u>mixed radix</u> number formed by the ordered set of assigned values $\langle v_1 v_2 \text{ -----} v_{n-1} v_n \rangle$. From (1) it is seen that the radix of v_i is R_i , so that the value of this number is given by

$$N = \sum_{i=1}^{n} \left[v_i \cdot \prod_{j=i+1}^{n} R_j \right] \qquad (2)$$

where $\prod$ is the product sign. N can assume any integral value in the range 0 to $N_{max.}$ where the maximum value is given by

$$N_{max.} = \left[\prod_{j=1}^{n} R_j \right] - 1 \qquad (3)$$

Each value of N in this range corresponds to a unique assignment of values to all the state variables in the problem. A certain set of values of N therefore correspond to the solutions of a given problem.

The abbreviated scan through the range of values of N commences with assigned values to all the state variables being zero, i.e.

$$v_i = 0 \quad \text{for} \quad i = 1,2,\text{---}, n \, .$$

A transformation, T_i, is defined corresponding to each problem constraint C_i. Let the $r^{th.}$ assignment of values in the abbreviated scan be $\{v_i\}_r$. The transition from the $r^{th.}$ to the $(r+1)^{th.}$ assignments in the abbreviated scan is obtained by operating on the assignment $\{v_i\}_r$ with one of the defined transformations, T_α, to give the assignment $\{v_i\}_{r+1}$.

If $\overline{C_i}$ represents NOT-C_i, for $i=1,2,---,p$, meaning that the $i^{th.}$ constraint is not satisfied, then the transformation, T_α, to be used upon any existing assignment $\{v_i\}_r$ in the next transition is given by

$$\alpha = \text{MIN.}\left[1.\overline{C_1}, 2.\overline{C_2}, ---, i.\overline{C_i}, ---, p.\overline{C_p}\right] \qquad (4)$$

If C_i is <u>not</u> satisfied, then $\overline{C_i}$ has the value unity, otherwise $\overline{C_i}$ has the value zero.

For the particular case of problems in two-valued logic treated in Section V.2 each number, v_i, is a binary digit and each transformation, T_i, has the effect of adding unity to the least significant position of the $i^{th.}$ group of variables.

In the generalization considered here the transformations, T_i, are, in the most general case, functions of those values in the existing assignment, $\{v_i\}_r$, corresponding to the state variables in groups 1 to i. If the value of N corresponding to the $r^{th.}$ assignment, $\{v_i\}_r$, is N_r, then the T_i are defined so that

$$N_{r+1} > N_r \qquad (5)$$

Some of the problem constraints may be embodied in the transformations, T_i, in which case these constraints would <u>not</u> appear in the set of explicitly stated constraints, $C_1, C_2, ----, C_p$.

Naturally, the set of transformations, T_i, are always defined so that it is impossible for any transition in the abbreviated scan to cause any solutions to the problem to be omitted.

In Section VIII these ideas are used in demonstrating one possible method of solving the problem of constructing an academic timetable.

VI. MECHANIZATION OF THE ABBREVIATED SCAN FOR PROBLEMS IN TWO-VALUED LOGIC

VI.1. The Purpose of the Machine.

In Section V.2 the method of generating an abbreviated scan of the truth-value assignments to two-valued problems was explained. Such a scan includes all the solutions to any given problem while considering only a fraction of the 2^n assignments to a problem in n variables. In the previous Section ideas were discussed of finding optimum programming techniques for an abbreviated scan machine so that, for a given problem, a scan might be obtained containing a minimal number of trial assignments. The purpose of building a machine is to enable experiments to be conducted to investigate these programming techniques.

The possibility was considered of programming a general purpose digital computer to simulate an abbreviated scan machine, and of making the investigation without actually constructing a machine.

In the proposed investigation numerous problems would need to be considered and it would be necessary to solve each problem many times over using different orderings of the variables and constraints in the problems. In an investigation of this kind it is best to consider problems having relatively few solutions. How are such problems to be found? The simplest way is to construct problems arbitrarily and to solve them. If any problem has too high a proportion of solutions its constraints can be modified or extra constraints can be added and the problem can be solved again. This process can be repeated until the number of solutions has been reduced to a satisfactory level.

If this investigation were carried out using a digital computer one sub-routine would be used embodying the logical constraints of the problem being considered. It is obvious that this sub-routine would require

continual alteration and modification as the variable and constraint orderings were changed and as the constraints themselves were modified. This would be a very time consuming business and the likelihood of errors occurring as numerous computer tapes were punched would be quite high. In addition, the cost of computer time would probably be of the same order as the cost of building a special purpose machine.

If a machine were built specially for the investigation the many changes in the ordering of variables and in the problem constraints would present no difficulty if a plug-board arrangement were used for setting up the problems.

The final reason for constructing a machine was that, as this work was being carried out in an Engineering Faculty, it would be a far more useful exercise than another computer programming session.

VI.2 The General Requirements of the Machine.

The basic requirements of the machine were:

(i) It should be capable of solving problems having a fair degree of complexity since it is possible that the advantage of an abbreviated scan machine might not be evident when solving simple problems.

(ii) It should have logical connective boxes set out in a plug-board arrangement so that problems may be easily set up or modified.

(iii) The logical connective boxes should be compatible so that many units may be connected in cascade, the output of one unit being connected to the input of the next.

(iv) The machine should have some facility for counting the number of trial truth-value assignments in any scanning sequence.

Another of the chief factors governing the design was the necessity to keep the overall cost of the machine as low as possible.

Bearing this in mind and considering requirement (i) it was decided that sufficient complexity should be obtainable if the machine could solve problems involving 16 logical variables. If there were fewer variables than this it would be difficult to make observations such as the effect of overlapping constraints upon the performance of the machine.

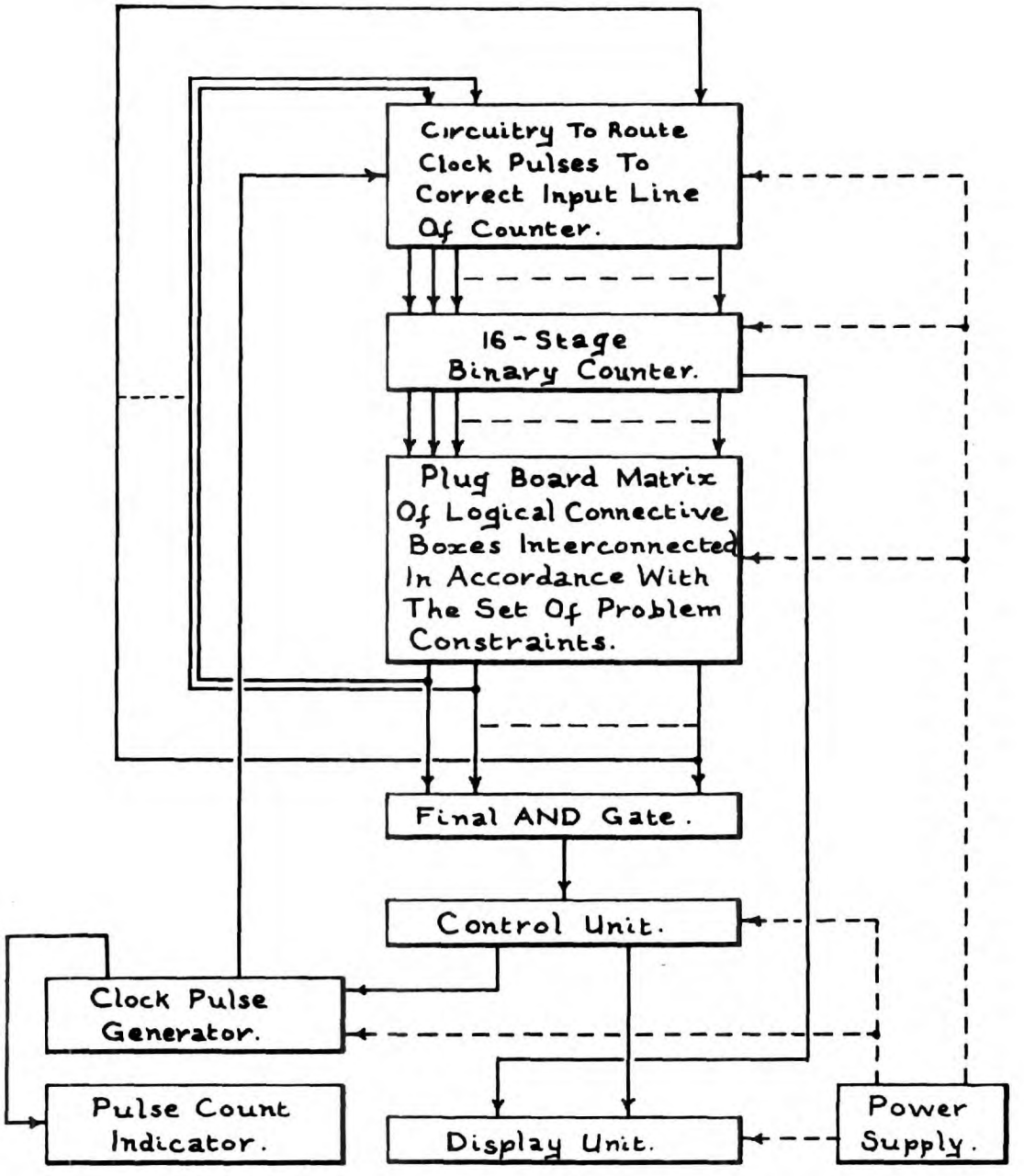
Relays were chosen as the basic elements of the machine since the compatibility of the logic units then presents no difficulty. If valves or transistors had been used it would have been necessary to use buffer stages such as cathode followers and possibly shaping amplifiers to reshape pulses when they had passed through two or three logic units. Another advantage of using relays is that all the logical functions can be obtained easily whereas with valves or transistors some of the functions cannot be obtained using one element.

It was difficult to estimate how many of each of the connectives AND, OR, IMPLICATION, etc., would be required. This was overcome by using universal logical units, each unit being capable of functioning as any one of the logical connectives as required. The increased flexibility gained by using universal elements meant that, for a given total number of connective boxes, a wider variety of problems could be put on the machine than would have been possible if each connective box performed only one function.

One hundred universal logical elements were used. This number of elements was used partly because the budget would not permit a higher number and partly because the reliability of a machine is likely to vary inversely with its complexity.

VI.3 Block Diagram of the Complete Machine

The block diagram of the machine appears in Figure VI.1. The machine having been set into action by the operator, the train of pulses from the clock-pulse generator is fed via routing circuitry to the input wire of one of the 16 stages of the binary counter. The states of the



——— WIRES CARRYING INFORMATION.
– – – – – WIRES CARRYING POWER.

BLOCK DIAGRAM OF ABBREVIATED SCAN LOGICAL COMPUTER.

FIGURE VI.1.

output wires of the counter at any instant represent the existing assignment of truth-values to the 16 logical variables available. The output wires from the counter are connected to the input sockets of logical connectives boxes which are interconnected in accordance with the logical constraints of the problem. Corresponding to each constraint is a main connective box, the output wire of which is held at a positive potential when that constraint is satisfied and at earth potential otherwise. These constraint output wires are each connected to one input socket of a final AND gate. The signals on the constraint output wires are also fed back to control the routing circuitry so that each successive pulse from the clock-pulse generator is directed to the correct stage of the binary counter. This results in the binary counter going through the sequence of states required to generate the abbreviated scan of truth-value assignments.

When a truth-value assignment corresponding to a solution is set up the output wire of the final AND gate becomes energized. This signal is passed, via the control unit, to the clock-pulse generator to inhibit its operation until the machine is once again set into action by the operator. At the same time the existence of a solution is indicated by the lighting of a lamp in the display unit. Another set of lamps in the display unit enable the assignment of truth-values constituting the solution to be read from the machine. The total number of transitions that have occurred in the abbreviated scan at this stage can be read from the pulse count indicator.

When the scan has reached the stage where it is known that no solutions remain to be found the machine stops and an orange indicator lamp is lit.

VI.4 A Binary Counter with Simultaneous Carry Propagation Between All Stages.

The heart of the scanning mechanism consists of a 16 stage binary counter. Several conventional counting circuits were tried and the one

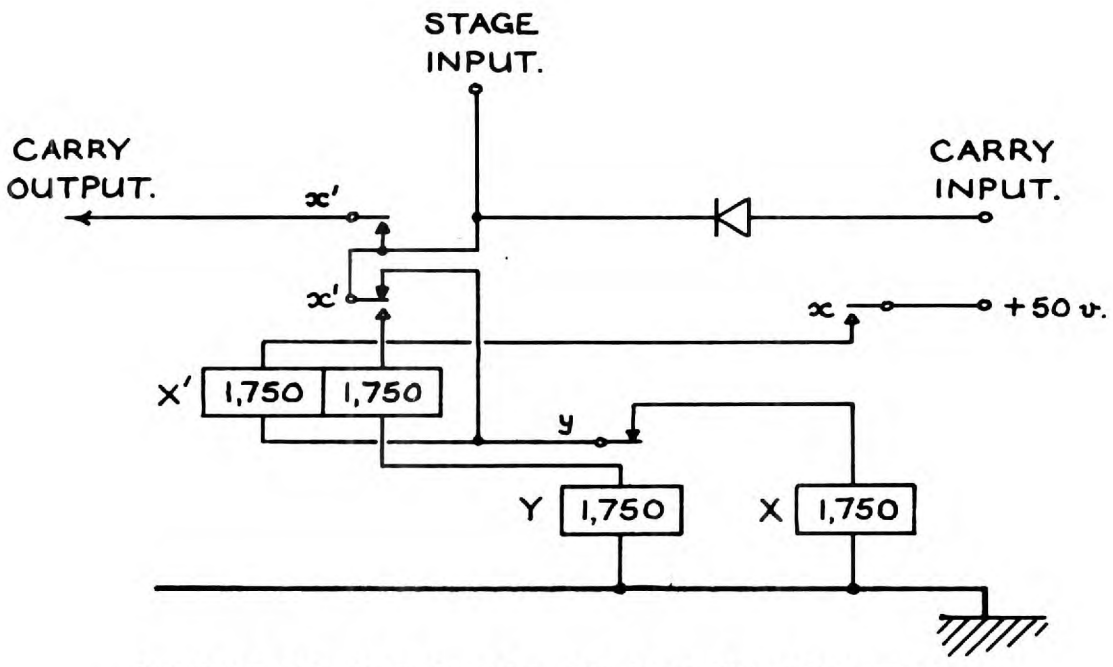
ultimately used was based on a circuit given in reference 9 . The circuit diagram for one stage of the counter is given in Figure VI.2a. Counting circuits can be devised using only two relays per stage but the operation of such circuits generally involves de-energizing a relay by shunting its coil. This tends to limit the speed at which such circuit arrangements will operate. Although speed of operation was not a major target it was felt that it was worth using a counter with one extra relay per stage to obtain a higher scanning rate.

The relays have 1,750 ohm coils, relay X' having two coils each of this resistance. The normal voltage used to energize the relays is 50 but half this voltage is actually sufficient, though the operating speed of the relays at this reduced voltage is lower.

Input pulses are fed to a given stage of the counter either directly, on the stage input line, or on the carry input line from the next stage of lower significance. The diode in the carry input line prevents carry pulses from being fed backwards through the counter. Figure VI.2b illustrates the sequence of operations as successive input pulses are fed to the counter stage. The state of the counter stage during each period is indicated by a digit 0 or 1 at the top of the sequence diagram.

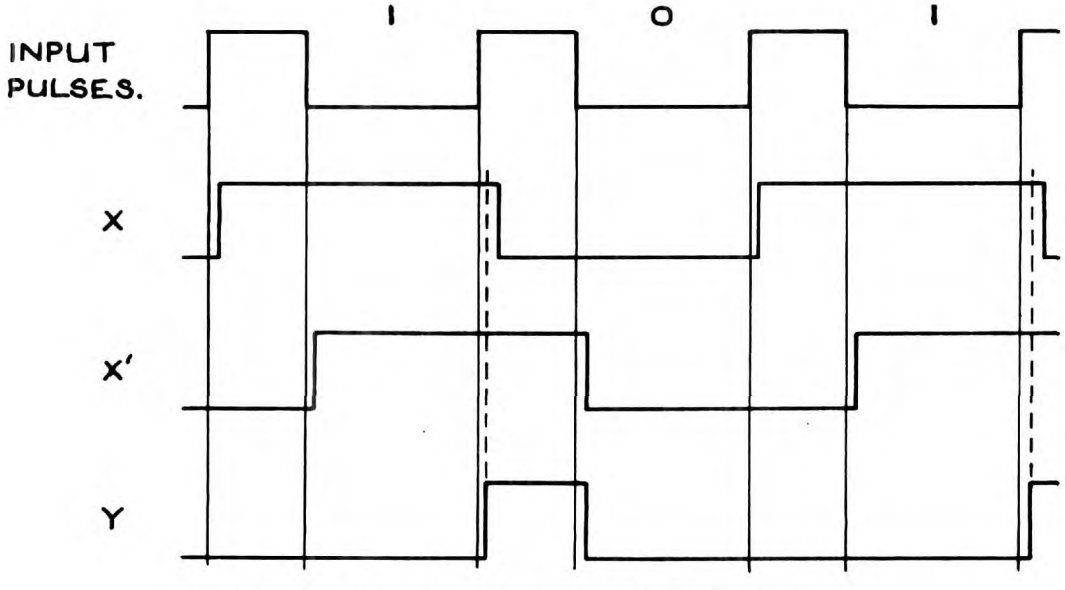
Since the operation of this counter is fully described on p. 244 of reference 9 it will not be dealt with here. However, the method used here to derive carry pulses does not appear to be mentioned in the literature. In any case, if it is, the idea in this instance occurred independently. This will now be explained.

Consider the first few combinations of the binary code (Figure V.3, p. 88 .) and imagine that a counter is going sequentially through these states. Note that if a pulse is added to the counter when the least significant stage is in the zero state no carry pulses occur. Carry pulses occur when a group of one or more successive stages at the least significant end of the counter are in the one state when a pulse is added to the counter. For example, if the state of the counter is 0000111 when



CIRCUIT OF ONE STAGE OF BINARY COUNTER.

(a)



OPERATION SEQUENCE DIAGRAM.

(b)

FIGURE VI.2.

another pulse is added in, then the subsequent state of the counter is 0001000. Note that the group of successive ones at the least significant end of the binary number are each replaced by zero and that the zero immediately following this group of ones is replaced by unity.

With the conventional carry pulse arrangement in binary counters, if a pulse from the clock-pulse generator is added to the least significant stage, causing this stage to change from the one state to the zero state, then after this change has occurrred, a carry pulse is passed on to the next stage of higher significance. A similar delay in the propagation of clock-pulses occurs at each successive stage. The operating speed of a multi-stage counter is therefore limited by the time required for a carry pulse to be propagated from one end of the counter to the other.

By studying the circuit in Figure VI.2a it is seen that with the arrangement used here, immediately after a given clock-pulse is added to the counter the complete route is established for the carry pulse that will occur when the next clock-pulse is added to the counter. This route consists of a path through a series of make contacts of successive X' relays. In this way the carry pulse supplied to any stage is the actual clock-pulse being added in to the counter and carry pulses occur at all the required stages simultaneous with the occurrence of this clock-pulse.

Although the operating speed of the counter is not limited by the carry pulse propagation time the circuit arrangement used here has one disadvantage. This is the number of successive relay contacts through which carry pulses must pass. This can cause an attenuation of the carry pulse which, if the circuit were operating under marginal conditions, could upset the operation of the counter. However, no such trouble was experienced in this case.

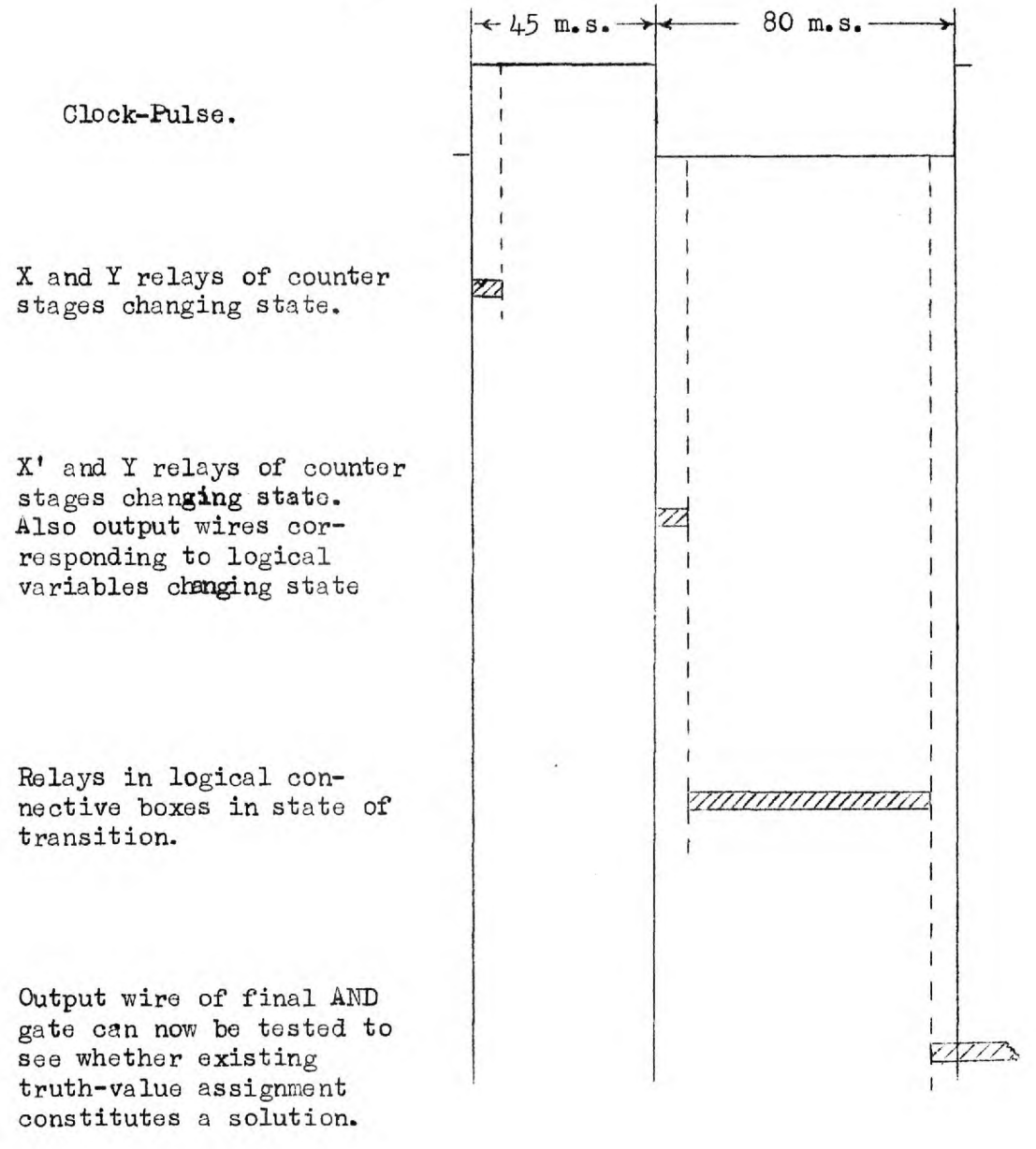
Post Office type 100 relays were used in the counter. These have two energizing coils and two sets of change-over contacts. The operating times for these relays are claimed to be of the order of two milliseconds. However, as used in this machine the relays were found to have operating times between five and eight milliseconds.

The counter was found to operate very satisfactorily at a counting rate of ten pulses per second, which speed it was not desired to exceed.

VI.5. The Operation Cycle of the Machine.

The clock-pulse rate of the machine is 8 c.p.s.. The clock period is therefore 125 milliseconds of which time the clock-pulse occupies 45 milliseconds. This makes the mark/space ratio approximately 1 : 2 . Figure VI.3 shows the way in which the clock period is used in performing the cycle of events through which the machine works. This machine is unusual in that the frequency of the basic operation cycle of the machine is the same as the clock-pulse frequency; in most machines it is a sub-harmonic of the clock-pulse frequency, several clock-pulses occurring during each basic cycle of operations.

During the first few milliseconds following the initiation of a clock-pulse the X and Y relays in some of the stages of the counter are changing state. For a similar period following the trailing edge of a clock-pulse the X' and Y relays in the same stages of the counter are changing state. The output wires whose states at any instant represent the assignment of truth-values to the logical variables are connected to make contacts on the X' relays (see Fig. VI.2). Therefore, the states of these output wires are also changing during the few milli-seconds following the trailing edge of a clock-pulse. Following this change of state to a new assignment of truth-values many of the relays in the logical connective units are in a state of transition. As some logical constraints require a string of connective boxes to be connected with the output of one box connected to the input of another this state of transition may persist for a considerable time. Some 70 milliseconds, being the major portion of the operating cycle, are reserved to allow for this. Shortly before the arrival of the next clock-pulse, when the logical connective units have "settled down", the output wire of the final AND gate (Fig. VI.1) can be examined to see whether the existing truth-value assignment constitutes a solution to the problem on the machine.



OPERATION CYCLE OF MACHINE

FIGURE VI.3.

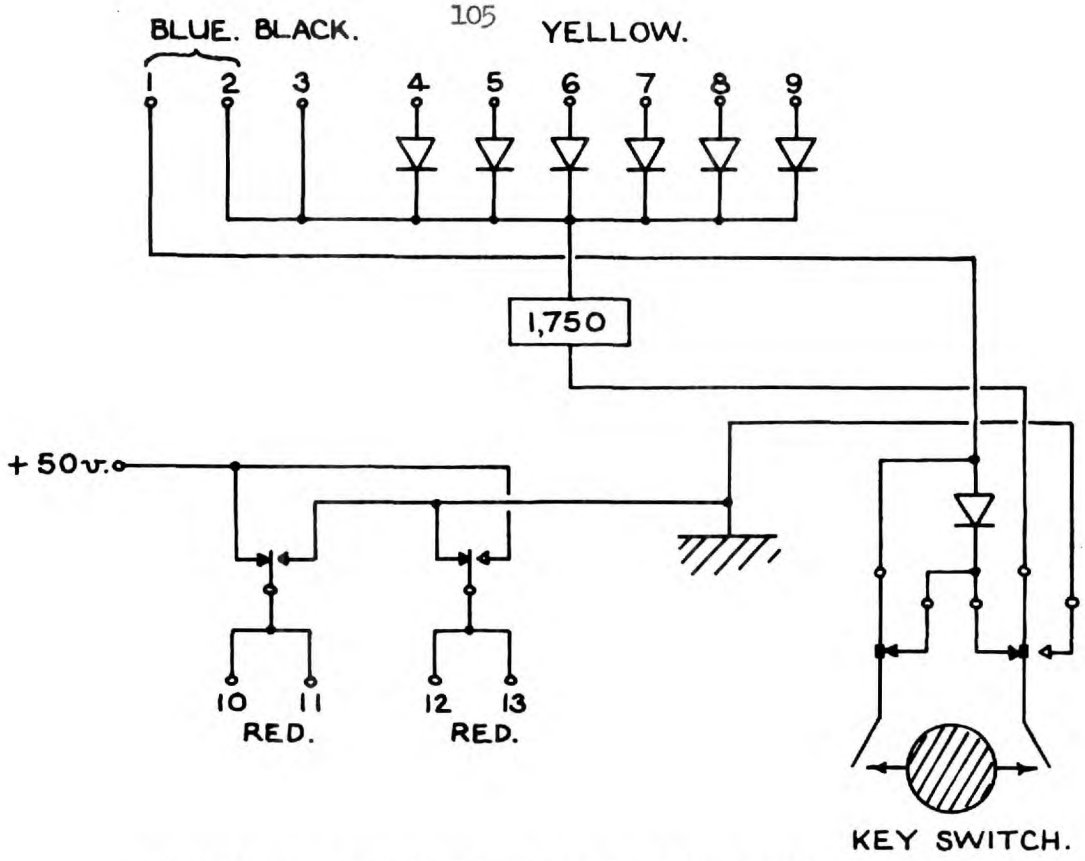
VI.6. The Logical Connective Units

It was stated in Section VI.2 that universal logical connective units are used. Actually the units are not quite universal. There are 16 possible logical connectives for two variables. Half of the units, 50 in number, can be used to obtain any of the connectives except AND while the other 50 units provide any connective except OR. This was done mainly to save money but also to stop the density of input and output sockets on the front panel of the machine from becoming too high.

The circuit of the units providing all connectives except the AND connective appears in Figure VI.4a. In the first of the papers on logical computers by McCallum and Smith (ref: 5 .) circuits were given for individual logical connective units each performing the function of one connective. These were combined with a telephone key switch and several sets of input and output sockets to form the circuit of Figure VI.4a.

As the operation of the individual circuits is explained in reference 5 , and, in any case, the circuit is self explanatory, such explanation will be omitted. It should, however, be noted that input sockets and output sockets are held at +50 volts to represent a truth-value of unity and at earth potential to represent a truth-value of zero. The table in Figure VI.4b indicates which input and output sockets are used and also the position of the key switch to obtain various logical connectives. The NEGATION of any given logical connective may be obtained merely by using the alternative set of output sockets. The colour of the various sets of sockets is indicated on the diagram.

The circuit of the other 50 units is very similar but a six input AND gate replaces the OR gate of Figure VI.4a. The input sockets for the AND connective on these units are green. The two types of units are arranged alternatively in a chess board fashion on the front of the machine as can be seen in the photograph of the machine in Figure VI.6. The units are mounted on racks in sets of five as can be seen in ~~the two views in~~ Figure VI.5.



CIRCUIT OF LOGICAL CONNECTIVE UNITS.

(a)

FUNCTION.	POSITION OF KEY SWITCH.	INPUT SOCKETS.	OUTPUT SOCKETS.
NEGATION.	RIGHT.	3.	10 AND 11.
INCLUSIVE OR.	RIGHT.	4 TO 9.	12 AND 13.
NON-IDENTITY.	CENTRE.	1 AND 2.	12 AND 13.
IDENTITY.	CENTRE.	1 AND 2.	10 AND 11.
IMPLICATION.	LEFT.	1 AND 2.	10 AND 11.

(b)

FIGURE VI.4.

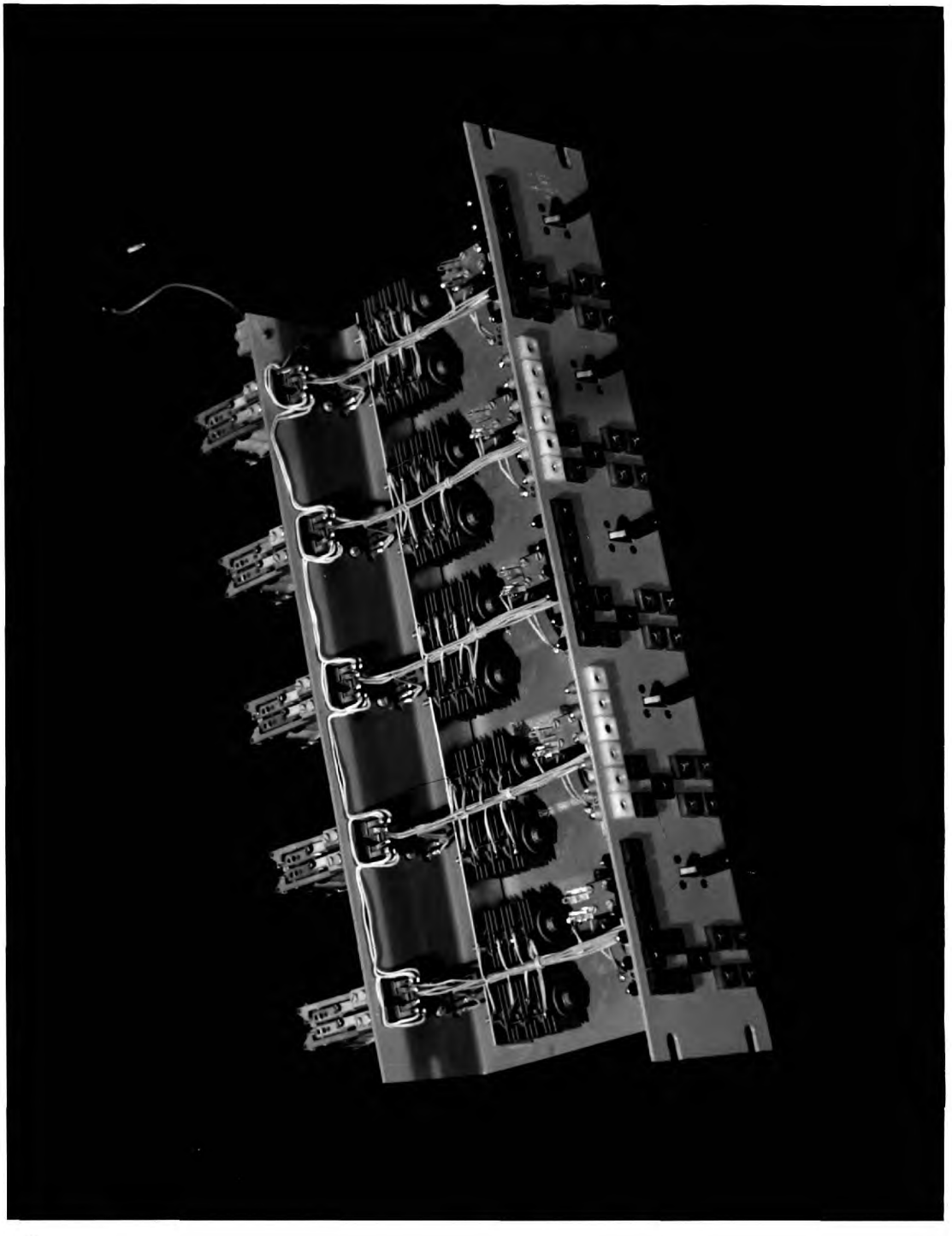


FIGURE VI.5.

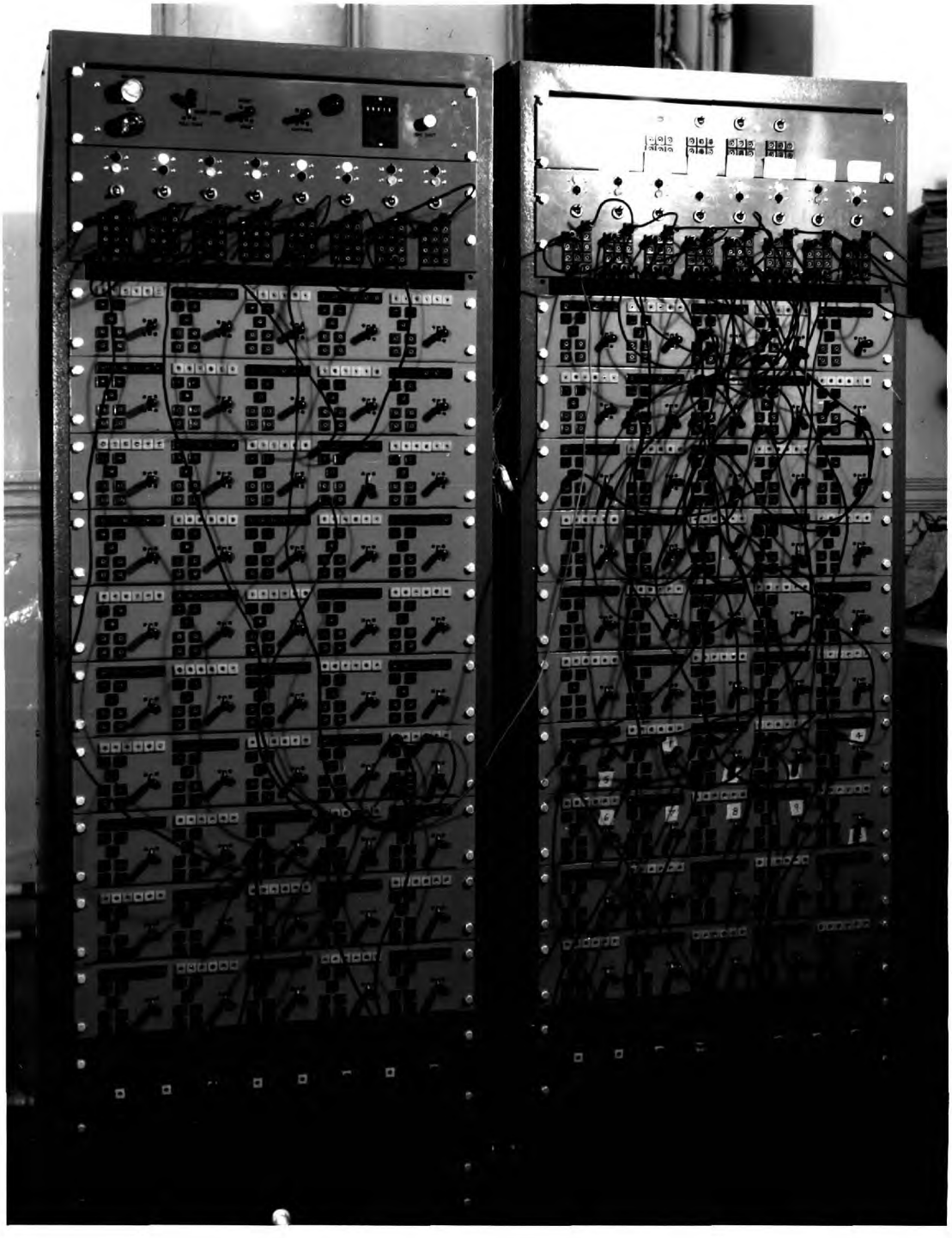


FIGURE VI.6.

VI.7. Operation of the Complete Machine

The circuit diagram of the complete machine including the scanning mechanism, clock-pulse generator and control unit, but excluding the logical connective units, is given in Figure VI.7 (to be found in pocket inside front cover).

Three of the sixteen stages of the binary counter are shown in the upper part of the diagram.

The 50 volt power supply is obtained using a stack of germanium diodes in a full-wave bridge rectifying circuit. Although the operation of the machine does not require the full output of 12 ampères which these diodes can supply the power pack has been designed to make this available. Even so, quite high currents will be demanded and, for this reason, only capacitance smoothing is used. The 20,000 microfarads smoothing should give 10% ripple at a load of 10 ampères and correspondingly less at lower loads. When the machine is switched on the smoothing capacitor acts as a dead short-circuit and the current surge is limited by resistor R34 which is cut out a fraction of a second later when the slugged relay, E, operates. A 30 volt line is taken from the junction of R33 and C5 to supply the transistors in the single-shot circuit in the control unit and those in the clock-pulse generator.

The clock-pulse generator is a simple free-running multivibrator using two OC71 transistors. Some control of the mark/space ratio and frequency of the pulses is obtained by adjusting RV1 and RV2. Relay C forms part of the load in the collector circuit of transistor T4. The clock-pulses which time the machine's operation are taken from the 50 volt supply through a make contact of relay C. At the instant of switching the machine on the base of T4 is connected, via D24, R31 and a break contact of relay G, to the +50 volt line while, for the moment, the "+30 volt" line is floating. After a fraction of a second slugged relay E operates and the +30 volt line is, in fact, held at +30 volts. This arrangement ensures that the 50 volt bias is applied to the base of T4 and that the clock-pulse generator is locked with T4 in the non-conducting state.

The machine remains in this state until further action by the operator.

With the key switches in their normal position the machine is ready to operate in the abbreviated scan mode. To reset the machine key K_2 is pressed in an upward direction operating the contacts K_{2u1} and K_{2u2}. This energizes the reset coil of the pulse count indicator relay and so resets the indicator to zero. Simultaneously relay F operates causing the heavy duty break contact, through which the power supply to the rest of the machine is taken, to open. This caters for the eventuality of the machine being reset while in operation. The removal of the supply causes all the energized relays in the binary counter to "drop off", thus leaving all stages of the binary counter in the zero state.

To start the machine key K_2 is pressed in a downward direction operating the single contact K_{2D}. Relay G is energized and remains energized by virtue of the "hold on" path through one of its make contacts. The path supplying the 50 volt bias to the base of T4 is now broken. The clock-pulse generator therefore comes into operation. The clock-pulses are fed along a line to the routing circuitry which directs them to the correct input lines of the binary counter. The clock-pulses are also used to energize relay H. A make contact of H applies energizing voltage to the stepping coil of the pulse count indicator relay.

At the top of the circuit diagram, corresponding to each of the 16 stages of the binary counter, is a rule socket. These 16 sockets are the input sockets of the final AND gate of the machine (see Fig. VI.1 on p. 97).

As explained in Section V.2, the logical variables are ordered and arranged into groups. The variables in group 1 are all those appearing in constraint 1 and they are represented by a corresponding group of stages at the most significant end of the binary counter. The variables in group 2 are all those appearing in constraint 2, but not already contained in group 1. These are represented by a corresponding group of the most significant of the remaining stages of the counter.

Now, corresponding to each constraint is a main logical connective unit, the output socket of which is held at +50 volts when the constraint is satisfied and at earth potential otherwise. The output socket of the main connective box of each constraint is connected to the rule socket of the least significant of the group of stages of the binary counter corresponding to that constraint.

Example: the output socket of constraint 5 would be connected to the rule socket of that stage of the counter corresponding to the least significant variable in group 5.

Since each group contains only one least significant variable, some of the rule sockets will be left "floating". However, note that the Z relays corresponding to these floating rule sockets are energized since they are each connected, in series with a 2.2 kilohm resistor, across the 50 volt supply. Note, also, that the Z relay corresponding to a rule socket which is not floating is energized when, and only when, the corresponding constraint is satisfied.

The path of the clock-pulses to the counter can now be traced. The clock-pulses pass through the break contact, K_{1D2}, of key switch K_1 and then come to a line containing a series connection of change-over contacts corresponding to the Z relays. The make contact corresponding to any floating rule socket is closed so that the clock-pulses are passed to the next change-over contact in the series. In this way each clock-pulse is routed to the input line of the least significant stage of the group of stages corresponding to the first unsatisfied constraint encountered by the pulse. Referring to Section V.2 it is seen that this is just what is required to generate the abbreviated scan of truth-value assignments.

When an assignment is generated for which all the constraints are satisfied the next clock-pulse passes through the make contact of every Z relay and then passes through break contact, K_{1D1}, of key switch K_1 to the line marked "SOLUTION". The clock-pulse then passes through diode D18 and the break contact, K_{2U2}, of key switch K_2 to trigger the

single-shot circuit. A short pulse appearing at the collector of T2 energized relay A which holds itself on. A make contact of relay A completes the path from the coil of relay B to the +50 volt line, thus energizing relay B. As a result of this a bias of +50 volts is applied to the base of transistor T3 which stops the action of the clock-pulse generator with T4 in the conducting state. The operation of relay B also causes a red signal lamp in the display unit to be lit, indicating that a solution has been found.

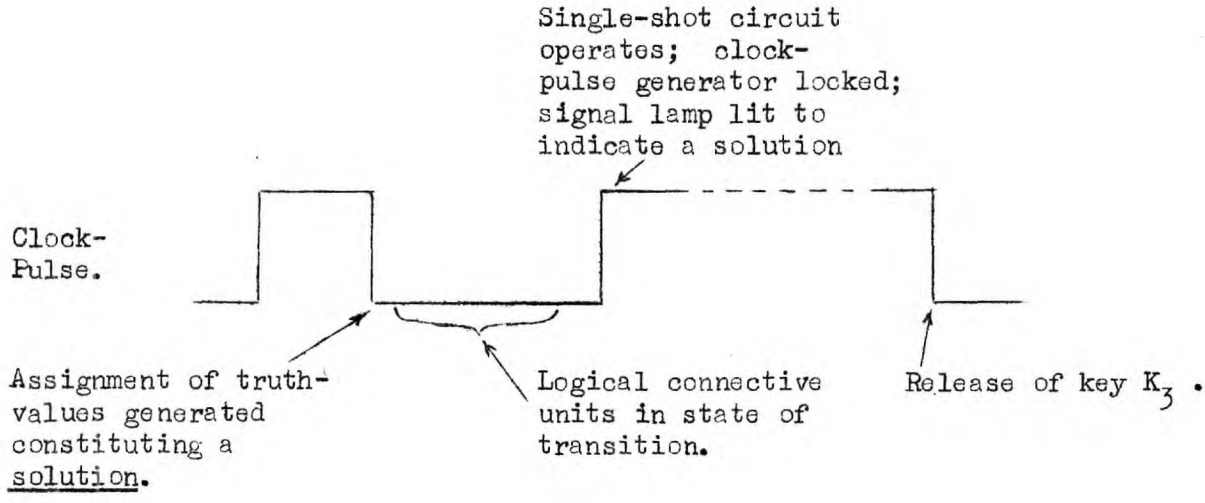


FIGURE VI.8.

Figure VI.8 shows the sequence of events which occur when a solution is found. Note that the action of the clock-pulse generator is not inhibited until after the initiation of the clock-pulse for the subsequent cycle of operations. This is because the clock-pulse itself is used to test the continuity of a path which, when complete, indicates a solution.

The machine might be left for quite a time before the operator presses the key which makes it continue its scan. During this interval the stepping coil of the pulse count indicator would be energized and this might damage the coil. To prevent this resistor R32 is put in series with

the coil of relay H and, when a solution exists, the following clock-pulse is routed through diode D17 to the junction of R32 and the coil of relay H. The coil of relay H therefore has both its ends held at +50 volts and consequently it remains de-energized. Thus, when a solution is found, the pulse count indicator gives the number of transitions made by the machine up to that point in its scan.

When the operator presses the key, K_3, in a downward direction to make the machine continue its scan relay A drops off but relay B remains energized while the key, K_3, is held in the downward position. The break contact K_{3D2} is open while the key is depressed and the junction of R32 and the coil of relay H is no longer maintained at +50 volts. The stepping coil is therefore energized while key K_3 is depressed. On releasing key K_3 it returns to its normal position and relay B drops off allowing the clock-pulse generator to continue operating. The machine continues scanning until it finds another solution or comes to the end of the scan.

The scan is terminated when a carry pulse occurs on the carry output line of the most significant stage of the binary counter. This indicates that all possible assignments of truth-values to the variables in the most significant group have been tried, although another pulse is being added to this group. If the machine were allowed to run on it would repeat the same scan. In order to stop the machine the carry pulse occurring on the carry output line of the most significant stage goes along the line marked "STOP", through diode D19, to trigger the single-shot circuit and stop the clock-pulse generator. This stops the machine in the same way as when a solution is found. The carry pulse also passes through diode D20 to energize relay D which holds itself on and lights an orange signal lamp in the display unit to indicate the termination of the scan. Unless the last truth-value assignment in the scan is a solution, no pulse is fed through diode D17 to the junction of R32 and the coil of relay H. Relay H is therefore energized in the absence of a solution and the opening of one of its break contacts prevents the red signal lamp from being lit and giving a false indication of the existence of a solution.

Two other modes of operation of the machine are possible. If key K_1 is left in the upward position the machine can be operated manually. The scan is made to progress one transition at a time by repeatedly depressing the push button, B_1. In this mode of operation the clock-pulse generator is out of action, the clock-pulses being derived from the 50 volt supply through a make contact of the button B_1. When a solution is found the button can be pressed but it will have no effect because it is then by-passed by a make contact of relay B. This prevents a solution from inadvertently being overshot. To continue the scan key K_3 must be pressed in a downward direction for the first transition. Further transitions are obtained by reverting to the push button, B_1. In this manual mode of operation the machine performs the abbreviated scan.

If key K_1 is left in the downward position the machine again operates automatically, the clock-pulse generator once more coming into action. However, clock-pulses are now directed always to the input line of the least significant stage of the binary counter. The machine therefore goes through an exhaustive scan of the truth-value assignments. This mode of operation is provided for means of comparison.

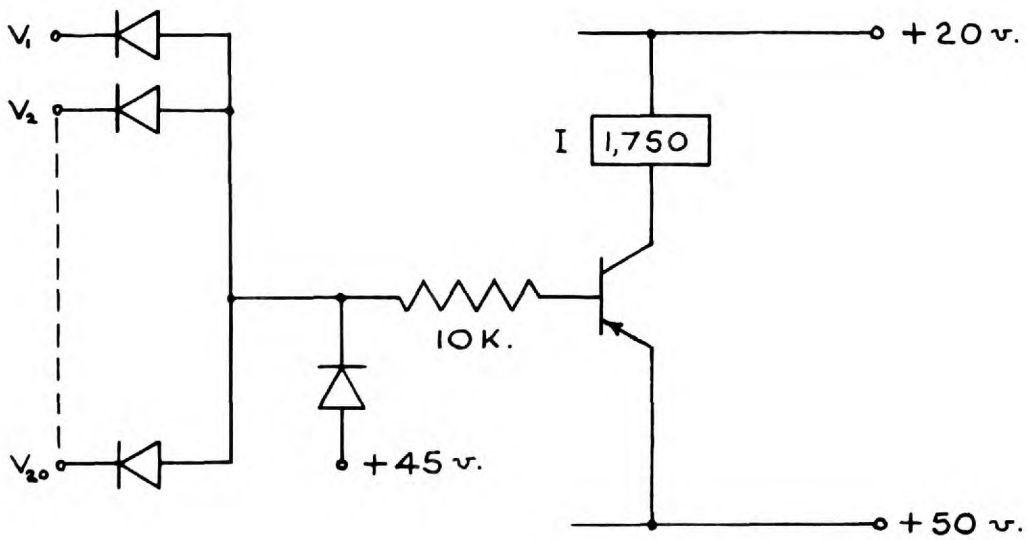
Above each stage of the binary counter (in the circuit diagram) there is a switch (S1 to S15), corresponding to that stage. If it is desired to put a problem on the machine involving fewer than 16 logical variables, then, for an n variable problem ($n \leqslant 16$), the least significant n stages of the binary counter must be used. Then, so that the unused variables do not enter in to the scan the switch, SN, corresponding to the N^{th} stage is closed (all the other switches being open). A pulse is then fed along the "STOP" line when a carry pulse appears on the carry output line of the N^{th} stage of the counter.

VI.8 A Fuse Indicator

As mentioned previously, the logical connective boxes are mounted on racks with five units on each rack. The same type of high-speed relays (P.O. type 100) are used in the connective units as in the binary counter.

The gap between open contacts on these relays is very small and, with the machine working in an atmosphere saturated with dust, it is not an infrequent occurrence for a relay to "arc over". To prevent the contacts from welding together when this happens each rack of five connective units has its 50 volt supply line individually fused. As the fuses are rated at 750 mA., while the current carrying capacity of each relay contact is one ampère, the contacts are well protected. However, when the fuse on one of the racks blows it is still possible for current to be supplied to the connective units on that rack through the connections to the input sockets of the units. Consequently the potential of the supply rail of a fused rack will not necessarily fall to earth level; it may float to a level as high as 40 volts above earth potential! In such a case there would be no indication that a fuse had blown and the machine would continue operating and getting all the wrong answeres. In an effort to avoid this a fuse indicating system has been installed. The circuit of this appears in Figure VI.9.

The emitter of the transistor is held at the level of the +50 volt line of the control unit. The +20 volt line and the +45 volt point are obtained by using a simple resistor dividing chain. The cathodes of the 20 diodes are connected to the +50 volt lines of the 20 racks of connective units on the connective unit side of each fuse. This circuit compares the potential level of the supply line of each rack of connective units with that of the +50 volt line in the control unit. Normally the transistor is conducting very little and relay I is not energized. However, if the potential of the supply line of any one of the 20 racks of connective units falls as much as four or five volts below that of the control unit the transistor conducts and relay I operates. This lights a green signal lamp in the display unit indicating that a fuse has blown. It is then a simple matter to test the supply line of each rack with a voltmeter to discover which rack is at fault.



CIRCUIT OF FUSE INDICATOR.

FIGURE VI.9.

VII. AN INVESTIGATION OF METHODS OF PROGRAMMING THE ABBREVIATED SCAN MACHINE

VII.1 The Ideas to be Tested

The abbreviated scan machine constructed for solving problems in two-valued logic was described in Section VI. In Section V.3 some ideas were discussed regarding the possibility of finding an optimal ordering of the constraints and variables of a given problem when programming the machine so that the length of the machine's scan is minimized.

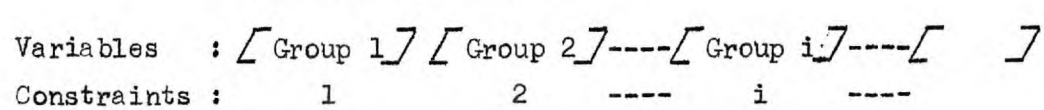
The n variables of a problem may be ordered in n-factorial different ways. However, when programming the abbreviated scan machine the constraints are first ordered and this partially determines the ordering of the variables; the variables are arranged in groups, each group corresponding to one or more constraints. Thus, not all the n-factorial orderings of the variables are possible.

The only factors that may be considered when ordering the constraints are:

(i) the number of variables involved in each constraint,
(ii) the logical form of each constraint and
(iii) the relative overlap of the constraints.

The groups of variables are numbered, as below, starting with number one at the left-hand side. The variables in group 1 are all those involved in the constraint chosen as number one when ordering. All possible assignments of truth-values to the variables in group 1 will be tried in the scan.

Most significant end



In describing the operation of the machine it was noted that clock-pulses are fed to the stages of the counter corresponding to the least significant variable in each group, being routed according to which constraints happen to be satisfied. When an input pulse is fed to a stage of high significance (corresponding to a variable near the left-hand end of the above ordering) the scan takes a large leap through the binary scale. For the scan to be kept short it is desirable that this should occur as often as possible. It seems reasonable to suppose that the chance of an input pulse being fed to a stage of high significance will be increased if the number of groups of variables at the end of high significance be increased. This can be achieved by keeping the number of variables in each group as low as possible. Thus, when ordering the constraints, the i^{th} constraint is chosen not only as a constraint having a low number of variables (relative to the remaining constraints) but also as a constraint having a high degree of overlap with the i-1 constraints already chosen. In other words the i^{th} constraint is chosen so as to minimize the number of variables in group i. This, then, takes into account factors (i) and (iii).

Another idea discussed in Section V.3 was that, when ordering the constraints, it would be advisable to select first constraints which are satisfied by a low proportion of truth-value assignments (which will be called strong constraints), leaving the weaker constraints until last; the reason for this is clear. When the variables in a group corresponding to a given constraint are being scanned the machine will only begin scanning the variables in groups of lower significance when the given constraint is satisfied. The scan will therefore be more efficient (shorter in length) when the given constraint is satisfied by only a low proportion of truth-value assignments.

In the following Section a method will be described of calculating the proportion of truth-value assignments satisfying any given constraint. Since this takes account of the logical form of the constraints factor (ii) is now being considered.

It would appear that the length of the machine's scan can be kept

short if, when ordering the constraints, preference is given to the stronger constraints which are satisfied by a low proportion of solutions and at the same time the number of variables in each group is kept as low as possible.

VII.2 Calculation of the Proportion of Truth-Value Assignments Satisfying a Constraint

Consider first the case of a single elementary proposition, P. The truth-values unity and zero may be assigned to this, the proposition being true when the truth-value unity is assigned. An elementary proposition, P, is therefore satisfied by half of the truth-values that may be assigned to it. The identical thing applies to the NEGATION of an elementary proposition.

Let $F \equiv F(U_1, U_2, ---, U_m)$ and $G \equiv G(V_1, V_2, ---, V_n)$ be two logical expressions where $U_1, U_2, ---, U_m, V_1, V_2, ---, V_n$ are all independent logical variables, i.e., F and G have no common variables. Out of the 2^m assignments of truth-values to $U_1, U_2, ---, U_m$ let a proportion f $(0 \leqslant f \leqslant 1)$ satisfy F. Similarly let a proportion g of the 2^n assignments of truth-values to $V_1, V_2, ---, V_n$ satisfy G.

Consider now the proportion of assignments of truth-values to the m+n variables satisfying the logical function F AND G. The number of assignments to $U_1, U_2, ---, U_m$ satisfying F is $2^m.f$. Each of these occurs with all the 2^n assignments of values to $V_1, V_2, ---, V_n$, of which $2^n.g$ satisfy G. Hence the number of assignments of values to $U_1, U_2, ---, U_m, V_1, V_2, ---, V_n$ satisfying F AND G is $2^m.f.2^n.g$. Thus, the proportion of truth-value assignments satisfying F AND G is given by the product f.g .

This is easily demonstrated in the case of the conjunction of two elementary propositions P,Q. Each of these is satisfied by half the values that may be assigned to it. The proportion of assignments to P,Q satisfying P AND Q is therefore one quarter. This is, of course, in agreement with the truth-table for the AND connective.

Consider next the proportion of truth-value assignments satisfying

the IDENTITY $F \equiv G$. The number of assignments to $U_1, U_2, \ldots, U_m$ satisfying F is $2^m.f$. Each of these occurs with the 2^n assignments to $V_1, V_2, \ldots, V_n$, of which $2^n.g$ satisfy G. This yields $2^m.f.2^n.g$ assignments of truth-values which satisfy the IDENTITY. Similarly, the number of assignments to $U_1, U_2, \ldots, U_m$ not satisfying F is $2^m.(1-f)$, and these combine with the $2^n.(1-g)$ assignments not satisfying G to yield a further $2^m.(1-f).2^n.(1-g)$ assignments of values satisfying the IDENTITY. Thus, the proportion of truth-value assignments to the variables $U_1, U_2, \ldots, U_m, V_1, V_2, \ldots, V_n$ which satisfy the IDENTITY $F \equiv G$ is given by $f.g+(1-f).(1-g)$.

In a similar way the proportion of truth-value assignments can be calculated which satisfy any of the logical functions of F,G and the following table summarizes the results.

Function	Proportion of Satisfying Truth-Value Assignments.
F	f
G	g
$\sim F$	1-f
F.G	fg
$F \vee G$	f + g - fg
$F \supset G$	1 - f + fg
$F \equiv G$	fg +(1-f)(1-g)
$F \not\equiv G$	f(1-g) + g(1-f)

F and G have no common variables.

FIGURE VII.1

As an example of the way in which the proportion of assignments satisfying a constraint is evaluated consider the following constraint.

$$[\sim(A.B.C) \vee (D \equiv E)] \supset \sim [F \supset (G \supset H.I)]$$

The sequence of steps involved is tabulated below:

Function	Proportion of Satisfying Truth-Value Assignments
A.B.C.	$\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$
$\sim(A.B.C)$	$1 - \frac{1}{8} = \frac{7}{8}$
$D \equiv E$	$\frac{1}{2} \cdot \frac{1}{2} + (1 - \frac{1}{2}).(1 - \frac{1}{2}) = \frac{1}{2}$
$\lceil$L.H. Side$\rfloor$	$\frac{7}{8} + \frac{1}{2} - \frac{7}{8} \cdot \frac{1}{2} = \frac{15}{16}$
H.I	$\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$
$G \supset H.I$	$1 - \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{4} = \frac{5}{8}$
$F \supset (G \supset H.I)$	$1 - \frac{1}{2} + \frac{1}{2} \cdot \frac{5}{8} = \frac{13}{16}$
$\lceil$R.H. Side$\rfloor$	$1 - \frac{13}{16} = \frac{3}{16}$
$\lceil$L.H.S.$\rfloor \supset \lceil$R.H.S.$\rfloor$	$1 - \frac{15}{16} + \frac{15}{16} \cdot \frac{3}{16} = \frac{61}{256}$

Thus, the above constraint is satisfied by a proportion $\frac{61}{256}$ of the truth-value assignments to the variables A,B,----,I. The evaulation may appear tedious but it can, in fact, be carried out very quickly, certainly in a much shorter time than would be required to construct the complete truth-table for the constraint.

Attention has so far been restricted to constraints in which each variable appears only once. A similar procedure may be worked out which applies quite generally to any constraints. However, if the logical expressions F and G have v variables in common it is first necessary to express any function of F and G as a disjunction of 2^v terms, each term containing a different combination of the v overlapping variables.

This indicates one way of determining the proportion of truth-value assignments satisfying a given constraint. An alternative method is simply to programme the machine to solve the given constraint.

VII.3. The Method of Investigation

Arbitrarily constructed problems were solved on the machine and those having fewer than 20 solutions were used for further investigation.

There are so many ways in which the constraints and variables of a 16 variable problem can be ordered that it is quite impossible to try all these ways when solving a problem on the machine. As a start, therefore, a very simple problem in five variables was put on the machine. In this case it was possible to solve the problem using every possible ordering and an interesting fact regarding the ordering was revealed (Section VII.4).

Other problems involving 12 and 16 variables were put on the machine. One method of ordering tried was to select first the constraint (or one of the constraints) involving the fewest variables and place these variables in group 1 at the most significant end of the binary counter of the machine. When selecting subsequent constraints the constraint was chosen which involved the least number of variables in addition to those contained in the constraints already chosen. In this way the groups of variables are kept small and a large number of groups is obtained. In general, ordering the constraints in this way will not produce a unique result. In this case several such orderings were tried.

At any intermediate stage when deciding the ordering of constraints a certain set of constraints will remain to be chosen. This set will consist of all those constraints involving at least one variable which is not contained in any of the groups of variables obtained so far in the ordering. When considering any given constraint in this set suppose that it is satisfied by a proportion f of its truth-value assignments and suppose, also, that the number of variables in the constraint which have not appeared in the ordering up to this stage is v. In Section VII.2 it was decided that a good ordering should be obtained if, at all stages in the ordering, the next constraint were chosen so as to minimize both v and f. In general it is not possible to minimize two independent quantities such as v and f simultaneously. However, it might produce

a good result if the product v.f is minimized. This method of ordering was also tried.

In order to have some basis on which to compare the results of these methods of ordering some arbitrary orderings were also tried. These were obtained by having a set of cards each bearing the number of one of the problem constraints and, after shuffling them, making a succession of arbitrary selections from the set of cards. Twenty such orderings were tried for each problem and the average length of scan resulting from these was used as a basis of comparison for the other methods of ordering.

It would take far too much space to present all the results obtained and for this reason an outline of the results of the five variable problem and one 16 variable problem are given together with the complete set of results for another 16 variable problem.

VII.4. Problem One

This is a five variable problem having two solutions and the following is a list of the constraints:

(1) $A \not\supset B$

(2) $A.E \equiv C$

(3) $B \equiv D$

(4) $(D \supset E) \supset A$

There are four-factorial ways of ordering the constraints and, due to the fact that some constraint orderings permit several different orderings of the variables, the total number of ways in which the constraints and variables can be ordered is 44. All these orderings were tried and the results are tabulated in Figure VII.2. The orderings are numbered from 1 to 44 so that they may be referred to conveniently. For each ordering the figures in the columns headed "S_1" , "S_2" and "E" give the number of transitions in the scan to reach the first solution, the second solution and the end of the scan respectively.

Order	S_1	S_2	E
1	2	6	8
2	2	6	8
3	1	5	8
4	1	5	8
5	2	5	7
6	1	4	7
7	2	5	7
8	1	4	7
9	2	6	8
10	1	5	8
11	6	11	13
12	6	11	13
13	4	11	13
14	2	11	13
15	3	11	13
16	2	11	13
17	7	13	16
18	7	13	16
19	4	13	16
20	3	13	16
21	5	13	16
22	3	13	16

Order	S_1	S_2	E
23	1	4	8
24	1	4	8
25	1	4	8
26	1	4	8
27	1	7	17
28	1	7	17
29	1	7	17
30	1	7	17
31	1	5	11
32	1	5	11
33	5	9	14
34	5	11	14
35	3	11	14
36	1	10	14
37	2	6	14
38	1	6	14
39	6	10	17
40	6	13	17
41	2	6	17
42	1	6	17
43	4	13	17
44	1	12	17

FIGURE VII.2.

The orderings have been subdivided so that all the orderings in any subdivision involve the same constraint ordering but the ordering of the variables within one or more of the groups is changed. As an example the four orderings in the first subdivision are given below, the brackets indicating the grouping of the variables A,B,C, etc., and the numbers referring to the constraints corresponding to the various groups.

Order 1. $\begin{bmatrix} A & B \\ 1 & \end{bmatrix}\begin{bmatrix} C & E \\ 2 & \end{bmatrix}\begin{bmatrix} D \\ 3,\ 4 \end{bmatrix}$

Order 2. $\begin{bmatrix} A & B \\ 1 & \end{bmatrix}\begin{bmatrix} E & C \\ 2 & \end{bmatrix}\begin{bmatrix} D \\ 3,\ 4 \end{bmatrix}$

Order 3. $\begin{bmatrix} B & A \\ 1 & \end{bmatrix}\begin{bmatrix} C & E \\ 2 & \end{bmatrix}\begin{bmatrix} D \\ 3,\ 4 \end{bmatrix}$

Order 4. $\begin{bmatrix} B & A \\ 1 & \end{bmatrix}\begin{bmatrix} E & C \\ 2 & \end{bmatrix}\begin{bmatrix} D \\ 3,\ 4 \end{bmatrix}$

The ordering of the variables in the first two groups varies but the constraint ordering is constant.

The significant point to notice in the tabulated results is that, for all the orderings in any subdivision, the same number of transitions is required to reach the end of the scan.

The $7^{th.}$ and $9^{th.}$ orderings are given below and it is seen that changing the order of variables D and E causes two groups of variables to merge into a single group. From Figure VII.2 it is seen that this produces a change in the number of transitions required to end the scan, and this is generally the case.

Order 7. $\begin{bmatrix} A & B \\ 1 & \end{bmatrix}\begin{bmatrix} D \\ 3 \end{bmatrix}\begin{bmatrix} E \\ 4 \end{bmatrix}\begin{bmatrix} C \\ 2 \end{bmatrix}$

Order 9. $\begin{bmatrix} A & B \\ 1 & \end{bmatrix}\begin{bmatrix} E & D \\ 4,3 & \end{bmatrix}\begin{bmatrix} C \\ 2 \end{bmatrix}$

These results suggest the following important rule:

The ordering of the variables within any group may be changed without affecting the length of the machine's scan provided that such change does not modify the grouping of the variables.

This postulate is easily substantiated. The scanning mechanism of the machine is such that once scanning of the variables within any given group has commenced it is bound, ultimately, to be exhaustively completed. This follows from the fact that the variables in the next group of higher significance will not be changed until a carry pulse is emitted from the stage of highest significance in the given group. Similarly, the complete scan is only terminated when a carry pulse occurs on the carry output line of the most significant stage of the counter. Since, whenever, the variables within any group are scanned, they are scanned exhaustively, the ordering of the variables within any group may be changed without affecting the length of the scan provided that the grouping of the variables is not affected.

VII.5. Problem Two

This is a 16 variable problem having 18 solutions. The constraints are listed below, the proportion, f, of truth-value assignments satisfying each constraint being indicated.

		f
(1)	$O \vee P$	$\frac{3}{4}$
(2)	$I \vee \left[(O \vee P).(B \supset I)\right]$	$\frac{11}{16}$
(3)	$\left\{\left[(O \vee P).(B \supset I)\right] \supset K\right\} \vee K$	$\frac{19}{32}$
(4)	$C \not\equiv \left\{\left[(O \vee P).(B \supset I) \supset K\right] \supset C\right\}$	$\frac{35}{64}$
(5)	$\left\{\left(\left[(L \vee M) \supset E\right] \supset F\right) \vee \left(\left[\left((J \vee N) \supset A\right) \supset D\right] \supset G\right)\right\} \supset H$	$\frac{9}{64}$
(6)	$\left[(E \not\equiv G) \equiv (L.J)\right] \not\equiv (C \equiv K)$	$\frac{1}{2}$
(7)	$\left[D \vee (L \not\equiv A)\right] \not\equiv \left[B.E \equiv N\right]$	$\frac{1}{2}$
(8)	$\left[(I.G) \vee P\right] \not\equiv (C \vee H)$	$\frac{7}{16}$
(9)	$(L.\bar{F}) \not\equiv \left[(G \vee \bar{H}) \supset (C.N)\right]$	$\frac{7}{32}$
(10)	$\left[(B.D) \not\equiv G\right] \supset (M \equiv L)$	$\frac{1}{4}$

One of the orderings of variables obtained by selecting the constraints so as to minimize the number of variables in successive groups is

$$\begin{bmatrix} 0 & P \\ 1 & \end{bmatrix}\begin{bmatrix} B & I \\ 2 & \end{bmatrix}\begin{bmatrix} K \\ 3 \end{bmatrix}\begin{bmatrix} C \\ 4 \end{bmatrix}\begin{bmatrix} G & H \\ 8 & \end{bmatrix}\begin{bmatrix} E & J & L \\ 6 & & \end{bmatrix}\begin{bmatrix} F & N \\ 9 & \end{bmatrix}\begin{bmatrix} A & D \\ 7 & \end{bmatrix}\begin{bmatrix} M & \\ 5, & 10 \end{bmatrix}$$

The numbers refer to constraints. There are 12 orderingsof the constraints which are equally good from the point of view of minimizing the number of variables in the groups. All these orderings were tried.

The next ordering tried was that obtained by taking into account not only the number of variables in each group but also the proportion of truth-value assignments satisfying the constraint corresponding to the group. Thus, successive constraints were chosen so as to minimize the product v.f as explained in Section VII.3.

In addition 20 arbitrary orderings were tried.

The results are given below. The number of truth-value assignments in an exhaustive scan for a 16 variable problem is 2^{16} which is approximately 65,000. For each method of ordering the factor of improvement is given relative to the exhaustive scan.

Constraints ordered:	Average number of transitions in complete scan.	Improvement factor relative to exhaustive scan.
to minimize the product v.f.	759	86
to minimize the number of variables in successive groups.	811	80
arbitrarily.	3,437	19

FIGURE VII.3.

Although the average of the orderings obtained by minimizing the number of variables in successive groups is not as good as, most of these orderings were actually better than the ordering obtained by minimizing the product v.f . For this problem there is no advantage to be obtained by taking into account the proportion, f, of truth-value assignments satisfying the individual constraints. It should be noted that, in this particular case, this proportion is of the same order of magnitude for all the constraints. The orderings obtained by rule give considerably better results than the average for the arbitrary orderings. However, one or two of the arbitrary orderings gave results as good as many of those obtained with calculated orderings.

VII.6. Problem Three

This 16 variable problem has no solutions. The constraints are listed and the proportion of truth-value assignments satisfying each one is indicated.

		f
(1)	$\left[B \not\equiv (C.H \equiv L)\right] \not\supset \left[(E \vee A) \supset (O.P)\right]$	$\frac{9}{32}$
(2)	$(J \vee \bar{F}) \supset \left[B \supset (K \equiv O)\right]$	$\frac{5}{8}$
(3)	$\left[D.(H \not\equiv M)\right] \equiv (A \not\equiv F \not\equiv O)$	$\frac{1}{2}$
(4)	$\left[(G \vee I) \supset E\right] \supset \left[D \equiv (B.P)\right]$	$\frac{11}{16}$
(5)	$\left[(N \vee D) \not\equiv (C \equiv J) \not\equiv (M \supset A)\right] \supset \bar{C}$	$\frac{3}{4}$
(6)	$(A \vee G \vee N) \supset (M.D \not\supset F)$	$\frac{57}{64}$
(7)	$(A \equiv E) \not\supset \left[(F.\bar{I}) \supset (O \vee I)\right]$	$\frac{1}{32}$

Ordering the constraints so as to minimize the number of variables in successive groups produces 22 possibilities. All these orderings are given. The same bracket notation is used to indicate the grouping and, again, the numbers refer to the constraints brought in by successive groups.

The number of transitions in the complete scan is indicated for each ordering.

Number	Ordering	Transitions
(1)	[B F J K O / 2] [A D H M / 3] [C N / 5] [G / 6] [E L P / 1] [I / 4,7]	7,935
(2)	[B F J K O / 2] [A D H M / 3] [C N / 5] [G / 6] [E I P / 4] [L / 1,7]	9,429
(3)	[B F J K O / 2] [A D H M / 3] [C N / 5] [G / 6] [E I L / 7] [P / 1,4]	6,957
(4)	[B F J K O / 2] [A D H M / 3] [G N / 6] [C / 5] [E L P / 1] [I / 4,7]	7,055
(5)	[B F J K O / 2] [A D H M / 3] [G N / 6] [C / 5] [E I P / 4] [L / 1,7]	8,549
(6)	[B F J K O / 2] [A D H M / 3] [G N / 6] [C / 5] [E I L / 7] [P / 1,4]	6,093
(7)	[B F J K O / 2] [A E I L / 7] [C H P / 1] [D M / 3] [G / 4] [N / 5,6]	849
(8)	[B F J K O / 2] [A E I L / 7] [C H P / 1] [D M / 3] [N / 5,6] [G / 4]	729
(9)	[B F J K O / 2] [A E I L / 7] [C H P / 1] [D G / 4] [M / 3] [N / 5,6]	789
(10)	[B F J K O / 2] [A E I L / 7] [D H M / 3] [C P / 1] [G / 4] [N / 5,6]	1,095
(11)	[B F J K O / 2] [A E I L / 7] [D H M / 3] [C P / 1] [N / 5,6] [G / 4]	975
(12)	[B F J K O / 2] [A E I L / 7] [D H M / 3] [G P / 4] [C / 1] [N / 5,6]	1,255

Number	Ordering	Transitions
(13)	[B F J K O / 2] [A E I L / 7] [D H M / 3] [G P / 4] [N / 6] [C / 1,5]	1,191
(14)	[B F J K O / 2] [A E I L / 7] [D H M / 3] [C N / 5] [P / 1] [G / 4,6]	1,383
(15)	[B F J K O / 2] [A E I L / 7] [D H M / 3] [C N / 5] [G / 6] [P / 1,4]	1,279
(16)	[B F J K O / 2] [A E I L / 7] [D H M / 3] [G N / 6] [P / 1] [C / 4,5]	911
(17)	[B F J K O / 2] [A E I L / 7] [D H M / 3] [G N / 6] [C / 4,5] [P / 1]	938
(18)	[B F J K O / 2] [A E I L / 7] [D G P / 4] [C H / 1] [M / 3] [N / 5,6]	909
(19)	[B F J K O / 2] [A E I L / 7] [D G P / 4] [H M / 3] [C / 1] [N / 5,6]	1,129
(20)	[B F J K O / 2] [A E I L / 7] [D G P / 4] [H M / 3] [N / 6] [C / 1,5]	1,065
(21)	[B F J K O / 2] [A E I L / 7] [D G P / 4] [M N / 6] [H / 3] [C / 1,5]	813
(22)	[B F J K O / 2] [A E I L / 7] [D G P / 4] [M N / 6] [C / 5] [H / 1,3]	799

The next ordering tried was that obtained by taking account of the proportion of truth-value assignments satisfying the individual constraints and selecting successive constraints to minimize the product v.f . For this problem this results in the following unique ordering:

(23)	[A E F I L O / 7] [B C H P / 1] [D M / 3] [G / 4] [N / 6] [J / 5] [K / 2]	171

Finally the following 20 arbitrary orderings were tried:

Number	Ordering	Transitions
(24)	[A C D J M N / 5] [I O F E L / 7] [G / 6] [P B / 4] [K / 2] [H / 1,3]	1,701
(25)	[B F J K O / 2] [M D H A / 3] [E P I G / 4] [C N / 5,6] [L / 1,7]	11,683
(26)	[B D E G I P / 4] [L F O A / 7] [M N / 6] [K J / 2] [H / 3] [C / 1,5]	841
(27)	[B D E G I P / 4] [F O A H M / 3] [J K / 2] [L / 7] [C / 1] [N / 5,6]	6,075
(28)	[A C D J M N / 5] [F O L I E / 7] [B P H / 1,3] [G / 4,6] [K / 2]	1,983
(29)	[A E F I L O / 7] [H M D / 3] [B K J / 2] [C P / 1] [N G / 4,5,6]	571
(30)	[A B C E H L O P / 1] [D I G / 4] [M F / 3,7] [J N / 5,6] [K / 2]	1,929
(31)	[A B C E H L O P / 1] [K J F / 2] [M G N D / 3,5,6] [I / 4,7]	8,307
(32)	[A D F H M O / 3] [L E I / 7] [G N / 6] [C J / 5] [P B / 1,4] [K / 2]	335
(33)	[A B C E H L O P / 1] [M D N G F / 3,6] [I / 4,7] [J K / 2,5]	2,715
(34)	[B D E G I P / 4] [F A M H O / 3] [L C / 1,7] [J N / 5,6] [K / 2]	3,629
(35)	[A D F H M O / 3] [J C N / 5] [K B / 2] [E P I G / 4,6] [L / 1,7]	11,386
(36)	[A D F G M N / 6] [C J / 5] [K O B / 2] [P E I / 4] [H L / 1,3,7]	6,231

Number	Ordering	Transitions
(37)	[A C D J M N / 5] [G F / 6] [B O K / 2] [I E P / 4] [L / 7] [H / 1,3]	3,816
(38)	[A E F I L O / 7] [D C N M J / 5] [K B / 2] [G P / 4,6] [H / 1,3]	809
(39)	[A C D J M N / 5] [F K B O / 2] [G I E P / 4,6] [H / 3] [L / 1,7]	12,640
(40)	[A E F I L O / 7] [P H C B / 1] [J K / 2] [D N M G / 3,4,5,6]	567
(41)	[A B C E H L O P / 1] [J N D M / 5] [I F / 3,7] [K / 2] [G / 4,6]	4,135
(42)	[A D F H M O / 3] [G N / 6] [C P B L / 1] [E I / 4,7] [K J / 2,5]	753
(43)	[A C D J M N / 5] [E O L I F / 7] [B K / 2] [H / 3] [G / 6] [P / 1,4]	2,048

These results are summarized in the Table in Figure VII.4 where the number of transitions in the complete scan is averaged for each method of ordering and the factor of improvement relative to an exhaustive scan is given. Ordering the constraints so as to minimize the number of variables in successive groups only reduces the length of scan by a factor of about three relative to arbitrary ordering. However, as will be explained, this is not a fair comparison.

Referring to the list of constraints on p.127 it is seen that the proportion of truth-value assignments satisfying constraint 7 is an order of magnitude less than the corresponding proportion for any other constraint. If the results of the first 22 orderings (minimizing the number of variables in successive groups) are inspected it is seen that they may be divided into two sets; the number of transitions in the scan for the first six orderings is in the range 6,000 to 10,000 while the corresponding figure for orderings

Constraints ordered:	Average number of transitions in complete scan.	Improvement factor relative to exhaustive scan.
to minimize the product v.f.	171	380
to minimize the number of variables in successive groups.	2,815	23
arbitrarily.	8,215	7.8

FIGURE VII.4.

7 to 22 is in the region of 1,000. The reason for this is that the strong constraint, constraint 7, was one of the last to be selected in orderings 1 to 6 but was the second constraint to be selected in orderings 7 to 22.

Ordering 23 was obtained by making use of the information regarding the proportion of truth-value assignments satisfying the individual constraints and, as a result, constraint 7 was given top priority by placing it first in the ordering. The second constraint chosen in this ordering was number one, which is seen to be the second strongest constraint. Consequently, the complete scan with this ordering contained only 171 transitions, which is nearly 400 times better than an exhaustive scan.

If all the results for this problem are studied there is found to be a marked correlation between the nearness of constraint 7 to the left-hand end of the ordering and the number of transitions in the abbreviated scan. To a lesser extent the same correlation is evident with respect to the position in the ordering of constraint 1.

For this particular problem then, it is definitely worth while considering the proportion of truth-value assignments satisfying the individual constraints.

VII.7. Summary

It appears unlikely that any rules can be found for obtaining an optimal ordering of the constraints and variables of a problem. This is because the formulation of such rules would require a knowledge of the way in which the subsets of points in the problem space corresponding to the individual constraints overlapped. This knowledge is not readily available. Nevertheless certain broad rules can be given which will, in general, result in an ordering of the constraints that will be considerably better than an arbitrary ordering. The following strategy should be quite effective:

(i) Determine the proportion of truth-value assignments satisfying each individual constraint. This can be done in the way described in Section VII.2 although this method gets out of hand when many variables occur in a constraint more than once. A better way is probably to solve the individual constraints using the machine.

(ii) If the proportions for all the constraints are of the same order it is not worth taking them into account. In this case the constraints should be ordered so as to minimize the number of variables in successive groups.

(iii) If the proportion of truth-value assignments satisfying one or more of the constraints is considerably lower than that for the remaining constraints then it is worth considering the strength of the constraints when deciding the ordering. This can be done by grouping the variables so that the product of the proportion of truth-value assignments satisfied by the constraint(s) corresponding to a group and the number of variables in that group is minimized. The ordering should be continued in this way until all the particularly strong constraints have been included. After that the ordering may be continued as in (ii).

(iv) Whatever the method of ordering the constraints, the ordering of the variables <u>within any group</u> is unimportant if the grouping of the variables is independent of such ordering.

An ideal ordering would be one in which each "group" of variables contained only a single variable with the stronger constraints occurring early in the ordering. Problems best suited to solutions on the abbreviated scan machine are those which permit an ordering of the constraints and variables in which the number of variables per group is low, particularly at the beginning of the ordering. A well suited problem should therefore have a few constraints in which only a few variables occur so that the first few groups of variables in the ordering can be made small. The remaining constraints should overlap (with respect to the variables they contain) in such a way that the number of variables in successive groups can be kept low. The stronger the constraints of a problem the more it is suited to solution by this machine.

VIII. APPLICATION OF THE ABBREVIATED SCAN TECHNIQUE TO THE ACADEMIC TIMETABLE PROBLEM

VIII.1. The Problem

Consider the problem of constructing a timetable for four classes (groups of people) for a week divided into 20 periods. There are five lecturers and four rooms available. The number of periods in the week during which each class must be taken by each lecturer is defined by an entry in the following matrix.

Classes \ Lecturers	1	2	3	4	5
1	8	9	3	0	0
2	2	0	12	6	0
3	4	5	0	7	4
4	6	3	5	0	6

Suppose that the following constraints must be satisfied between any class/lecturer combination and the allocated room:

(i) Lecturer 1 may take class 1 only in rooms 1 or 4.
(ii) Lecturer 2 may not take any class in room 3.
(iii) Lecturer 3 may not take any class in room 3.
(iv) Lecturer 5 may take any class only in room 3.

Constraints of this nature are necessary to ensure that, for example, a class is not taken for a period of mathematics in a machine-shop.

Suppose, also, that no class may be taken by the same lecturer for more than two consecutive periods.

In addition the following must be satisfied:

(v) A lecturer cannot take two classes simultaneously, and
(vi) Two or more classes may never occupy the same room.

For each of the 20 periods in the week, to each class must be allocated a lecturer and a room. The complete timetable for the four classes contains 80 class-periods. The total number of possible ways of filling in these 80 class-periods is $(4x5)^{80}$ - an astronomical number! Of course, most of these timetables correspond to impossible situations such as one lecturer being called upon to take three classes at once, or all the classes being in the same room.

VIII.2. Expression of the Problem in Terms of State Variables

Let the state variable $L_{\alpha i}$ represent the lecturer taking class α during period i (the i^{th} period of the week). Similarly, let $R_{\alpha i}$ represent the room allocated to class α for period i.

α has values 1, 2, 3, 4

i has values 1, 2, 3,-------,20

Hence there are 80 state variables $L_{\alpha i}$ and 80 state variables $R_{\alpha i}$ corresponding to the 80 class-periods in the timetable.

Since there are five lecturers, each state variable $L_{\alpha i}$ can have assigned to it one of the values 1,2,3,4 or 5, each corresponding to a particular lecturer. Similarly, one of the values 1,2,3 or 4 is assigned to each state variable $R_{\alpha i}$ corresponding to a particular room.

A solution to the timetable problem is now an assignment of values to the 160 state variables which satisfies all the constraints enumerated in the previous Section. There will be a set of these constraints corresponding to each of the 20 periods in the week. This leads to an obvious grouping of the state variables, as below, with each group containing the state variables corresponding to one period.

Group 1. Group i.

$$\left[L_{1,1}L_{2,1}L_{3,1}L_{4,1},\ R_{1,1}R_{2,1}R_{3,1}R_{4,1}\right] \text{ ----} \left[L_{1,i}L_{2,i}L_{3,i}L_{4,i},\ R_{1,i}R_{2,i}R_{3,i}R_{4,i}\right]-$$

Group 20.

$$\text{--------}\left[L_{1,20}L_{2,20}L_{3,20}L_{4,20},\ R_{1,20}R_{2,20}R_{3,20}R_{4,20}\right]$$

The assignment of values to the 160 state variables forms a mixed radix number as discussed in Section V.4. The state variables in group 1 correspond to the most significant digits of this number while those in group 20 correspond to the least significant digits. The set of constraints corresponding to the i^{th} period of the week involve the state variables in groups 1 to i.

VIII.3. The Method of Scanning the Variables within a Group.

The problem is solved by scanning the variables in group 1 to find a satisfactory arrangement for period 1. When this has been found the variables in group 2 are scanned to determine the timetable entries for period 2. The procedure continues in this way until it proves impossible to find a satisfactory arrangement for one of the periods. The variables corresponding to the previous period are then further scanned to find an alternative arrangement for that period.

Consider the variables in group i -

$$\left[L_{1,i}L_{2,i}L_{3,i}L_{4,i},\ R_{1,i}R_{2,i}R_{3,i}R_{4,i}\right]$$

The value assigned to $L_{1,i}$ indicates the lecturer taking class 1 during period i while the value assigned to $R_{1,i}$ indicates the room allocated to class 1 for period i. Thus, the assignment of values [2 1 3 5, 4 3 2 1] to the variables in group i would mean that, during period i,

class 1 is taken by lecturer 2 in room 4
class 2 is taken by lecturer 1 in room 3
class 3 is taken by lecturer 3 in room 2
class 4 is taken by lecturer 5 in room 1.

An entry, $P_{\alpha\beta 1}$, in the matrix below (copied from Section VIII.1) gives the number of periods in the week during which class α must be taken by lecturer β.

Lecturers

$P_{\alpha\beta}$ 1	1	2	3	4	5	β
1	(8)	9	3	0	0	
2	2	0	(12)	6	0	
Classes 3	4	(5)	0	7	4	
4	6	3	5	0	(6)	
α						

Period 1 matrix

$L_{1,1}$	$L_{2,1}$	$L_{3,1}$	$L_{4,1}$	(Period 1)
1	1	1	1	
1	1	1	2	
1	1	1	3	
1	1	1	4	
1	1	1	5	
1	1	2	1	
1	1	2	2	Exhaustive scan.
1	1	2	3	
1	1	2	4	
1	1	2	5	
1	1	3	1	

The solution procedure starts by scanning the variables $L_{1,1}$ $L_{2,1}$ $L_{3,1}$ and $L_{4,1}$ of group 1. The first few assignments of an exhaustive scan are given above. However, most of the possible assignments are unacceptable since one lecturer is required to give more than one lecture during period 1. A much more efficient scan is obtained in the following way. To $L_{1,1}$ assign the lowest value corresponding to a non-zero entry in row 1 of the period 1 matrix. In this case the assigned value is unity, representing lecturer 1 (this is indicated in the matrix by encircling the entry corresponding to class 1 and lecturer 1). Then, considering class 2 (row 2), assign to $L_{2,1}$ the lowest value corresponding to a non-zero entry in a column not yet used. Column 1 has been used and column 2 (row 2) contains a zero entry. Hence assign to $L_{2,1}$ the value three, corresponding to column 3. The lecturers taking classes 3 and 4 during period 1 are found in a similar manner and the first assignment of values to $L_{1,1}L_{2,1}L_{3,1}L_{4,1}$ is 1 3 2 5 .

The next assignment in this scan would be obtained by increasing the fourth digit (5) to the next lowest possible value. In this case it cannot be increased. Therefore the third digit (2) is increased to the next lowest possible value, which is four, and the fourth digit is then reduced to the value two. The first few assignments in this scan are:

$L_{1,1}$	$L_{2,1}$	$L_{3,1}$	$L_{4,1}$	(Period 1)
1	3	2	5	
1	3	4	2	
1	3	4	5	
1	3	5	2	
1	4	2	3	
1	4	2	5	
1	4	5	2	
1	4	5	3	
2	1	4	3	

This scan progresses far more rapidly through the range of values than the exhaustive scan. Note that the constraints to ensure that a lecturer is not called upon to deliver more than one lecture in any period are implicitly contained in this scanning system. It is therefore unnecessary to apply these constraints to the assignments generated by the scan. Similarly, the constraints to prevent one room being allocated to more than one class for the same period will be implicitly contained in the system used to scan the assignments of values to the state variables representing the rooms.

VIII.4 The Complete Scanning Procedure

One constraint imposed is that a class must not have the same lecturer for more than two consecutive periods. This is allowed for when scanning the variables of group i (period i) by observing the assignments to the variables in groups i-1 and i-2. The constraint may be written in the form:

$$\text{If } \left[L_{\alpha(i-2)} = L_{\alpha(i-1)}\right], \text{ then } \left[L_{\alpha(i-1)} \neq L_{\alpha i}\right] \qquad (1)$$

$$\text{for } \alpha = 1,2,3,4 \; ; \quad i = 3,4,5,\text{---},20.$$

The equality sign is used here to indicate equality of the values assigned to the state variables.

The complete scanning procedure will now be outlined.

The first assignment of values to the L-variables of group 1 is obtained by considering the period 1 matrix as previously explained. In a similar way the R-variables are scanned starting with the assignment of values 1 2 3 4. The first assignment of values to group 1 is thus

$$\left[1\ 3\ 2\ 5,\ 1\ 2\ 3\ 4\right] \quad \text{Period 1.}$$

Referring to the constraints between lecturer and rooms given on p. 135 it is found that lecturer 2 cannot teach in room 3. The above assignment is therefore unsatisfactory and the next assignment to the R-variables must be tried, making the next assignment to group 1.

$$\left[1\ 3\ 2\ 5,\ 1\ 2\ 4\ 3\right] \quad \text{Period 1.}$$

This is a suitable arrangement for period 1.

The matrix for period 1 is now modified by subtracting unity from the relevant entries (encircled) to give a new matrix. Each entry, $P_{\alpha\beta 2}$, in this new matrix gives the number of periods after period 1 during which class α must be taken by lecturer β.

		Lecturers					
	$P_{\alpha\beta 2}$	1	2	3	4	5	β
	1	7	9	3	0	0	
	2	2	0	11	6	0	
Classes	3	4	4	0	7	4	
	4	6	3	5	0	5	
	α						

Period 2 matrix

Two further rules can now be explained which make the scanning more efficient. The first is simply that the number of outstanding lectures that a given lecturer must deliver cannot exceed the number of outstanding periods in the week. The sum of the entries in column β of the period i

matrix gives the number of lectures that lecturer β must deliver after the $(i-1)^{th}$ period. Thus,

$$\sum_{\alpha=1}^{4} p_{\alpha\beta i} \leqslant 21 - i \qquad (2)$$

$$\text{for } \beta = 1,2,3,4,5$$
$$i = 1,2,3,\text{-----},20.$$

and, if

$$\sum_{\alpha=1}^{4} p_{\alpha\beta i} = 21 - i \ ,$$

then lecturer β must lecture in period i.

The second rule is a further implication of the fact that a class must not have the same lecturer for more than two consecutive periods. The value of $p_{\alpha\beta i}$ gives the number of periods after the $(i-1)^{th.}$ during which class α must be taken by lecturer β. It follows that any entry in the period i matrix must not exceed twice the sum of all other entries in the same row (i.e. corresponding to the same class). This is expressed thus:

$$p_{\alpha\beta i} \geqslant 2 \sum_{\mu \neq \beta} p_{\alpha\mu i} \qquad (3)$$

$$\text{for } \alpha = 1,2,3,4$$
$$i = 1,2,3,\text{-----},20.$$

and, if

$$p_{\alpha\beta i} > 2 \sum_{\mu \neq \beta} p_{\alpha\mu i} \ ,$$

then lecturer β must take class α during period i.

Applying rule (2) to the period 2 matrix it is seen that lecturers 1 and 3 must lecture in period 2. At this stage rule (3) is satisfied. By considering the period 2 matrix it is found that the first assignment

of values to be tried for the variables of group 2 is the same as that for group 1.

i.e. $\left[1\ 3\ 2\ 5,\ 1\ 2\ 3\ 4\right]$ Period 2.

Once again the room allocation is not suitable and the next assignment is therefore

$$\left[1\ 3\ 2\ 5,\ 1\ 2\ 4\ 3\right] \quad \text{Period 2.}$$

which is satisfactory.

The period 3 matrix is thus

		Lecturers.					
	$p_{\alpha\beta 3}$	1	2	3	4	5	β
	1	6	9	3	0	0	
	2	2	0	10	6	0	
Classes.	3	4	3	0	7	4	
	4	6	3	5	0	4	
	α						

Period 3 matrix

Rule (1) is now applied and, since each class has had the same lecturer for two periods running, it must have a change in period 3.

Rule (2) indicates that lecturers 1 and 3 must lecture in period 3.

Rule (3) is still satisfied.

Since each class must have a change of lecturer, when considering the period 3 matrix the first value to be assigned to the most significant of the L-variables (i.e. $L_{1,3}$) is two. The first assignment to the group 3 variables is then

$$\left[2\ 1\ 4\ 3,\ 1\ 2\ 3\ 4\right] \quad \text{Period 3.}$$

which gives each class a change of lecturer. The allocation of rooms is also found to satisfy the constraints (given on p. 135).

The period 3 matrix is then modified to give the matrix for period 4 and the scanning is continued in the above way, applying rules (1), (2) and (3) at each stage and observing the constraints between lecturer and allocated room.

The complete scan of the variables to the stage where the first possible timetable is found is given in Appendix 2. The total number of "moves" in the scan is only 26.

Any other constraints that might exist in a problem of this nature could be allowed for either by embodying them in the system of scanning the variables or by applying them to the assignments of values generated by the scan. Although the problem considered above is very trivial it serves to indicate the type of problem that can be dealt with by this method. As the variables may be many-valued (as distinct from two-valued), and as the problem constraints are not restricted to being of a logical nature (algebraic constraints being permitted), this method of scanning should be applicable to a wide variety of problems. Problems in production scheduling or any other problems which can be expressed in terms of state variables (as defined in Section V.4) should be amenable to this method of attack.

This method would normally be suitable only where it is sufficient that one or a few possible solutions to a problem be found. If all possible solutions are required then, even with this abbreviated scan technique the time taken would probably be far too great.

The best way of solving problems by this abbreviated scan method would probably be to programme a general purpose digital computer.

IX. CONCLUSIONS

IX.1. Measure Functions for Many-Valued Logics

The concept has been discussed of a problem space consisting of a set of discrete points, each point uniquely representing one assignment of truth-values to the logical variables in a problem. One possible problem space for an n variable problem in m-valued logic is an n-dimensional hypercube in which all the sides contain m equally spaced points. One dimension in the space corresponds to one variable and the m levels in that dimension represent the different truth-values that may be assigned to that variable. A single transition in this space corresponds to a change in the truth-value of one variable of unity.

In Section III it was shown that any n variable problem in m-valued logic can be expressed as an equivalent problem in two-valued logic involving n.N two-valued variables, where N is the least integer such that $2^N \geq m$. When many-valued problems are expressed in terms of two-valued logic in this way the most appropriate problem space is an n.N-dimensional hypercube. A single transition in this space corresponds to a change in the truth-value of one of the two-valued variables in terms of which the problem is being expressed.

The class of problems considered consists of problems in which a set of logical constraints involving many-valued variables and connectives must be satisfied simultaneously. A constraint is satisfied when its truth-value assumes the highest value possible in the given system of logic. The truth-values of the individual constraints are joined by a many-valued AND connective to give the final truth-value of the problem. A solution to the problem is any assignment of truth-values to the logical variables under which the final truth-value assumes the maximum possible value.

It has been shown that, when the first type of problem space is being used (in which each dimension of an n-dimensional hypercube is subdivided to give m equally spaced points), the final truth-value of a problem can

be used as a measure function when searching through the problem space for solutions.

When a many-valued problem is being expressed in terms of two-valued logic and the problem space consists of an n.N-dimensional hypercube it has been shown that the parity of the final truth-value may be used as a measure function. The parity is the number of 1's in the binary expression of the final truth-value.

In Section IV.6 ideas were expressed on the way in which a machine could be made to search through the problem space for solutions to a problem, being guided by the values of a measure function evaluated at each point covered by the search. Starting at some point in the space the machine could evaluate the measure function at this point and at all points within, say, four single transitions of the given point. The average value of the measure function in the neighbourhood of the given starting point could be calculated and this should give an indication of the likelihood of finding a solution to the problem in that region of the space. Also, if the mean value of the measure function increased when the search point moved in a particular direction in the space, then the machine could be made to concentrate its search in that direction. In general, the machine could be made to intensify its search in areas where the mean value of the measure function exceeded some stipulated level and to spend little time in areas where the mean value of the measure function fell below that level. Since this sort of procedure would probably tend to find a solution near the starting point, should one exist, the machine could be made to perform a series of searches from starting points either chosen at random or uniformly spaced throughout the problem space.

In this way the machine should grope its way through a problem space and home on solutions. For problems involving large numbers of variables where it is necessary to find only one or a few solutions a solution procedure of the type outlined here should prove to be very effective.

IX.2. The Abbreviated Scan Machine

With one exception, all the machines previously constructed for solving sets of simultaneous constraints in two-valued logic have operated on the principle of making an exhaustive scan of the truth-value assignments to all the logical variables appearing in a problem. Any machine of this kind is severely limited in the number of logical variables that it can handle since, as this number increases, the number of steps in the exhaustive scan increases prohibitively. Even allowing for the most optimistic improvements in computing techniques and for consequent increases in operating speeds it is extremely unlikely that problems involving more than 40 to 50 variables could ever be solved in a practicable length of time using the exhaustive scan method.

The exception mentioned above is a "feedback logical computer" built by McCallum and Smith in 1951. This machine searches through the problem space of a problem according to a stochastic process. Since its scan is non-deterministic it is impossible to tell, when solving a problem on the machine, whether the machine has found all the solutions. If the machine has not succeeded in finding a solution it is impossible to determine whether or not the problem in question has any solutions.

Section V describes a method that has been developed for finding all the solutions to a problem in two-valued logic without making an exhaustive scan of the truth-value assignments. This method of solution has been fully mechanized and a relay operated machine has been built which is capable of solving problems involving up to 16 logical variables and not more than 100 logical connectives. Due to its method of operation this machine has been called an "abbreviated scan" computer.

Methods of programming this machine have been investigated and it has been shown that the number of steps taken by the machine in the scan to completely solve a problem depends upon the order in which the problem constraints and variables are presented to the machine.

It does not seem possible to devise any optimal programming techniques enabling a scan of minimal length to be obtained for a given problem. However, rules have been formulated which should, in general, result in an ordering of the constraints and variables of a problem yielding a scan containing far fewer steps than that obtained using any arbitrary ordering.

Although the effectiveness of the abbreviated scan machine depends upon the particular problem being solved, the results show that for problems in 16 variables the machine usually performs a scan in which the number of steps is about two orders of magnitude lower than that in the corresponding exhaustive scan. It is expected that this factor of improvement, relative to an exhaustive scan, should increase greatly with the number of variables appearing in the problems.

In Section V.4 and Section VIII it has been shown that the principle of the abbreviated scan machine can easily be generalized to deal with problems in many-valued logic or with any problems that can be expressed in terms of state variables (as defined in Section V.4). Such problems may involve any variables capable of assuming any one of a finite number of different states, and may therefore involve algebraic variables and constraints and/or logical variables and constraints.

IX.3. On the Application of Logic

As yet, symbolic logic is not widely used in application to engineering or other problems. However, it seems inevitable that, in the near future, many applications will be found in the ever increasing classes of organizational problems arising in industry and elsewhere. Indeed, many people realize this and are working on these ideas. Some possible applications and a few references have been quoted in the Introduction.

APPENDIX 1 .

Solution of a Set of Logical Constraints by Systematic Reduction

The four rules of the manufacturer problem given in the Introduction are expressed in the following conjunction:

$$\left[(A.B)\vee C\right].\ \overline{(B.C)}.\ (D\supset B).\ (D\vee E)$$

A list of tautologies was given on p. 25 in Section I. These will be repeatedly used to simplify the logical expression of the problem until, finally, the solutions are obtained. The numbers of the tautologies mentioned correspond to the numbering in the above mentioned list.

In the first place tautology (11) is used to re-express the NEGATED rule and tautology (13) is used to express the IMPLICATION in terms of other connectives. The logical expression becomes

$$\left[(A.B)\vee C\right].\ (\overline{B}\vee\overline{C}).\ (\overline{D}\vee B).\ (D\vee E)$$

Tautology (9) is used to expand the first two rules and similarly to expand the second two rules. The resulting expression is

$$(A.B.\overline{B}\vee A.B.\overline{C}\vee\overline{B}.C\vee C.\overline{C}).(D.\overline{D}\vee\overline{D}.E\vee B.D\vee B.E)$$

Tautology (2) implies that any logical conjunction containing both a variable and the NEGATION of that variable (e.g. the term $A.B.\overline{B}$ above), is always false. All such terms are therefore omitted to give the following expression -

$$(A.B.\overline{C}\vee\overline{B}.C).(\overline{D}.E\vee B.D\vee B.E)$$

A further application of tautology (9) gives the following -

$$A.B.\overline{C}.\overline{D}.E\vee A.B.\overline{C}.D\vee A.B.\overline{C}.E\vee\overline{B}.C.\overline{D}.E\vee B.\overline{B}.C.D\vee B.\overline{B}.C.E$$

Tautology (2) is applied once more -

$$A.B.\bar{C}.\bar{D}.E \vee A.B.\bar{C}.D \vee A.B.\bar{C}.E \vee \bar{B}.C.\bar{D}.E$$

A disjunction of this form will yield the solutions to the problem if each term contains all the variables in the problem. Tautology (3) indicates that a term such as $(E \vee \bar{E})$ is always true. The second term in the above expression can therefore be made to involve the variable, E, by joining it, with an AND connective, to the term $(E \vee \bar{E})$. Applying tautology (3) in this way the expression becomes -

$$A.B.\bar{C}.\bar{D}.E \vee A.B.\bar{C}.D.(E \vee \bar{E}) \vee A.B.\bar{C}.E.(D \vee \bar{D}) \vee \bar{B}.C.\bar{D}.E.\ (A \vee \bar{A})$$

Tautology (9) is then used to expand the terms in this expression to obtain:

$$A.B.\bar{C}.\bar{D}.E \vee A.B.\bar{C}.D.E \vee A.B.\bar{C}.D.\bar{E} \vee A.B.\bar{C}.D.E \vee A.B.\bar{C}.\bar{D}.E \vee A.\bar{B}.C.\bar{D}.E \vee \bar{A}.\bar{B}.C.\bar{D}.E$$

Applying tautology (2) as before the list of five solutions to the manufacturer problem is obtained -

$$A.B.\bar{C}.\bar{D}.E \vee A.B.\bar{C}.D.E \vee A.B.\bar{C}.D.\bar{E} \vee A.\bar{B}.C.\bar{D}.E \vee \bar{A}.\bar{B}.C.\bar{D}.E$$

This method of solution is of no use whatever for problems containing many variables due to the extremely large numbers of terms that would be obtained when expanding the logical expressions.

APPENDIX 2

The Abbreviated Scan for the Timetable Problem in Section VIII.

In summary, the statement of the problem is:

Classes : 1, 2, 3 and 4 .
Lecturers : 1, 2, 3, 4 and 5 .
Rooms : 1, 2, 3 and 4 .
Periods in week : 1, 2, 3,-----, 20.

The number of periods in the week for which each class must be taken by each lecturer is defined by an entry in a matrix referred to as the "period 1 matrix".

No class must receive more than two consecutive periods with the same lecturer.

A lecturer cannot take more than one class in the same period.

Not more than one class may occupy the same room during any period.

A group of state variables is defined corresponding to each period. The group for period i is

$$\left[L_{1,i}L_{2,i}L_{3,i}L_{4,i} \ , \ R_{1,i}R_{2,i}R_{3,i}R_{4,i}\right] \ ,$$

where the value assigned to $L_{\alpha,i}$ denotes the lecturer taking class α during period i and the value assigned to $R_{\alpha,i}$ denotes the room allocated to class α for period i , for $\alpha = 1,2,3,4$.

The solution procedure commences by scanning the variables in group 1. corresponding to period 1. When a suitable assignment is found the period 1 matrix is modified by subtracting unity from the relevant entries to give the period 2 matrix. The variables in group 2 are then scanned to find a suitable arrangement for period 2.

When determining the assignment of values to the variables in group i, corresponding to period i, the following points must be observed:

(i) to prevent any class receiving more than two consecutive periods with the same lecturer,

$$\text{if } \left[L_{\alpha,(i-2)} = L_{\alpha,(i-1)}\right], \text{ then } \left[L_{\alpha,(i-1)} \neq L_{\alpha,i}\right] \quad (1)$$

for $\alpha = 1,2,3,4.$

Where a class has had the same lecturer for periods i-2 and i-1 the entry in the period i matrix corresponding to that lecturer is <u>crossed out</u>.

(ii) if the number of outstanding lectures for any lecturer is equal to the number of outstanding periods then that lecturer <u>must</u> lecture in period i ,

$$\left.\begin{array}{l} \text{i.e.} \quad \text{if } \sum_{\alpha=1}^{4} p_{\alpha\beta i} = 21 - i, \\ \text{then lecturer } \beta \text{ must lecture in period i.} \end{array}\right\} \quad (2)$$

When this occurs it is indicated by placing an <u>arrow</u> below any column of the period i matrix corresponding to a lecturer who must lecture in that period.

(iii) to ensure that the situation does not arise where a class has so many outstanding periods with one lecturer that it would necessitate that lecturer taking the class for more than two consecutive periods at some stage,

$$\left.\begin{array}{l} \text{if } p_{\alpha\beta i} > 2 \sum_{\mu \neq \beta} p_{\alpha\mu i}, \quad \text{for } \alpha = 1,2,3,4, \\ \text{then lecturer } \beta \text{ <u>must</u> take class } \alpha \text{ in period i.} \end{array}\right\} \quad (3)$$

This is indicated in the period i matrix by <u>encircling</u> the entry corresponding to any class/lecturer combination that must occur in period i .

When a suitable assingment of values has been found for the L-variables in group i the R-variables of that group are scanned until an allocation of rooms is found satisfying the constraints -

(i) lecturer 1 may take class 1 only in rooms 1 or 4.
(ii) lecturer 2 may not take any class in room 3.
(iii) lecturer 3 may not take any class in room 3.
(iv) lecturer 5 may take any class only in room 3.

The complete scan up to the stage where the first possible complete timetable is found is given below:

Period i matrix

i=1	1	2	3	4	5	β
1	8	9	3	0	0	
2	2	0	12	6	0	
3	4	5	0	7	4	
4	6	3	5	0	6	
α	↑		↑			

i=2	1	2	3	4	5	β
1	7	9	3	0	0	
2	2	0	11	6	0	
3	4	4	0	7	4	
4	6	3	5	0	5	
α	↑		↑			

i=3	1	2	3	4	5	β
1	~~8~~	9	3	0	0	
2	2	0	~~10~~	6	0	
3	4	~~3~~	0	7	4	
4	6	3	5	0	~~4~~	
α	↑		↑			

Assignments

Period 1.

1 3 2 5 , 1 2 3 4 (1)
1 3 2 5 , 1 2 4 3 (2)

Period 2.

1 3 2 5 , 1 2 3 4 (3)
1 3 2 5 , 1 2 4 3 (4)

Period 3.

2 1 4 3 , 1 2 3 4 (5)

Period i matrix

i=4	1	2	3	4	5	β
1	6	8	3	0	0	
2	1	0	10	6	0	
3	4	3	0	6	4	
4	6	3	4	0	4	
α	↑		↑			

i=5	1	2	3	4	5	β
1	5	8	3	0	0	
2	1	0	9	6	0	
3	4	2	0	6	4	
4	6	3	4	0	3	
α	↑		↑			

i=6	1	2	3	4	5	β
1	~~4~~	8	3	0	0	
2	1	0	~~3~~	6	0	
3	4	~~1~~	0	6	4	
4	6	3	4	0	~~2~~	
α	↑		↑			

i=7	1	2	3	4	5	β
1	4	7	3	0	0	
2	0	0	8	6	0	
3	4	1	0	5	4	
4	6	3	3	0	2	
α	↑		↑			

i=8	1	2	3	4	5	β
1	3	7	3	0	0	
2	0	0	7	6	0	
3	4	0	0	5	4	
4	6	3	3	0	1	
α	↑		↑			

Assignments

Period 4.

1 3 2 5 , 1 2 3 4 (6)

1 3 2 5 , 1 2 4 3 (7)

Period 5.

1 3 2 5 , 1 2 3 4 (8)

1 3 2 5 , 1 2 4 3 (9)

Period 6.

2 1 4 3 , 1 2 3 4 (10)

Period 7.

1 3 2 5 , 1 2 3 4 (11)

1 3 2 5 , 1 2 4 3 (12)

Period 8.

1 3 4 2 , 1 2 3 4 (13)

Period i matrix

i=9	1	2	3	4	5	β
1	∅	7	3	0	0	
2	0	0	∅	6	0	
3	4	0	0	4	4	
4	6	2	3	0	1	
α	↑		↑			

Assignments

Period 9.

2 4 1 3 , 1 2 3 4 (14)

i=10	1	2	3	4	5	β
1	2	6	3	0	0	
2	0	0	6	5	0	
3	3	0	0	4	4	
4	6	2	2	0	1	
α	↑		↑			

Period 10.

1 3 4 2 , 1 2 3 4 (15)

i=11	1	2	3	4	5	β
1	1	6	3	0	0	
2	0	0	5	5	0	
3	3	0	0	3	4	
4	6	1	2	0	1	
α	↑		↑			

Period 11.

1 3 4 2 , 1 2 3 4 (16)

i=12	1	2	3	4	5	β
1	∅	6	3	0	0	
2	0	0	∅	5	0	
3	3	0	0	∅	4	
4	6	∅	2	0	1	
α	↑		↑			

Period 12.

2 4 1 3 , 1 2 3 4 (17)

i=13	1	2	3	4	5	β
1	0	5	3	0	0	
2	0	0	4	4	0	
3	2	0	0	2	4	
4	(6)	0	1	0	1	
α	↑		↑			

Period 13.

2 3 4 1 , 1 2 3 4 (18)

Period i matrix

i=14	1	2	3	4	5	β
1	0	~~4~~	3	0	0	
2	0	0	3	4	0	
3	2	0	0	1	4	
4	(5)	0	1	0	1	
α	↑		↑			

i=15	1	2	3	4	5	β
1	0	4	2	0	0	
2	0	0	3	3	0	
3	2	0	0	1	3	
4	~~4~~	0	1	0	1	
α	↑		↑			

i=16	1	2	3	4	5	β
1	0	3	2	0	0	
2	0	0	2	3	0	
3	1	0	0	1	3	
4	(4)	0	1	0	0	
α	↑		↑			

i=17	1	2	3	4	5	β
1	0	~~2~~	2	0	0	
2	0	0	~~1~~	(3)	0	
3	1	0	0	0	(3)	
4	(3)	0	1	0	0	
α	↑		↑			

i=18	1	2	3	4	5	β
1	0	2	1	0	0	
2	0	0	1	2	0	
3	1	0	0	0	2	
4	~~2~~	0	1	0	0	
α	↑		↑			

Assignments

Period 14.

3 4 5 1 , 1 2 3 4 (19)

Period 15.

2 3 1 5 , 1 2 3 4 (20)

2 3 1 5 , 1 2 4 3 (21)

Period 16.

2 3 4 1 , 1 2 3 4 (22)

Period 17.

3 4 5 1 , 1 2 3 4 (23)

Period 18.

2 4 1 3 , 1 2 3 4 (24)

Period i matrix

i=19	1	2	3	4	5	β
1	0	1	1	0	0	
2	0	0	1	⊘	0	
3	0	0	0	0	(2)	
4	(2)	0	0	0	0	
α	↑		↑			

i=20	1	2	3	4	5	β
1	0	⊘	(1)	0	0	
2	0	0	0	(1)	0	
3	0	0	0	0	(1)	
4	(1)	0	0	0	0	
α	↑		↑	↑	↑	

Assignments

Period 19.

2 3 5 1 , 1 2 3 4 (25)

Period 20.

3 4 5 1 , 1 2 3 4 (26)

The complete timetable obtained in the above scan appears in Figure A·2·1.

In the case of the simple problem considered here the scan is seen to progress from the first period to the last without ever having to go back a step. Normally, however, when solving a problem in this way it would be found at some stage that a suitable arrangement could not be found for a certain period, say period r . In this case the values assigned to the variables in group r-1 , corresponding to period r-1 , would be further scanned to find an alternative arrangement for period r-1 . If no other arrangement should exist for period r-1 the variables in group r-2 would be further scanned. Thus, the scan progresses in the same way as the abbreviated scan for problems in two-valued logic described in Section V.

		MON		TUE		WED		THUR		FRI	
		Period 1		Period 5		Period 9		Period 13		Period 17	
	Class 1	L1	R1	L1	R1	L2	R1	L2	R1	L3	R1
M	Class 2	L3	R2	L3	R2	L4	R2	L3	R2	L4	R2
O	Class 3	L2	R4	L2	R4	L1	R3	L4	R3	L5	R3
R	Class 4	L5	R3	L5	R3	L3	R4	L1	R4	L1	R4
N		Period 2		Period 6		Period 10		Period 14		Period 18	
I	Class 1	L1	R1	L2	R1	L1	R1	L3	R1	L2	R1
N	Class 2	L3	R2	L1	R2	L3	R2	L4	R2	L4	R2
G	Class 3	L2	R4	L4	R3	L4	R3	L5	R3	L1	R3
	Class 4	L5	R3	L3	R4	L2	R4	L1	R4	L3	R4
		Period 3		Period 7		Period 11		Period 15		Period 19	
A	Class 1	L2	R1	L1	R1	L1	R1	L2	R1	L2	R1
F	Class 2	L1	R2	L3	R2	L3	R2	L3	R2	L3	R2
T	Class 3	L4	R3	L2	R4	L4	R3	L1	R4	L5	R3
E	Class 4	L3	R4	L5	R3	L2	R4	L5	R3	L1	R4
R		Period 4		Period 8		Period 12		Period 16		Period 20	
N	Class 1	L1	R1	L1	R1	L2	R1	L2	R1	L3	R1
O	Class 2	L3	R2	L3	R2	L4	R2	L3	R2	L4	R2
O	Class 3	L2	R4	L4	R3	L1	R3	L4	R3	L5	R3
N	Class 4	L5	R3	L2	R4	L3	R4	L1	R4	L1	R4

L_μ - Lecturer μ ; R_μ - Room μ.

FIGURE A·2·1.

REFERENCES

1. D.G. Prinz and J.B. Smith -
Chapter 15 of "Faster than Thought", Sir Isaac Pitman and Sons London, 1955.

2. A. Rose. -
Many-Valued Logical Machines, Proceedings of the Cambridge Philosophical Society, Volume 54, Part 3, pp.307-321, July 1958.

3. B. Rosser. -
On the Many-Valued Logics, American Journal of Physics, Volume 9, No. 4, pp. 207-212, 1941.

4. D.A. Bočvar.-
On a Three-Valued Calculus and its Application to the Analysis of Contradictions, Matématičeskij Sbornik, Volume 4, pp287-308 1939. For a review of this paper in English see Journal of Symbolic Logic, Volume 4, pp.98-99, 1939.

5. D.M. McCallum and J.B. Smith.
Mechanized Reasoning, Electronic Engineering, April 1951.

6. D.M. McCallum and J.B. Smith.
Feedback Logical Computers, Electronic Engineering, Dec. 1951.

7. J.D. Cowan.
D.I.C. Thesis, Imperial College, 1959.

8. W.S. Jevons.
On the Mechanical Performance of Logical Inference.
Philosophic Transactions of the Royal Society, Vol. 160.
p.497, 1870.

9. W. Keister, A.E. Ritchie and S.W. Washburn.
"The Design of Switching Circuits", D. Van Nostrand Co., Inc., New York, 1951.

10. H. Reichenbach.
"Philosophic Foundations of Quantum Mechanics", University of California Press, 1944.

11. E.L. Post.
Introduction to a General Theory of Propositions, American Journal of Mathematics, Vol. 43, pp.163-185, 1921.

12. C.I. Lewis and C.H. Langford.
"Symbolic Logic", Dover Publications Inc., New York, 1950.

13. J.B. Rosser and A.R. Turquette.
"Many-Valued Logics", North-Holland Publishing Co., Amsterdam, 1952.

14. P. Rosenbloom.
"The Elements of Mathematical Logic", Dover Publications Inc., New York, 1950.

15. E.C. Berkeley.
"Giant Brains or Machines that Think", Berkeley Enterprises, Inc., New York, 1949.

16. A. Rose.
Applications of Logical Computers to the Construction of Electrical Control Tables for Signalling Frames, Zeitschrift fur Mathematische Logik und Grundlagen der Mathematic, Band 4, pp. 222-243.

17. R.S. Ledley.
Digital Computational Methods in Symbolic Logic, with Examples in Biochemistry, Proceedings of the National Academy of Sciences, Vol. 41, Number 7, pp.498-511, July 1955.

18. R.S. Ledley.
Mathematical Foundations and Computational Methods for a Digital Logic Machine. Journal of the Operations Research Society of America. Vol. 2, Number 3, August 1954.

19. A. Porter and P.K.T. Vaswani.
The Optimization of Logical Goal-Seeking Procedures, Journal of Electronics and Control. Volume 6, Number 22, pp. 168-185, February 1959.

20. M. Gardner.
"Logical Machines and Diagrams", McGraw Hill Book Co., Inc., New York, 1958.

21. J.H. Calderwood and A. Porter.
Pattern Recognition in the Synthesis of Complex Switching Systems. Journal of Electronics and Control, Volume 4, Number 5, pp. 466-480, May 1958.

22. C.Y. Lee and W.H. Chen.
Several-Valued Combinational Switching Circuits.
Transactions of the American Institute of Electrical Engineers Volume 75, July 1956.