Notre Dame Journal of Formal Logic
Volume IV, Number 4, October 1963

PROGRAMMING THE FUNCTIONS OF FORMAL LOGIC. II
(Multi-valued Logics)

S. SUMMERSBEE and A. WALTERS

We consider here in greater detail a problem mentioned en passant in our previous paper[1] viz., the programming of problems in multi-valued logic for solution by means of a digital computer. Once again, to give the inquiry a definite direction a particular problem is chosen for discussion. This problem has already been discussed by Rose[2] but from a different point of view; it has already been attacked from an entirely different direction, and some solutions obtained, by programme on DEUCE and ACE.[3,4]

The discussion divides naturally into three parts: a description of the machine which is to be programmed to obtain a solution to the problem; the problem to be solved and the formal multi-valued logic used to obtain a solution.

I The Machine In our previous paper we discussed the use of a digital computer working in the binary scale of notation i.e. each number or "machine word" is represented in the machine in the form

$$\sum_{i=1}^{n-1} a_i 2^{i-1} \quad (1 \leq i \leq n-1;\ a_i = 0 \text{ or } a_i = 1).$$

Because of such number representation a binary machine is particularly suitable for operating on the values of two-valued propositional variables.

Our present problem is solved in terms of multi-valued logic, although a multi-valued logic can be represented within a two-valued system. The calculation of the values of logical functions is easier of the radix of the number system coincides with that of the n-valued logic used. However, a decimal machine is easier to use than a binary machine for programmes written in an n-valued logic.

---

1. This Journal, v. III No. 3. No knowledge of this paper is assumed.
2. cf ref. (1) at end.
3. cf ref. (2) at end.
4. See ref. (4).

Received January 22, 1963

The machine in question uses an instruction form which consists of a function part and two addresses e.g. the instruction

$$10\ a_1\ a_2$$

may be interpreted as

"add the contents of address a_1 to the contents of address a_2 and return the result to address a_2."

The instruction

$$15\ a_1\ a_2$$

may be interpreted as

"test the result of the previous operation: if positive take a_1; if negative take a_2; if neither take next instruction."

The word length of the machine is ten decimal digits, one of which is used as a sign indicator. Positive numbers have a zero in the sign place and negative numbers have a nine in the sign place and the number stored as a nine's complement. The machine has a parallel adder and a ferrite-core memory store operating at a 6 μs cycle time, both of these features making for relatively fast operating times. The input of information is by means of punched cards, this speed being 3600 instructions per minute.

These details are sufficient to show that the machine is a fairly conventional decimal digital computer. Although logical functions are available, e.g. logical "and" (collate) and "negate" no use will be made of these functions in what follows. Indeed, these functions may be regarded as something of an anomaly in a machine operating with numbers in the form

$$\sum_{i=1}^{n-1} a_i 10^{i-1} \quad (1 \leq i \leq n-1;\ a_i = 1, 2 \ldots 9).$$

II The Problem Suppose that in a department of a school or college there are l lecturers and c classes to be taught by these teachers. If any teacher can take any class at any teaching period of the week a simple allocation of one teacher to each class will solve the problem, provided that the number of classes can be adjusted to suit the number of teachers, or vice versa (- we neglect the problem of classroom allocation). In fact however there are other considerations which must be taken into account. If the timetable to be compiled is for a department it usually means that only a fewer number of teaching periods are available, than the total number; and moreover the available teaching periods would be scattered throughout the week. Usually two classes cannot be merged. Some classes can (sometimes) be merged but these are exceptions and they merge for single lectures only.

It is usually the case too, that the total teaching time per week is limited for the teacher, to less than the total time available for teaching. For example, there may be thirty hours available each week for teaching but each teacher is limited to twenty hours teaching ("class contact") each

week. Also, there will be some teachers who will lecture to certain classes but not to others, and some teachers may be available for teaching only during certain periods of the week, and not for other periods.

We can then set out a list of "conditions" which must be satisfied before a timetable can be said to be compiled. Furthermore, there very often is more than one solution to a given timetable problem, and some solutions may be more acceptable to the lecturers and classes than other solutions. For example, if the last lecture of the day commences at 5 p.m. and the first lecture begins at 9 a.m. a solution to a particular timetable problem may give the first and last lectures of each day to one particular lecturer. While this may be a satisfactory formal solution to the problem it probably would not be regarded as very satisfactory by the lecturer concerned.

The problem of compiling a timetable may therefore be regarded as the problem of finding a satisfactory arrangement of period-class-lecturer throughout the week given (i) the number of lecturers (ii) the number of classes (iii) the total number of lecture periods per week; and a set of conditions such as "no lecturer may teach two classes simultaneously" "no lecturer may teach for more than n-hours per week" "lecturer l_i is not available during teaching periods p_x and p_y," etc.

When the timetable problem has been solved for a particular department it usually has to be fitted into the timetables for other departments in the College, and then an allocation of classrooms, laboratories and workshops made to the appropriate period-class-lecturer arrangement.

It might be worth mentioning that this is only an example of a more general type of scheduling problem; air traffic control and scheduling factory work are other examples.

III The Logic The formal logic employed is multi-valued i.e., its propositional variables can take values other than *1* or *0* (or other than *T* or *F*). The range of values which the propositional variables can take must be fixed. For example, a value v may lie in the range $0 \leqslant v \leqslant 1$, or the range may be so chosen that any integral number greater than *1* may be a value. For convenience we adopt the latter course i.e., the range of values of v will be $1 \leqslant v \leqslant m$, where m defines the logic i.e. the logic is m-valued. In multi-valued logic there are functions which are analogous to the functions of two valued logic. For example, a function of two variables, $f(P,Q)$, may be defined in m-valued logic which is analogous to the function conjunction, in two-valued logic, and in fact will reduce to conjunction in the two-valued case when $m = 2$. There is however a difficulty here since there is no accepted notation for the functions of multi-valued logic. In the two-valued case the notation of Łukasiewicz will be used, "*C*" for "*if - - - - - then - - - - -*"; "*K*" for "*- - - - - and - - - - -*"; "*A*" for "*either - - - - - or - - - - -*"; "*N*" for "*not - - - - -*". Similar symbols will be used for the multi-valued case except that a bar will be written above the appropriate symbol. Thus in the multi-valued case, the "conjunction" of *P* and *Q* will be written as "$\overline{K}PQ$".

In two-valued logic, of the two possible values which a propositional variable can take, one is a "designated" value; the one attached to propo-

sitions which are asserted—the other is attached to propositions which are denied. Because in two-valued logic there are two values and because assertion and denial are two cases the value attached to propositions which are asserted can be taken to be a sign of assertion itself. However, in the multi-valued case this systematic ambiguity no longer holds. A propositional variable can have any of the values $1 \leq v \leq m$ attached to it, and here there is no sharp distinction between those propositions which are asserted and those which are denied. The following rule[5] may be adopted to meet this situation; there is a truth-value v such that $1 \leq v \leq m$, and the truth-values $1 - - - - - v$ are referred to as "designated" while the truth-values $v + 1, - - - - - m$ are undesignated.

These values may be interpreted in the following way. First note that "*1*" is always a designated value and that "m" never is, so that these two represent the extreme ends of a scale of values, with the others lying inbetween. At one end of the scale there is the value "truth" (*1*), or the greatest assertability, while at the other end of the scale there is "falsehood" (m) or the least assertability. Between these two extremes of any two values the lesser one has the greatest assertability. For example, in a four-valued logic a proposition P may take any one of four values *1*, *2*, *3*, *4*; these may be interpreted as

$P = 1$; P is true
$P = 4$; P is false
$P = 2$; P is more likely to be true than not.
$P = 3$; P is more likely to be false than not.

This interpretation is only one of many and in fact below a different one will be used.

In the two-valued case the truth-table for APQ is

TABLE I

P	Q	APQ
1	*1*	*1*
1	*0*	*1*
0	*1*	*1*
0	*0*	*0*

that is, APQ has the highest value of P and Q or, if p and q are the truth-values of P and Q, then the truth-value of APQ is max (p,q).

Since however "*1*" is always to be a designated value in multi-valued logic APQ is taken as min (p, q). For example, in the three-valued case

5. cf ref. (3) at end.

TABLE II

P	Q	$\bar{A}PQ$
1	1	1
1	2	1
1	3	1
2	1	1
2	2	2
2	3	2
3	1	1
3	2	2
3	3	3

Conjunction and negation for multi-valued logic is given by $\max(p,q)$ and $((m - p) + 1)$. For the three-valued logic these values are given in

TABLE III

P	Q	$\bar{N}P$	$\bar{K}PQ$	$\bar{N}Q$	$\overline{\bar{A}\bar{N}P\bar{N}Q}$	$\overline{\overline{NA\bar{N}}P\bar{N}Q}$
1	1	3	1	3	3	1
1	2	3	2	2	2	2
1	3	3	3	1	1	3
2	1	2	2	3	2	2
2	2	2	2	2	2	2
2	3	2	3	1	1	3
3	1	1	3	3	1	3
3	2	1	3	2	1	3
3	3	1	3	1	1	3

The last three columns of the table give the values of $\bar{N}Q$, $\overline{\bar{A}\bar{N}P\bar{N}Q}$ and $\overline{\overline{NA\bar{N}}P\bar{N}Q}$ and a comparison of the last column with the fourth column shows that the equivalence of KPQ with $NANPNQ$ of two-valued logic holds for multi-valued logic.

In two-valued logic the equivalence of two functions is defined as $KCPQCQP$. If a table is drawn up for a three-valued logic then the values of this function will be as TABLE IV, where the value of CPQ is given by $\max(1,(q - p) + 1)$.

TABLE IV

P	Q	$\overline{C}PQ$	$\overline{C}QP$	$\overline{KC}PQ\overline{C}QP$
1	1	1	1	1
1	2	2	1	2
1	3	3	1	3
2	1	1	2	2
2	2	1	1	1
2	3	2	1	2
3	1	1	3	3
3	2	1	2	2
3	3	1	1	1

It will be noticed that when P and Q have the same values the value of the equivalence function is 1, i.e. $\overline{KC}PQ\overline{C}QP$ takes the designated value; when the difference in values of P and Q is greatest (when $p = 3$ and $q = 1$, or when $p = 1$ and $q = 3$) then the value of $\overline{KC}PQ\overline{C}QP$ is m and for other values of P and Q, $\overline{KC}PQ\overline{C}QP$ takes a value intermediate between 1 and m. If "$\overline{KC}PQ\overline{C}QP$" is abbreviated to "$EPQ$" (taking a value e), then this function may be interpreted as

$e = 1$; P and Q are equivalent
$e = 2$; P and Q are nearly equivalent
$e = 3$; P and Q are not equivalent

As will be shown below the functions cited above will be used extensively in the solution of the time-table problem. It may not be out of place here to discuss the manner in which the machine handles such functions.

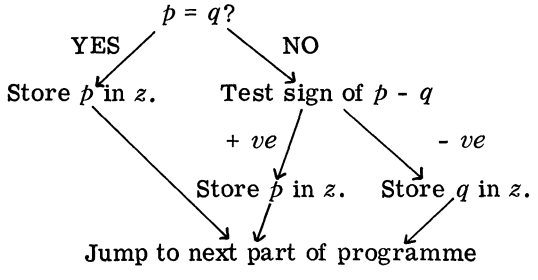
The machine has a "word length" of ten decimal digits and for our purpose may be taken as operating in the range

$$+999,999,999 \geqslant x \geqslant -999,999,999$$

and consequently if the number which each register holds can represent the value of one of the variables of an m-valued logic, a logic of $m = 999,999,999$ values can be dealt with by the machine.

The machine can be programmed to calculate the values of the functions given above from the values of the variables.

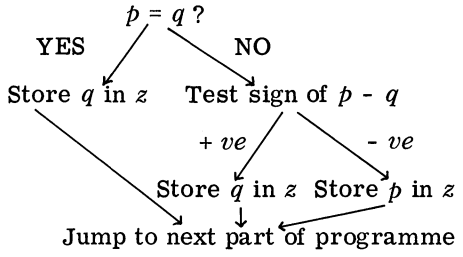
If the values of P and Q are given, the value of $\overline{K}PQ$ can be calculated from the rule max (p,q). The flow diagram, or logic, of the calculation is given by the machine programme.



In the language of the machine the calculation is as follows: $c(x) = p$, $c(y) = q$ $C(z) = \overline{K}PQ$ (at end).

Programme Step	*Function*	*Address* $1(a_1)$	*Address* $2(a_2)$	
n	13	x	y	p-q. Result used for test.
$n+1$	15	$n+2$	$n+4$	Test sign: $+ve$ jump a_1 $-ve$ jump a_2
$n+2$	14	x	z	$p \rightarrow z$
$n+3$	18			Return link to main programme.
$n+4$	14	y	z	$q \rightarrow z$
$n+5$	18		$n+3$	Jump to $(n+3)$.

The value of $\overline{A}PQ$ can be calculated in a similar way:



Programme Step	*Function*	*Address* $1(a_1)$	*Address* $2(a_2)$	
n	13	x	y	p-q. Result used for test.
$n+1$	15	$n+2$	$n+4$	Test sign: $+ve$ jump a_1; $-ve$ jump a_2.
$n+2$	14	y	z	$q \rightarrow z$
$n+3$	18			Return link to main programme
$n+4$	14	x	z	$p \rightarrow z$
$n+5$	18		$n+3$	Jump to $(n+3)$.

The time of execution of both of these programmes will depend on which branch of the programme is followed.

The maximum time taken for both of these programmes to be executed is 56 micro-seconds, and the minimum time is 50 micro-seconds. If now either of Table II or Table III is examined it will be seen that for a three-valued logic the short branch of the programme will be followed for one third of the total number of executions of the programme (i.e. in just those cases where p - q). And the long branch of the programme will be followed in the remaining number of executions of the programme. Hence a mean time of execution would be 54 micro-seconds.

The time taken to calculate the value of $\overline{N}P$ from the value of P is 72 micro-seconds, while the mean time of execution of a programme to calculate the value of $\overline{C}PQ$ from the values of P and Q is 68 micro-seconds. To calculate $\overline{E}PQ$ takes a mean time of 280 micro-seconds.

From TABLE II it is clear that in order to calculate all the values of $\overline{A}PQ$, nine applications of the programme for calculating $\overline{A}PQ$ are necessary and the total time taken will be (9×54 μs) plus the time required for the master programme arranging the count etc. In general if there are n variables in an expression, each of which can take m values, there will be m^n rows in the "truth-table" for the expression (and m^{m^n} possible column entries in the "truth-table"). Suppose that we consider an expression of three variables $\overline{A}P\overline{A}QR$ in a three valued logic. To calculate the value of a single row in the "truth-table" of $\overline{A}P\overline{A}QR$, two applications of the $\overline{A}$ routine are necessary. If there are n variables in an expression it will require (n - 1) applications of the routines for calculating the values of $\overline{A}$, $\overline{K}$, $\overline{C}$, $\overline{E}$ and n applications of the routine for calculating the value of $\overline{N}$. As an example, suppose that instead of two variables in Table II there were three variables, there would be ($3^3 = 27$) rows in the table. Each row in the table would require ($3 - 1 = 2$) applications of the $\overline{A}$ routine, and the total number of applications of $\overline{A}$ would be (2×27) i.e. the time taken to calculate the entries in a column of the table for $\overline{A}P\overline{A}QR$ would be $2 \times 27 \times 54$ μs plus the time taken for the master programme.

The problem which has been chosen for discussion is that of compiling a timetable for a College department. To serve as an example and to keep the problem as close to reality as possible, the facts quoted to illustrate the general argument refer to an actual College Mathematics Department. There are actually forty-three classes of students to be covered, and thirteen lecturers available for teaching. Some classes can only be taught by certain lecturers. For example, a class taking Applied Mathematics (Aerodynamics) would only be taught by those lecturers whose special fields include this subject. The classes can meet on any of the first five days of the week (Monday to Friday), the first lecture starting at 8:45 a.m. and the last at 5:30 p.m., but not all lectures are of the same duration.

Each lecture period must be identified and one way of doing this would be to re-time the lecture periods so that each lecture will consist of multiples of the shortest lecture period. For example, if the shortest lecture period were half-an-hour, then this is counted as a unit period and a two-hour lecture would count as four periods. In this way a day would comprise fifteen periods, and there would be seventy-five periods in a working week. However, by keeping lecture periods of unequal length the number of periods in a day is reduced to nine. The starting times will be 8:45; 9:45; 11:00; 11:30 a.m., 2:00; 3:00; 3:30; 4:00; 5:30 p.m. and each period in the week will be identified with a number. There are five days on which lectures can be given, and since $9 \times 5 = 45$, we shall be concerned with a forty-five valued logic. In general if λ are the lecture periods and μ the lecturing days then a timetable will be compiled by means of a $\lambda\mu$-valued logic. Let each lecturer be denoted by "l_j" ($1 \leqslant j \leqslant 13$) and each class by "$c_i$" ($1 \leqslant i \leqslant 43$).

Some lectures are repeated during a week and it is assumed that the timetable is exactly the same for each week of the term. Since there are λ lecture periods in each day and μ days of the week, any particular lecture period k will be numbered $1 \leqslant k \leqslant \lambda\mu$. The first lecture on each successive day beginning with Monday is numbered 1, 2, 3, 4, 5; the second lecture period on each successive day 6, 7, 8, 9, 10 etc., or, if i is the i-th lecture on the j-th day and n the lecture number $n = (i - 1)\ \mu + j$.

Each lecturer l_j may take any particular class more than once, or may not take a particular class at all. Suppose that l_j lectures to c_i at least once then let the number of lectures given to c_i by l_j be $f(i,j)$. Instead of propositional variables "P" "Q" etc., we use "$P_{i,j,k}$" where "$P_{i,j,k}$" denotes the proposition 'class i has lecturer j for the k-th time', with $1 \leqslant i \leqslant 43$; $1 \leqslant j \leqslant 13$; $1 \leqslant k \leqslant f(i,j)$. And if l_j takes c_i for the k-th time during the period whose number is v then $P_{i,j,k}$ will take the value v.

Let us take a few examples:

Lecturer l_4 takes class c_1 for the first period on Monday hence $k = 1$ and $P_{1,4,1}$ takes the value 1.

Lecturer l_4 takes class c_1 for the third time during the twenty-second period, hence $k = 3$ and $P_{1,4,3}$ takes the value 22.

In the example the greatest number of times which any one lecturer meets the same class is five. This is lecturer l_4 and class c_1. Lecturer l_4 meets class c_1 on five occasions, and all other lecturers meet their classes on less than five occasions. The number of times that a particular lecturer will meet a particular class will not in general be known until the timetable has been compiled. However an upper limit may be placed on k, thereby making this one of the conditions which an acceptable timetable must satisfy. For the timetable quoted the greatest value k takes is 5, and this may be taken as satisfying the condition that "no lecturer may meet any class for more than five periods during any one week." Propositions are denoted by "$P_{i,j,k}$" and since the maximum values of i, j and k are 43, 13 and 5 respectively there will be $43 \times 13 \times 5 = 2795$ independent propositions.

Let us consider some of these propositions in more detail. They may each take one of 45 values, and we may imagine a table showing these propositions in rows followed by a function of two of them e.g.,

TABLE V

$P_{1,1,1}$ - - - - -	$P_{a,b,c}$ - - - - -	$P_{x,y,z}$ - - - - -	$f(P_{a,b,c}\ P_{x,y,z})$
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
v_1	v_2	v_3	v_4
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

If $P_{a,b,c}$ and $P_{x,y,z}$ both have the same value, say v, then it means that lecturer 1_b meets class c_a in the same period that lecturer 1_y meets class c_x. Hence if lecturer 1_i is not to teach two classes simultaneously the timetable must satisfy the condition that $\overline{NE}P_{a,b,c}\ P_{x,y,z}$ for $a \neq x$ and $y = b$.

Suppose that lecturer 1_m cannot lecture to classes c_{n_1} - - c_{n_x} (because the lectures given to these classes are of a specialist nature outside the field of lecturer 1_m) then each proposition $P_{n_1,m,k}$; $P_{n_2,m,k}$; $P_{n_3,m,k}$ - - - $P_{n_x,m,k}$ represents a condition for the timetable which is not acceptable i.e. the timetable must satisfy the condition $\overline{N}\,\overline{A}\,P_{n_1,m,k}\,P_{n_2,m,k}$ - - - $P_{n_x,m,k}$. These are now two conditions which the timetable must satisfy. We can abbreviate these to Π_1 and Π_2. By setting out systematically conditions such as Π_1 and Π_2 which an acceptable timetable must satisfy, a set of such conditions

$$\Pi_1 \text{ - - - - } \Pi_n$$

can be laid down. The conjunction of Π_1 - - - - Π_n would represent a final column in a truth-table the computation of whose values would represent the compiling of a timetable. We call the conjunction of Π_1 - - - - Π_n "condition I".

We mentioned the case where a particular lecturer is only available for teaching during certain periods. Suppose this lecturer to be 1_α and that he is not available for periods x, y and z. This means that propositions of the form $P_{i,\alpha,k}$ must not take values x, y and z in an acceptable timetable. And hence that certain rows in the truth-table can be deleted. For example, suppose that TABLE VI represents the final computation for a truth-table.

TABLE VI

$P_{1,1,1}$ - - - -	$P_{i,\alpha,k}$ - - - -	$P_{43,13,5}$ - - - -	$f(P_{1,1,1}$ - - - -)
1	1	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	x	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	y	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	z	.	.
.	.	.	.
.	.	.	.
.	.	.	.

Where the column $f(P_{1,1,1}$ - - - -) represents possible timetables fulfilling all the required conditions. The entries with values x, y, z under "$P_{i,\alpha,k}$" represent an undesirable feature of the timetable if lecturer 1_α is not available during periods x, y, z. The number of rows in the truth-table can correspondingly be reduced.

Quite often it is the case with a departmental timetable that certain periods for particular classes are reserved for periods of instruction in other departments. For example, laboratory periods may be confined to afternoons, and consequently any class which has a laboratory period included in its course of instruction would not be available for mathematics lectures during the afternoon i.e. for the classes $P_{i,j,k}$'s taking values greater than *21* are not acceptable conditions for a timetable. Once again the rows of the table where these values occur in the column referring to those particular classes can be deleted.

There are now two sets of conditions which a satisfactory timetable must fulfil. The first set concerns the periods of the week during which classes and teachers are available. This we have called "condition I". The second concerns the allocation of teachers to classes, such as those outlined in the immediately preceding paragraphs. This we call "condition II". It will be seen that condition II reduces the number of rows in the truth-table, while condition I represents a computation from entries in the rows of the table. The final column of the table would give the arrangement of all possible timetables satisfying conditions I and II. Not all of these timetables would be equally acceptable. The degree of "acceptability" may be indicated by the value which a proposition (or function of propositions) takes e.g., $P = 1$ may be interpreted as "P is acceptable." Hence if the "degree of acceptability" is fixed in advance, say $P = \beta\ (1 \leqslant \beta \leqslant 10)$ then when the first acceptable value is found in the final column of the truth-table the computation may be stopped.

We may now make use of the actual timetable compiled (by hand) and use it to form an estimate of the time required to carry out an actual computation. Of course, the time which a machine actually takes to compile a timetable could only be found either by having the machine carry out the computation. Or by sorting out each step in the computation and then, by summing the times taken by the machine to carry out every operation, to estimate the machine time. It may be that in a particular case the machine will find an acceptable timetable in the first line of the truth-table. This is unlikely, and it is equally unlikely that the whole of the truth-table must be worked through before an acceptable timetable is found. But at least the time taken to compute every line in the table must be provided for, and by considering the time required to compute all the rows in the table, a conservative time estimate may be made for the whole calculation.

In the timetable taken as an example there are forty-three classes, thirteen lecturers and each lecturer may teach a particular class on as many as five occasions (but no more). This gives a maximum of 2795 independent propositions i.e. 2795 columns in the truth-table. However, calculation by hand from the actual allocation of teachers to classes shows that in fact this number is reduced to 179. Furthermore, since each teacher is limited in the number of class-contact hours to (on average) eighteen hours, and each hour corresponds (on average) with a single teaching period, each teacher will be free for 27 periods of the teaching week. Hence, at most there will be $27 \times 13 = 351$ rows which can be deleted from the truth-table.

Since there are 179 propositions each of which can take one of forty-five values, there will be in all 45^{179} rows in the truth-table,

$$\text{i.e. approx. } 10^{288}$$
$$\text{less the deleted rows i.e.,}$$
$$> 10^{286} \text{ rows}$$

Let us suppose that there are no more than three conditions to be satisfied by the truth-table, and each one takes no more than 50 μs (i.e. the minimum time to compute $\overline{A}\ P\ Q$). Because there are 179 columns in the table there will be 178 applications of the sub-routine concerned i.e. to calculate the value of one column (corresponding to one of the conditions) of the table would take $178 \times 50\ \mu$s or approximately 9×10^{-3} seconds. As there are three conditions to be satisfied (assuming that they are independent) the time taken to calculate an entry in the final column of the truth-table would be greater than

$$3 \times 9 \times 10^{-3} = 27 \times 10^{-3} \text{ seconds.}$$

There are more than 10^{286} rows in the table and hence the time taken to calculate a value for every row would be greater than

$$27 \times 10^{-3} \times 10^{286} \text{ seconds}$$
$$\text{or} \backsimeq 10^{277} \text{ years.}$$

It will be appreciated that it is an extreme case where every row of the truth-table would have to be calculated before an acceptable solution was found. Never-the-less the time estimate omits any allowance for organizing the machine programme (quite often this can exceed the actual time taken to perform a computation). The length of time to carry out a computation depends on (a) the number of propositions and (b) the value of the logic (e.g.,a two-valued logic would give a 2^{179}-rowed table as compared with the 45^{179}-rowed table under consideration.

As in our previous paper we conclude that if more than a few propositional variables are involved the evaluation, of the desired valued of a propositional function, by means of a truth-table algarithm is too slow. Various methods have been suggested to shorten the time taken by the computation but even so they are restricted to fewer variables than are often desired in practice. We hope to discuss these in a further paper.

REFERENCES

[1] A. Rose, Many-valued Logical Machines. *Proc. of Camb. Phil. Soc.* Vol. 54 pp. 307-321 1958.

[2] J.S. Appleby, D.V. Blake and E. A. Newman, Techniques for Producing School Timetables. *The Computer Journal*, Vol. III No. 4.

[3] J. B. Rosser and A. R. Turquette, *Many-Valued Logics*. North-Holland Publishing Co. 1952.

[4] The following appeared in *"The Computer Journal"*, vol. 5 No. 3 after this paper had been written.

"Considerable interest was shown in the paper "The Construction of Class-Teacher Time Tables" presented by Professor Gottieb at the recent I.F.I.P. Congress in Munich. The regular discussion was followed by a special meeting at which some 40 persons were present. It became evident that work on this subject is in progress at many institutions. A list of groups interested in timetable construction was drawn up and copies are available from the Assistant Secretary of the British Computer Society at (Finsbury Court, Finsbury Pavement, London E.C.2.) Professor Gottieb has undertaken to circulate any short summaries of progress which are submitted to him, to the organizations listed."

The Computing Laboratory,
Letchworth College of Technology,
Letchworth, Hertfordshire, Gt. Britain.