ENGINEERING RESEARCH INSTITUTE
UNIVERSITY OF MICHIGAN
ANN ARBOR

TRUTH-FUNCTION EVALUATION USING THE POLISH NOTATION

By

Arthur W. Burks

Don W. Warren

Jesse B. Wright

Project M828

BURROUGHS ADDING MACHINE CO.
DETROIT, MICHIGAN

July 8, 1952

TABLE OF CONTENTS

ABSTRACT

An inexpensive special-purpose relay machine is proposed (requiring some 40 standard relays, 90 crystal diodes, and a rotary selector switch) which will produce the complete truth-table for a formula 25 characters long containing 10 distinct variables in about 15 minutes machine time.

Some theory is presented concerning the Polish notation, the type of language upon which the machine is based, e.g., theorems concerning the relation between structure and length of a formula and the amount of equipment required for its evaluation.

It is pointed out that the proposed machine is not adaptable to extremely long formulas but that the theory could be embodied in an electronic machine which could handle such formulas and which would have some advantages over comparable versions of existing machines used for this purpose.

ACKNOWLEDGMENT

The authors wish to express their gratitude to Carl H. Pollmar for his many helpful suggestions and constructive criticism throughout the writing of this report.

TRUTH-FUNCTION EVALUATION USING THE POLISH NOTATION

1. INTRODUCTION

Two different methods for the mechanized evaluation of truth-functions (i.e., expressions in the propositional calculus) have been proposed recently. The first is a parallel, analogue method. To each occurrence of a logical connective in the formula to be evaluated is assigned a physical element (rudimentary circuit of relays, vacuum tubes, etc.) which realizes that connective. An input which has two stable states (one for each of the two truth-values "true" and "false") is assigned to each variable occurring in the formula. The inputs and physical elements are then interconnected in accordance with the arrangement of variables and connectives in the formula so that there is a single output wire which, when the inputs represent a given assignment of values to the variables, will represent the truth-value of the formula for that assignment. A counting or stepping device then causes the inputs to represent succesively all possible truth-value assignments to the variables and the output wire will then successively represent the corresponding truth-values of the formula. Two machines embodying this method have been built, one by Kalen and Burkhart (1)‡ and a second by engineers at Ferranti (2). Both machines used relays as their basic components, but it is obvious that special-purpose high-speed electronic machines embodying this same method could be readily designed.

The second method employs a general-purpose computer, the evaluation being done by means of a special sequence of instructions (4). These instructions cause the machine to first assign truth-values to all variables and second reduce down to a single truth-value the resultant formula. This is repeated until all possible assignments of values to variables have been exhausted. The reductions are generally accomplished by arithmetic operations (e.g., negation of a truth-value can be done by subtracting the given truth-value from a constant) but in some instances logical operations might be used (e.g., in some binary machines digit extraction is a form of bitwise logical. conjunction).

‡Parenthesized numbers refer to the bibliography at the end of the volume.

It is difficult to compare computing methods per se, that is, independently of the physical means used to realize them, and yet comparisons which do not abstract from equipment are of little use. With this reservation in mind, consider formulas of great length (e.g., 250 characters) containing a large number of distinct variables (e.g., 25). Even assuming the use of high-speed electronic components, neither of these two methods seems practical for such formulas. The former method requires an excessive amount of equipment, especially if the machine is designed so that setting it up for a particular problem is automatic (e.g., from a tape). The latter method seems to require an excessive amount of computation time, presumably because this is not the type of problem which present general purpose machines were designed to handle efficiently.

In the present report a new method for the evaluation of truth-functions is proposed which does seem to be practical for formulas of great length and many variables and which has other features of interest. This new method is based on the Polish notation. We will first make a few remarks concerning this notation and our use of it. Following that (Section 2) a rather detailed design is presented for a special-purpose relay machine embodying the new method. This machine cannot handle formulas of the size referred to above, but it will serve to illustrate the principles involved. (The design work is further justified by the fact that there is some interest in Philadelphia in constructing it, mainly as a test for certain types of relay equipment.) Section 3 contains some general theorems relevant to the proposed method of truth-function evaluation. Finally, Section 4 considers briefly the use of this method for the evaluation of formulas of great length and many variables by means of high-speed electronic equipment.

In the Polish notation a logical operator is followed by its arguments instead of being placed between them as is normally the case and no parentheses are needed. E.G. '$(p \vee q) \cdot r$' is expressed as 'KApqr', where 'K' stands for conjunction and 'A' for disjunction (alternation).

The machine of section 2 evaluates formulas in a language containing the following primitive symbols:

Ten propositional variables: p, q ,..., y

Six dyadic operators: defined by Table I below

A "left-end-of-formula" symbol: *

TABLE I

Name	Symbol	Truth Values			
First argument	p	0	0	1	1
Second argument	q	0	1	0	1
Conjunction	Kpq	0	0	0	1
Alternation (Inclusive Disjunction)	Apq	0	1	1	1
Conditional (Material Implication)	Cpq	1	1	0	1
Negation†	Npq	1	0	1	0
Exclusive Disjunction (Material Inequivalence)	Dpq	0	1	1	0
Biconditional (Material Equivalence)	Epq	1	0	0	1

The number of variables (10) was decided upon somewhat arbitrarily as providing reasonable capacity for handling an "interesting" range of problems and at the same time keeping the size of the machine within fairly modest bounds. Changing this number would require only the most obvious changes in the design of the machine.

The six operators were chosen similarly on practical grounds—obviously a single stroke-function operator would have been logically sufficient, but would have seriously restricted the practical application of the machine. The decision to make all operators dyadic, however, is in a somewhat different category. The possibility of changing this condition will be considered in section 3.4, but for the present it will be said merely that the restriction seems advisable for a "Mod I" experimental machine.

†See third paragraph following.

In connection with dyadic operators, Negation deserves special mention. Whereas the other five operators are "naturally" dyadic, Negation is ordinarily monadic. The present definition requires that N take two well-formed formulas as arguments, the first of which is irrelevant—a "dummy". It has been found most efficient (as will be explained in section 3) to insist that

1) the first rather than the second argument be the dummy

2) the dummy be a single variable

3) the variable be one that occurs elsewhere in the formula.

Rigorous discussion of the properties of the language will be reserved for section 3. However, some concepts which are particularly relevant to the operation of the machine—namely those defining the class of formulas which the machine is capable of handling—will be mentioned briefly at this point.

First the formation rules of the language must be given (section 3.1, p. 17). For the present, a formula may be recognized as well-formed on intuitive grounds if it contains no symbols other than those listed above and if it constitutes a translation from a well-formed formula in the more familiar propositional notation ($\vee$, $\cdot$, $\sim$, etc.).

Second, it must be known that evaluation of a given well-formed formula will not exceed the capacity of the machine. This depends, obviously, on the formula's containing a number of symbols (tokens) which can be stored. But it depends, also, on a less obvious structural property called the Rank (section 3.1 Def. 18). For the present, it can be safely stated that any well-formed formula of less than 20 symbol-takens can be handled by the proposed machine†.

Finally, before passing to a detailed consideration of the machine, the general procedure of evaluation will be outlined, since it is relevant to both section 2 and section 3.

The entire operation is sequential. For a formula of n distinct variables, 2^n Major Cycles are required. A Major Cycle consists of, first, making an assignment of a 0 or a 1 to each variable of the formula and, second, reducing this Evaluant (with binary digits properly substituted for the variables) to a single binary digit by successive application of the operators, proceeding from right to left (until the * is reached). If any Evaluant

†This fact is assured by Theorem 3, section 3.3, but does not by any means indicate the complete class of formulas which can be handled by the machine.

reduces to a 0, the formula is a non-tautology; if any evaluant reduces to a 1, the formula is a non-contradiction; and if all 2^n evaluants reduce to 1 (0) the formula is a tautology (contradiction).

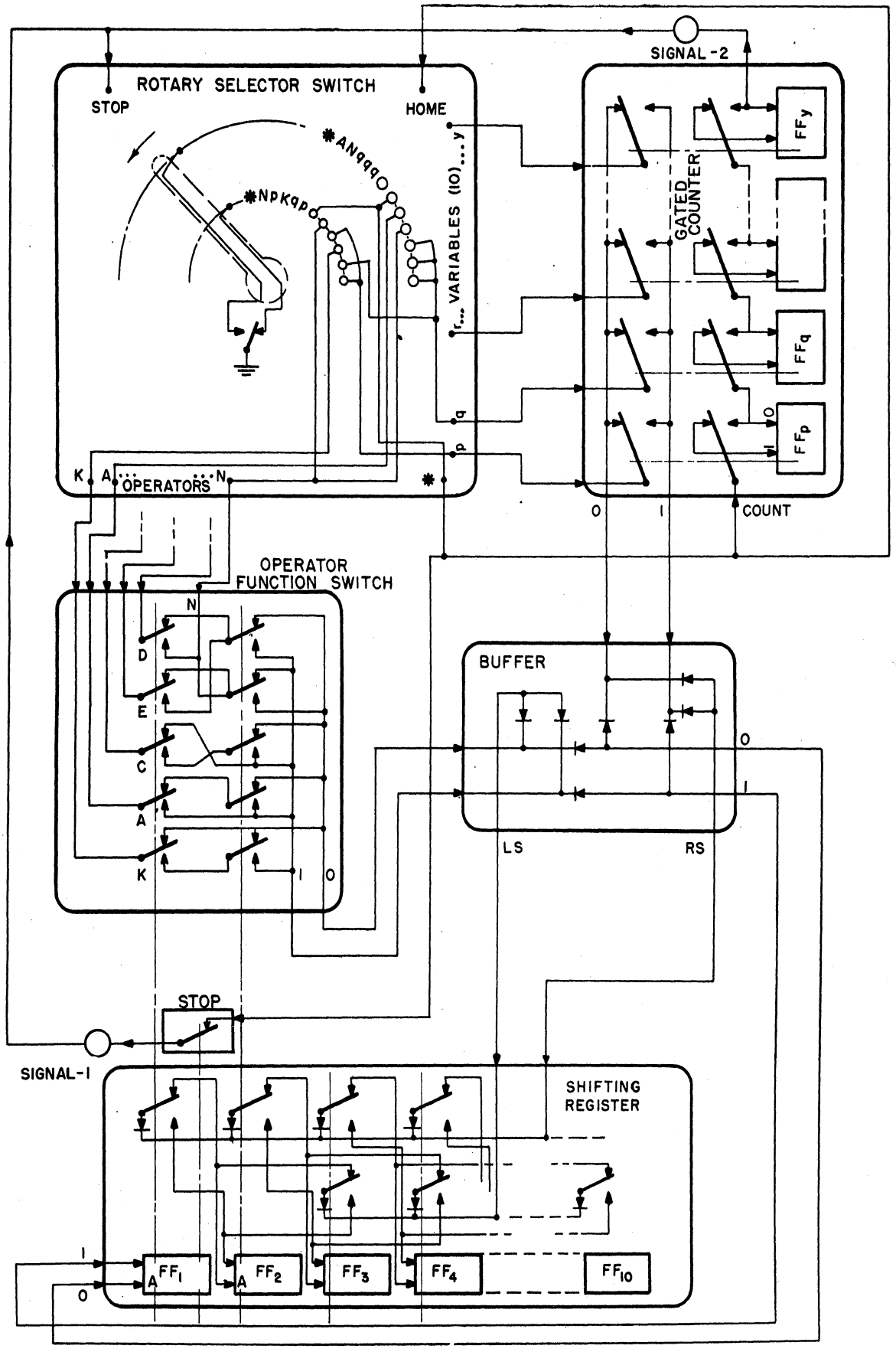


FIG. I
TRUTH-FUNCTION EVALUATOR: SCHEMATIC DIAGRAM

2. A TRUTH FUNCTION EVALUATOR

The operation of the proposed machine will be described in three stages: first (section 2.1) at the block-diagram level, pointing out the purposes of the various components, second (section 2.2) a detailed discussion of the construction and operation of these components, and third (section 2.3) a description of the operation of the machine as a whole. Section 2.4 considers a few simple modifications which would make the machine somewhat more versatile.

2.1 Block-Diagram Analysis

Figure 1 is a schematic representation of the complete machine. For the present, attention is directed primarily to the "block-diagram" aspect of this drawing: the five heavily-outlined boxes (Rotory Selector Switch, Gated Counter, Operator Function Switch, Buffer, Shifting Register) and their interconnections. The detailed discussion of the internal operation of these components, along with such overall problems as timing, is reserved for sections 2.2 and 2.3.

The Rotary Selector Switch stores the original formula and establishes the timing cycle for the entire operation. The formula is set up manually, each of a sequence or arc of contacts being connected to the proper one of the 10 variable, 6 operator, or * outputs. As the wiper passes over these contacts, the outputs are activated, one at a time, in a sequence corresponding to the sequence of symbols in the formula reading from right to left.

The Gated Counter produces Evaluants of the formula. It consists of a 10-stage binary counter. Each stage drives a gate (actually a double gate) which is fed by one of the variable (p through y) outputs of the Selector Switch. Each gate has two outputs, the corresponding outputs of all gates being connected in parallel to produce the 0 , 1 outputs at the bottom of the box. The result is that when one of the variables is encountered by the Selector, the Counter will produce a 0 or a 1 output corresponding to the state of the stage of the Counter associated with that variable. Initially the counter is cleared to zero and is then advanced "1" by each occurrence of the * , thus producing a new Evaluant for each complete "run" of the formula (Major Cycle). When all 2^n Evaluants have been used, i.e., when the Counter returns to 0 , Signal-2 will be operated indicating tautology and the machine stopped†.

†This can occur only if none of the Evaluants reduces to 0 .. Provision for handling formulas with less than 10 distinct variables is discussed in section 2.2.

The Operator Function Switch is fed by the six operator outputs of the Selector and is controlled by the first two stages of the Shifting Register (q.v.)—as indicated in Figure 1 by the center lines—in such a way that whenever an operator is encountered by the Selector, the values of its two arguments occur in the left end of the Register. These three signals are combined in the Operator Function Switch to produce an output which is 0 or 1 according to the truth-table for that operator and its arguments.

The Shifting Register stores the successive portions of the Evaluant as they are produced and facilitates the reduction to a single value. It consists of 10 binary stages with provision for shifting its contents one stage to the right or to the left upon command. Whenever a variable occurs and is assigned a 0 or 1 by the Counter, the Register is shifted right and this 0 or 1 stored in the leftmost stage. When an operator is encountered it is applied to its two arguments from the Register (as in the preceding paragraph), the Register is shifted left to dispose of the arguments just used, and the new value from the Function Switch is stored in the leftmost stage of the Register.

When a star occurs, the state of the leftmost stage of the Register represents the final reduction (to 0 or 1) of the current Evaluant. If this value is 0, the Stop box will operate Signal-1 representing non-tautology and stop the machine.

The purpose of the Buffer is to combine for one purpose and to isolate for another its two pairs of 0, 1 inputs. A signal on either of the input pairs (variable or operator but never both) must be sent to the Register. This is done by the outputs on the right of the diagram. In addition, a left shift of the Register is required for an operator and a right shift for a variable. These are effected by the LS and RS outputs at the bottom of the box.

2.2 Detailed Analysis of Components

The preceding subsection gives an idea of the operation of the machine "in the large". Now the details will be filled in.

The operation of the proposed machine depends importantly on a special type of flip-flop. Such a flip-flop must have the ability to record the occurrence of a 0 or 1 input signal and at the same time remembers its previous state. This situation could be realized in many different ways—some of the most efficient requiring special-purpose equipment. One of the "ground rules" of the present design, however, is to make use of certain available equipment: in this case standard, 8-spring relays. The designing

of the most practical and efficient circuits for the proposed machine is left to the more specialized skills of the engineers and technicians who will be responsible for the actual construction. Nevertheless, in order to be sure that the proposed design was workable, it seemed advisable in several instances to work out some rather detailed circuits. Such instances are to be taken primarily as proof that the proposed design can be carried out with a reasonable amount of equipment. If any of these circuits prove adequate for the actual construction of the machine, so much the better, but it is fully expected that modifications may be necessary.

In the case of the special flip-flop mentioned above, a circuit is presented in figure 2, which accomplishes the desired purpose by the use of two relays: A, the registering relay, and B, the auxiliary relay. Speaking generally, an input signal, throughout its duration will affect only the auxiliary relay, B. When the signal is removed, relay B will drive the registering relay, A, into the same state as B.

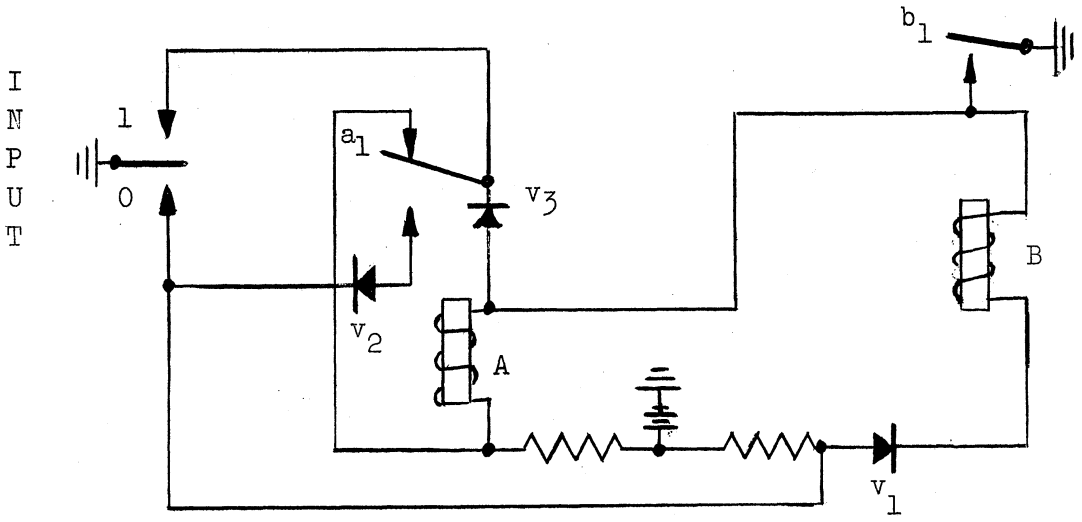
Because of the memory embodied in this flip-flop, its state cannot be determined, in general, from a knowledge of the state of the inputs alone—it is necessary to know the previous state. The various possibilities are presented in Table II. The digit-pairs represent the state of the two relays, e.g., '01' indicates that the A relay is unoperated and the B relay operated. The positions marked 'X' represent conditions that can never occur in normal operation. Note that the states 00 and 11 are "permanent" in the sense that one or the other is produced whenever there is no input. Similarly, the states 01 and 10 might be termed "transient".

TABLE II

Previous state →		00	11	01	10
0	Close	00	10	X	X
	Open	00	X	X	00
1	Close	01	11	X	X
	Open	X	11	11	X

In detail, the operation of the flip-flop is as follows:

Figure 2 shows the flip-flop storing a 0 and with the input circuits open. (The notation 'X', '$\underline{X}$' refers to the upper and lower terminals, respectively, in the orientation of the diagram, of relay coil X.)



Flip-Flop

Figure 2

The "1" Input (from the "0" state)

Closing this input grounds $\bar{B}$ and $\underline{A}$ shunting out A and closing b_1 which locks-up B.

Opening this input removes the ground from $\underline{A}$. $\bar{A}$ remains grounded through b_1 and coil A (protected by the diode or "varistor", V_3) is energized closing a_1. B remains locked-up to V_2 and V_3 (either one alone would be sufficient for this purpose). The state of the flip-flop is now "stable" for storing a 1. Closing the "1" input again has no effect because of V_2.

The "0" Input (from the "1" state)

Closing this input shunts out B at V_1, opening b_1 if V_1 is present. (V_1 might not be required for this purpose if the forward resistance of V_2 and V_3 is small enough, but is necessary in the next step.)

Opening this input removes all ground and a_1 opens. Closing the "0" input again has no effect because of V_1.

The equipment requirements for this circuit are two standard relays with transfer contacts, 3 varistors, and 2 resistors.

In the Schematic Diagram (Figure 1) the flip-flops are represented as small rectangles, labeled 'FF', in the boxes for the Gated Counter and the Shifting Register. Each flip-flop has a "0" and a "1" input. The transfer contacts required on each (the A and the B) relay for internal operation of the flip-flop are not shown. Additional contacts required for external operations are shown and are connected to the proper flip-flop by means of center lines. Except in the case of the Stop box, all contacts are (and must be) controlled by the A relays.

The detailed operation of circuits incorporating the flip-flop are most easily grasped by a careful study of figure 1. Some general remarks, however, may help to establish the proper orientation—particularly with respect to the timing and overall organization of the operation.

A Minor Cycle will be defined as the period between the instant the wiper touches a contact on the Rotary Selector Switch and the instant that it touches the next contact in its direction of travel. This Minor Cycle has two phases: the Contact phase, during which a contact is connected through the wiper to ground, and the Open phase, during which no contact is grounded. The design is such that in each phase at most one relay operation (either an opening or a closing) is required. It is a fundamental requirement of the proposed design that the Rotary Selector can be operated at a speed that will permit this. The evidence—admittedly incomplete—suggests that the Selector could be operated in the continuous or "running" mode and this mode of operation will be assumed. If the assumption proves to be unrealistic, the Switch could be advanced in discrete steps at the cost of a slight increase in equipment but a rather considerable loss of speed.

An intermediate possibility is that, while the Minor Cycle is of sufficient duration to allow two relay operations in sequence, the division into Open and Contact phases leaves the Open phase too short for one relay operation. In this event it might be possible to introduce a "slow-acting" relay—one whose operation time is invariably longer than the operation time for any other relay in the circuit—between the Selector and the flip-flops in order to artifically adjust the phase division. Another less efficient scheme would be to use only every other contact on the arc of the Selector Switch.

To proceed, then, the above indicates that s signal on an input of any box will consist of a grounding of that input, the input remaining grounded throughout the Contact phase of a Minor Cycle. The signal will be preceded and followed by the Open phase in which the input is floating. The destination of such a signal will, in general, be a flip-flop input of the Shifting Register (the major exception being for the * which operates the Counter). From origin to destination the signal may be routed through contact networks and through the Buffer but with negligible loss of time—i.e., there is no sequential operation of relays during the Contact phase (or, for that matter, during the Open phase) of a Minor Cycle. During this phase, the only relays which can change state are the B relays. During the second (Open) phase of this Cycle, the A relays are driven by the B relays, thus setting up the contact networks for the next Minor Cycle.

The specific applications of the flip-flop can now be described. In the Gated Counter of Figure 1, note the two vertical banks of transfer contacts. These contacts are controlled by the A relays of corresponding flip-flops (FF's) as indicated by the center lines. The inner bank effects the counting operation. The binary number recorded by the counter is interpreted as having its least significant digit at the left. The FF inputs are connected to transfer contacts in such a way that a "count" signal will change the state† of every digit to the left of and including the first 0 .

As indicated in Figure 1, the Counter is connected to produce an output which will stop the machine when it has counted to 2^{10} (returned to the all-0 state). This corresponds to producing all evaluants of a formula of 10 distinct variables—the maximum capacity of the machine. To handle efficiently formulas of fewer variables, two alternative modifications are suggested, both assuming that the variables omitted are those nearest the end of the alphabet.

A. Connect the "stop output" directly to the 0 input of the nth FF from the left— n being the number of variables in the formula. This could be done easily as part of the manual set-up.

B. Provide means (e.g., push buttons) for "clearing" the (10-n) rightmost FF's to 1 and leaving the "stop output" unchanged.

The purpose of the outer bank of contacts is to route the variable signal from the Selector to the 0 or 1 output of the Counter in accordance with the state of the FF corresponding to that variable.

†The FF —i.e., the A relay—of course does not change until the "count" signal is removed.

The second application of the flip-flop—the Shifting Register—depends again merely upon the proper interconnection between transfer contacts on the A relays and FF inputs. In the figure, the upper row of contacts controls the right shift—the springs being connected in parallel to the RS input. The make and break contacts controlled by FF_i are then connected, respectively, to the 1 and 0 inputs of FF_{i+1}. The left shift is effected similarly by the lower bank, the connections being in this case from the contacts of FF_i to the inputs of FF_{i-1}. The varistors connected to the transfer contacts are to prevent interaction between the LS and RS circuits. Note that three springs have been eliminated ($FF_{1,2,10}$) since the corresponding shifts are never required.

It was mentioned previously that the Operator Function Switch is controlled by the Shifting Register. This is shown in Figure 1 by the center lines connecting the two vertical columns of transfer contacts in the Switch to FF_1 and FF_2 (A relays) of the Register. When one of the operators, N, D, E, C, A, or K, occurs (no more than one can occur at a time) the signal is fed through the contact network producing an output (0 or 1) which is the result of that operator applied to its arguments (from FF_1 and FF_2).

The purpose of the Buffer has been stated and the internal operation of this simple diode circuit hardly requires further discussion.

2.3 Operation

Given a well-formed formula to be evaluated by the machine, the first step will be to set up the Rotary Selector Switch. In Figure 1, two arcs of contacts are indicated, one set up for the formula, *ANqqq, the other for *NpKqp. The use of two separate arcs in this manner permits the setting up of one formula while the machine is evaluating another. The switch just below the wiper selects the desired arc. Then the wiper is placed in the "home" position: on (or perhaps just ahead of) the first used contact.

Clearing the machine consists merely of opening all relays and could be accomplished simply by removing all sources of power. It is worth noting that under normal operation very little clearing is required. Irrelevant information in the Shifting Register will never be able to move into FF_1 or FF_2 at a time when it could affect the operation; hence the register need never be cleared. The counter will automatically clear to zero upon completing the evaluation of a tautology but in any case when the machine has not completed all evaluants it must be cleared.

Assuming then that the necessary clearing has been done, the next step is to set the maximum to which the Counter is to count—i.e., 2^n for a formula of n variables. This can be done by method A or B of section 2.2 and the operation is ready to start‡.

The first step of the evaluation will be, invariably, the activation (grounding) of one of the variable outputs of the Selector. In the example of Figure 1, with the inner arc of the Selector in use, the p output is grounded. Due to the state of the Counter, its zero output becomes activated; this in turn activates the RS and 0 outputs of the Buffer. Simultaneously, a 0 is recorded in FF_1 (B relay) and the Register shifts right—i.e., puts the B relay of FF_{i+1} in the same state as the A relay of FF_i . This concludes the operations of the contact phase of the first Minor Cycle.

The open phase begins when the wiper of the Selector leaves the first contact. This removes the ground from all FF inputs in the Register thus allowing the A relays to be driven to the state of the B relays and changing, correspondingly, the state of the contact networks both in the Register and in the Function Switch.

The second Minor Cycle repeats the procedure of the first for the next contact of the arc which in the present example corresponds to the variable, q , and is assigned the value 0 . At the conclusion of this Cycle, the values (both 0) for p and q are stored in FF_1 and FF_2 .

The third Cycle begins when the wiper reaches the third contact which, in this case, is connected to the K operator output of the Selector. By tracing the "K" path through the Function Switch it is seen that the 0 output is activated as is required since the value of K00 is 0 . Passing through the Buffer it is seen that its 0 and LS outputs are activated thus storing a 0 in FF_1 and simultaneously shifting the Register, this time, to the left.

The next two Cycles dispose of p and N , leaving in FF_1 the value NOK00 ≡ 1 . The next Cycle, which is the last Minor Cycle of the first Major Cycle, grounds the * output. This signal is transmitted simultaneously (1) to the Stop box, where it encounters an open circuit (since FF_1 is storing a 1), (2) to the counter which it advances one step—i.e., changes the state of FF_p from 0 to 1, and (3) to the "Home"

‡Here, as elsewhere, the simplest alternative is assumed: that all power can be applied and the Selector set in operation simultaneously. It is realized, however, that more selective control might be useful or possibly necessary and it is left to the engineers to make such modifications as seem advisable.

input of the Selector which will return the wiper to its initial position† preparatory to the next Major Cycle.

The remaining three Major Cycles proceed quite analogously until the final Minor Cycle of the problem—that for * . At this point FF_1 holds the value of N1K11 which is 0 . Hence the * signal can pass through the Stop box, turn on Signal-1 indicating non-tautology and stop the machine. Given the proper timing arrangement, the * signal could also effect a final "count" and "home" operation which might or might not be of some slight advantage.

The computation time for a formula of length (number of character-tokens), L , containing n distinct variables, on a machine whose Minor Cycle time is t , is given by the formula,

$$T = tL2^n \quad .$$

This is the time required to test all evaluants. Of course in many cases it will not be necessary to test them all. Assuming that t = 1/30 second (30 steps per second of the Rotary Selector Switch under the mode of operation described above) the computation time would be, for example, for a "large" formula with n = 10 , L = 25 , just under 15 minutes; for a "small" formula with n = 5 , L = 10 , T would be slightly over 10 seconds.

2.4 Modifications

Depending upon the use to which the machine will be put, it might be advantageous to provide certain facilities not included in the basic design. Naturally, only a few of the many possible contingencies can be provided for the four suggestions below are offered as examples.

1) Over-length Formulas. If the number of character-tokens in a formula is greater than the number of contacts on an arc of the Selector, it could be set up on two or more arcs and a simple relay circuit added to "home" and to effect automatic switching of wiper contacts from one arc to the next. The formula for running time, $T = tL2^n$ applies to this case without change.

2) Tests other than for Tautology. If the connections in the Stop box are reversed—closing the circuit through the "make" rather than the "break" contact—then Signal-1 will be operated and the machine stopped upon the

†If a "non-homing" type of switch is used, certain modifications must be made at this point.

occurrence of a "1" or "true" evaluant indicating that the formula is not a contradiction.

It might be required to determine the values of the variables for which the formula is true or for which it is false. For this purpose neon lights might be attached to the FF's of the Counter for easy reading. The states of these lights would be recorded, manually, each time the machine stops. The "non-contradiction stop" of the preceding paragraph would be used to advantage in this test on formulas known to have more false than true values.

3) Formulas of more than Ten Variables. If it seemed advisable to use the Evaluator at all frequently for such formulas, it would of course be a simple matter to add the necessary number of FF's to the Counter. However, given any fixed capacity, the following slight modification makes it possible to "program" the handling of formulas with an excess number of variables. Provide two additional Selector Switch outputs, connected directly to the 0 and 1 outputs of the Counter. This has the effect of adding the constants 0 and 1 to the formula language. The assignments of 0 and 1 to the "excess variables" can then be made manually. This procedure requires 2^{n-10} complete "runs" to test all possibilities where $n-10$ represents the number of excess variables†. The standard formula for running time gives a good approximation for this case, but a more accurate time would be given by the formula $T = tL2^n + t_m 2^{n-10}$ when t_m is the time required for manually changing the connections to the 0 , 1 outputs and 2^{n-10} is the number of possible combinations of the excess variables—i.e., the number of times the connections must be made.

4) Formulas of Excess Rank. Though the formal definition of rank is reserved for section 3, it can be informally characterized for present purposes as that property of a formula which determines the capacity required in the Shifting Register. A Formula of "excess rank", therefore, is one whose evaluation would exceed the capacity of Register. Although it has been assumed that such formulas will not be allowed on the machine, it might be desirable for the machine to detect excess rank, if only for the purpose of error prevention. This facility can be provided as follows:

a) Clear the Register before each problem.

b) Provide a signal to indicate a right shift of a 1 from the rightmost FF (FF_{10}) .

c) Provide for a 0 to be shifted into FF_{10} by every left shift.

†Certain special types of formulas are adaptable to less time-consuming variations of the above procedure and, in fact, it seems probable that the general procedure might be improved upon; but the possibilities have not been seriously investigated.

Though the fact appears to be of little practical value, it is worth noting that this modification actually increases the "rank capacity" of the machine for certain rare formulas, due to the effect of shifting 0's off the right end of the register and then shifting them back in again.

This concludes the description of the machine, per se. The remainder of the report is concerned primarily with the theory underlying the method evaluation.

3. THEORY OF TRUTH FUNCTION EVALUATION

The present section serves two purposes: to provide a more rigorous basis for the design of the machine in section 2, and to express the theory behind this machine in more generalized form so that it may be applied to a wider variety of problems in truth function evaluation.

3.1 Formal Languages.

The symbolism used throughout the remainder of the report will be as follows:

θ^n (n an integer) will range over operators of degree n .

Γ , Δ will range over formulas—usually well-formed formulas.

p , q ,... will range over propositional variables.

α , β ,... will range over the two truth-constants, 0 and 1 .

(Any of these symbols may have subscripts or superscripts.)

All of the above symbols are in the meta-language. In addition, the language of section 1 will be used, frequently, for examples. Note that the use of 'p' , 'q' ,... constitutes an ambiquity between the object language and the meta-language; this ambiguity will be resolved by the context in all cases, however.

Several languages will now be defined: Def. 1A giving the set of symbols for each, Def. 1B giving the formation rules which are common to all.

Def. 1A.

S contains a finite number of operators, which may be of different degrees, and a finite number of propositional variables.

$\bar{S}$ contains the same operators as S , the two truth constants, 0 and 1 , but no propositional variables.

S^n ($\bar{S}^n$) contains the same symbols as S ($\bar{S}$) except that all operators are of degree n —i.e., "n-adic".

Def. 1B. Any propositional variable or truth constant is a Well-formed Formula (henceforth, wff) with Rank, $R = 1$. If $\Delta_1, \ldots, \Delta_n$ are wf (Well-formed) , then $\theta^n \Delta_n \ldots \Delta_1$‡ is wf with

$$R = \operatorname{Max}_{i=1}^{n} (R(\Delta_1) + i - 1) .$$

The Length, $L(\Delta)$ of a wff , Δ , is the number of characters (tokens of operators, variables, truth constants) in Δ .

Note the assymetrical role played by the various arguments of an operator in the determination of the rank of the operator-plus-its-arguments. This can be taken advantage of in minimizing the rank of formulas under various circumstances. E.g., suppose we need the alternation of p and Kqr . ApKqr has a rank of 2 , AKqrp has a rank of 3 , while the lengths of both formulas are the same.

A further example concerns the conditional operator, C . It is often the case that the antecedent of a conditional has a greater rank than the consequent; this is especially true with the hypothetical corresponding to a given argument. Suppose $R(\Gamma) > R(\Delta) > 1$. Then $R(C\Gamma\Delta) = R(\Gamma) + 1$ and $L(C\Gamma\Delta) = L(\Gamma) + (L(\Delta) + 1$. The rank could be reduced by 1 if C were replaced by a "reversed conditional" operator, C_r : $C_r\Delta\Gamma =$ def. $C\Gamma\Delta$. Then $R(C_r\Delta\Gamma) = R(\Gamma)$. However, the same saving in rank can be obtained without the introduction of a new operator by using the equivalent expression, $A\Delta N p\Gamma$, whose rank is also $R(\Gamma)$ although its length is $L(\Gamma) + L(\Delta) + 3$, an increase of 2 .

It is not obvious that the decomposition of a wff into its arguments is unique; i.e., that, given a wff Δ of the form $\theta^J \Delta_J \ldots \Delta_1$, Δ cannot also be expressed as $\theta^J \Delta'_J \ldots \Delta'_1$ unless it is true that all $\Delta_i = \Delta'_i$. Since the uniqueness of Rank and of truth values depend upon this uniqueness of decomposition it will now be formally established.

Lemma. Given a wff , $\Delta = \theta^J \ldots$, and a wf initial segment of Δ , $\Gamma = \theta^J \ldots$ (having its sequence of m characters identical to the sequence of the first m characters of Δ) , then $\Delta = \Gamma$.

Proof (by induction on $L(\Delta)$) . The Lemma is true, obviously, for $L(\Delta) = 1$. Assume that it is true for all Δ having $L(\Delta) \leq n$; then it can be shown to hold for $L(\Delta) = n + 1$ by the following: By the formation rules, Δ can be written as $\theta^J \Delta_J \ldots \Delta_1$, and Γ as $\theta^J \Gamma_J \ldots \Gamma_1$ with $L(\Delta_j) \leq n$ and $L(\Gamma_j) \leq n$. Therefore, since either Δ_J is an

‡The order of the Δ_i is significant only in relation to the Rank formula which follows. For an informal discussion of Rank see section 2.4, modification 4.

initial segment of $\ulcorner_J$ or vice versa, the two formulas satisfy the induction hypothesis and $\Delta_J = \ulcorner_J$. Suppose it has been shown that $\Delta_{j'} = \ulcorner_{j'}$ for all j' satisfying $j > j' \leq J$. Then the argument above (for subscript J) applies and it follows that for all j such that $1 \leq j \leq J$, $\Delta_j = \ulcorner_j$; and therefore $\Delta = \ulcorner$. Q.E.D.

Theorem 0. The decomposition of a wff into an operator followed by a sequence of wff's is unique; i.e., if $\Delta = \theta^K \Delta_J \ldots \Delta_1 = \theta^K \Delta'_K \ldots \Delta'_1$ ‡, then $K = J$ and $\Delta_i = \Delta'_i$ for $1 \leq i \leq K$.

Proof. Each of the pairs of formulas (Δ_J , Δ'_K) , $(\Delta_{J-1} , \Delta'_{K-1})$,... in turn satisfies the hypothesis of the Lemma and consequently the conclusion. Continuation of the process until Δ is exhausted will establish simultaneously the identity of the paired formulas and the relation $J = K$. Q.E.D.

On the strength of Theorem 0 , it can be assumed henceforth, that the Rank of any wff is unique, and that the truth value of a wff in $\bar{S}$ (or $\bar{S}^n$) is independent of the order of evaluation.

3.2 Concepts and Theorems Concerning the Process of Evaluation.

Let us call any wff , Δ, of $\bar{S}$ an evaluant. Δ is, of course, an evaluant of some wff , $\ulcorner$ of S , i.e., Δ is obtainable from some $\ulcorner$ of S by substituting 0 for all occurrences of some (possibly none) of the variables of $\ulcorner$ and substituting 1 for all occurrences of the remaining variables. Hereafter, section 3.2 will be concerned with the language, $\bar{S}$, exclusively.

The Length of the Uninterrupted Sequence of bits (occurrences of truth constants(counting from the right of an evaluant Δ is called Lus (Δ) ; e.g., Lus $(\theta^2_1 \alpha \; \theta^2_2 \beta\alpha) = 2$, Lus $(\alpha) = 1$.

Theorem 1. Lus (Δ) ≥ the degree of the rightmost operator of Δ , or 1 if there is no operator in Δ . (I.e., the rightmost operator θ^n of an evaluant Δ is followed by at least n bits.)

Proof. Else Δ would not be wf . Q.E.D.

($\ulcorner$ is the immediate reductum of an evaluant, Δ) = def. ($\ulcorner$ is formed from Δ by substituting for the rightmost operator θ^n of Δ and the following n bits its value). E.g., K01 is the immediate

‡The '=' means that the sequences of symbols are identical.

reductum of KOA01 . Note that the immediate reductum of the evaluant is itself an evaluant.

(Δ_1 ,..., Δ_M is the complete sequence of reducta of an evaluant Δ_1) = def. (Δ_{i+1} is the immediate reductum of Δ_i , and Δ_M is 0 or 1 .) E.g., KOA01 , K01 , 0 is the complete sequence of reducta of KOA01.

The main theorem on evaluation is:

Theorem 2. If Δ_1 ,..., Δ_M is the complete sequence of reducta of the evaluant $\Delta(\Delta = \Delta_1)$, then

$$\operatorname{Max}_{m=1}^{M} (\text{Lus } (\Delta_m)) = R(\Delta).$$

Proof. (by induction on $L(\Delta)$).
(I) It is obvious that it holds for $L(\Delta) = 1$.
(II) Assume it holds for all wff of $L \leq n$. It will be proved that it holds for any formula Δ of $L = n + 1$. Note that Δ is of the form $\theta^J \Gamma_J \cdots \Gamma_1$, where $L(\Gamma_j) \leq n$ $(1 \leq j \leq J)$, and each Γ_j is an evaluant.

Let the complete sequence of reducta of Δ be Δ_1 ,..., Δ_m , ...Δ_M (as before) and the complete sequence of reducta of Γ_j be Γ_j^1 ,..., Γ_j^p ,..., $\Gamma_j^{P_j}$ (where Γ_j^1 is Γ_j) . Then Δ_m $(m = p + \sum_{i=1}^{j-1} P_i)$ is of the form

$$\theta^J \Gamma_J \cdots \Gamma_{j+1} \Gamma_j^p \alpha_{j-1} \cdots \alpha_1 ,$$

where $\operatorname{Max}_{p=1}^{P_j} (\text{Lus } (\Gamma_j^p)) = R(\Gamma_j)$ by the inductive hypothesis. It follows that

$$\operatorname{Max}_{p=1}^{P_j} (\text{Lus } (\theta^J \Gamma_J \cdots \Gamma_{j+1} \Gamma_j^p \alpha_{j-1} \cdots \alpha_1)) = R(\Gamma_j) + j - 1$$

since there are $j - 1$ α's in the sequence α_{j-1} ,..., α_1 .
Hence,

$$\operatorname{Max}_{m=1}^{M} (\text{Lus } (\Delta_m)) = \operatorname{Max}_{j=1}^{J} (R(\Gamma_j) + j - 1) .$$

But, by definition,

$$R(\Delta) = \operatorname{Max}_{j=1}^{J} (R(\Gamma_j) + j - 1) .$$

Therefore,

$$\max_{m=1}^{M} \left(\text{Lus} \left(\Delta_m \right) \right) = R(\Delta) \quad . \quad \text{Q.E.D.}$$

3.3 Theorems Concerning Languages with only Dyadic Operators.

With the machine of section 2 specifically in mind, the present subsection will establish some theorems concerning the language, S^2 , in which all operators are dyadic. Theorems 3 and 4 establish connections between $L(\Delta)$ and $R(\Delta)$.

Theorem 3. If Δ is in S^2 then $L(\Delta) \geq 2R(\Delta) - 1$.

Proof. (by induction on $L(\Delta)$.
(I) The theorem clearly holds for $L(\Delta) = 1$.
(II) Assume it holds for all wff of $L \leq n$. It will be proved that it holds for any formula Δ of $L = n + 1$. Note that Δ is of the form $\theta^2 \Delta_2 \Delta_1$ where $L(\Delta_1)$, $L(\Delta_2) < n$.‡ Obviously, $L(\Delta) = L(\Delta_2) + L(\Delta_1) + 1$. By the inductive hypothesis $L(\Delta_2) \geq 2R(\Delta_2) - 1$ and $L(\Delta_1) \geq 2R(\Delta_1) - 1$. Hence,

$$L(\Delta) \geq 2R(\Delta_2) + 2R(\Delta_1) - 1 \quad .$$

Depending upon the relative Ranks of Δ_1 , and Δ_2 , there are two cases to consider:

(A)	(B)
$R(\Delta_2) + 1 \geq R(\Delta_1)$	$R(\Delta_2) + 1 < R(\Delta_1)$.
Then $R(\Delta) = R(\Delta_2) + 1$	Then $R(\Delta) = R(\Delta_1)$
and $L(\Delta) \geq 2R(\Delta) + 2R(\Delta_1) - 3$.	and $L(\Delta) \geq 2R(\Delta_2) + 2R(\Delta) - 1$.
But $R(\Delta_1) \geq 1$	But $R(\Delta_2) \geq 1$
so $L(\Delta) \geq 2R(\Delta) - 1$.	so $L(\Delta) \geq 2R(\Delta) - 1$.

Hence, $L(\Delta) \geq 2R(\Delta) - 1$. Q.E.D.

Remark: For each rank there exists a formula Δ of that rank such that $L(\Delta) = 2R(\Delta) - 1$, i.e., a minimum length

‡Note that for even values of L no formulas exist in S^2 and the Theorem holds vacuously.

formula of that rank. Such a formula can be obtained by starting with an α and simultaneously prefixing a θ^2 and suffixing an α as many times as required, e.g., $\theta^2\theta^2\theta^2\,\alpha\alpha\alpha$ is of rank 4 and length 7 .

Theorem 4. For any $R \geq 2$ there is no upper limit to the length (L) of formulas of S^2 of this rank‡.

Proof. Consider any formula Δ of $R \geq 2$. An indefinitely long formula of the same rank may be constructed from it by repeatedly preceding it by $\theta^2 p$. Q.E.D.

Note that Theorem 3, the remark following it, and Theorem 4 establish the following concerning the formula of any given rank ≥ 2 : there is a lower bound to the length of these formulas but no upper bound. In the absence of an upper bound it would be of interest (in determining the relative capacity of the memory and the shifting register) to have more information concerning the relation between the length and rank of the formulas that would actually be tested on such a machine as has been proposed. Note in this connection that for all formulas Δ with the property that for every operator the arguments of that operator are of equal length (e.g., $\theta^2\theta^2\theta^2\,\alpha\alpha\theta^2\,\alpha\alpha\theta^2\theta^2\,\alpha\alpha\theta^2\,\alpha\alpha$) it is the case that

$$L(\Delta) = 2^{R(\Delta)} - 1 .$$

For any formula $\ulcorner$ let $\ulcorner'$ denote the formula derived from $\ulcorner$ by replacing each occurrence of N^1 (if any) by $N^2 p$ (where p is a dummy argument)‡‡.

Theorem 5. Let Δ be a wff in S . Then $R(\Delta) \leq R(\Delta') \leq R(\Delta) + 1$.

Proof. (by induction on $L(\Delta)$) :
(I) The theorem holds for $L(\Delta) = 1$ (trivially).
(II) Assume it holds for all wff , Δ , such that $L(\Delta) \leq n$. It will be proved that it holds for any formula, Δ , such that $L(\Delta) = n + 1$. This proof involves two cases according to whether or not the first operator of Δ is N^1 :

Case 1. The first operator in Δ is not N^1 ; Δ is of the form $\theta^J \Delta_J \ldots \Delta_1$.

‡This theorem applies, more generally to any language, S .
‡‡Note that the "prime" symbol in the expression $\ulcorner'$ here denotes an operation on $\ulcorner$.

1. Then, $\Delta' = \theta^J \Delta'_J \ldots \Delta'_1$

2. By the inductive hypothesis: $R(\Delta_j) \leq R(\Delta'_j) \leq R(\Delta_j) + 1$
J

3. But, by definition, $R(\Delta') = \underset{j=1}{\overset{J}{\text{Max}}} (R(\Delta'_j) + j - 1)$

4. Therefore,

$$\underset{j=1}{\overset{J}{\text{Max}}} (R(\Delta_j) + j - 1) \leq R(\Delta') \leq \underset{j=1}{\overset{J}{\text{Max}}} (R(\Delta_j) + j - 1 + 1)$$

5. $R(\Delta) \leq R(\Delta') \leq R(\Delta) + 1$.

Case 2. The first operator in $\Delta = N^1$; Δ is of the form $\Delta = N^1 \Delta_1$.

1. By definition of the substitution operation, $\Delta' = N^2 p \Delta'_1$

2. $R(\Delta_1) \leq R(\Delta'_1) \leq R(\Delta_1) + 1$ by hypothesis.

3. Two sub-cases must be considered according to whether $R(\Delta_1) \geq 2$ or $R(\Delta_1) = 1$.

(a) Suppose $R(\Delta_1) \geq 2$, then $R(\Delta') = R(\Delta'_1)$.
Now, since $R(\Delta) = R(\Delta_1)$, Step 2 yields
$R(\Delta) \leq R(\Delta') \leq R(\Delta) + 1$.

(b) Suppose $R(\Delta_1) = 1$, then $R(\Delta) = 1$,
$R(\Delta') = 2$, satisfying the theorem. Q.E.D.

Note that any of the alternative ways of constructing the N^2 operator would, in general, increase both R and L . If the second rather than the first argument is the dummy, there is no upper limit to $R(\ulcorner\ ) - R(\Delta)$ as shown by

$$\ldots N^2 N^2 N^2 p q_1 q_2 q_3 \ldots \quad (= \ulcorner)$$

$$\ldots N^1 N^1 N^1 x \quad (= \Delta)$$

Hence $L(\ulcorner\ ) - L(\Delta)$ is equal to the number of occurrences of N^2 in $\ulcorner$ (or of N^1 in Δ) .

If neither argument is a dummy, then in order to satisfy $N^1 \equiv N^2 \Delta \ulcorner$, it is necessary that Δ be a function of $\ulcorner$ or a constant

formula (tautology or contradiction)†. In either case extra computing time and perhaps extra equipment would be required. Using identity — $N^1 \neg \equiv N^2 \neg\neg$ — which is probably as simple a function as any, each occurrence of negation has its Rank increased by 1 and its Length increased by L - 1 as compared to the N^2 proposed in this report††. Using for Δ a simple constant formula, the contradiction Dpp , a negation has its L increased by 2 (as compared to the standard N^2) ; its Rank is not increased except for the case of $R(\neg) \leq 2$.

†If the symbols for 0 and 1 are included in the formula language (as suggested in section 2.4) the use of one of these constants as dummy, rather than a constant formula, would be a satisfactory alternative.

††This is true for all instances except L = 3 — the shortest possible expression.

4. CONCLUSION

In conclusion a few remarks will be made concerning the use of the proposed method of evaluating truth-functions in connection with formulas of great length and many variables. This method, which was illustrated in section 2, may be summarized as follows: For each Evaluant of the given formula, the formula is scanned from right to left, each variable being replaced by its value in the Evaluant and each operator together with its arguments being replaced by its functional value until a single truth-value is obtained. One of the advantages this method has is that the capacity of a machine embodying it may be increased with regard to the length, rank, or number of variables independently, merely by enlarging the memory, the shifing register, or the counter respectively.

The speed of operation required for extremely long formulas may be shown by reference to an example. Consider a formula containing 25 distinct variables and 250 characters. There are 2^{25} Evaluants of this formula, and if these are all reduced the machine must scan 250×2^{25} or roughly 10^{10} characters. For this to be feasible the time per character must be of the order of microseconds; for example, 250×2^{25} microseconds equals approximately 140 minutes.

It is clear from the preceding sections that a serial, circulating memory is naturally adapted to the present method of evaluating truth-functions. Mercury delay lines would be especially suitable; a magnetic drum would be satisfactory but somewhat slower. A brief "design-sketch" will be presented for an evaluator using mercury delay lines of 1000 bit capacity and 1 microsecond pulse period. The design of an evaluator employing a magnetic drum would be very similar. We assume that the formula will be read into the machine from a magnetic or paper tape so that set-up time will be small.

Consider a language containing the six dyadic operators K , A , C , E , D , N , the marker symbol * , and 25 distinct propositional variables. Express these 32 characters in a five-bit code and store them in parallel in a bank of five delay lines. Store the formula to be evaluated in this memory as many times as it will go, placing a * after each occurrence of it and filling every unused memory position with a * . There will be a certain waste of memory capacity (and hence of time) whenever the formula is not an integral factor of 1000 characters in length; e.g., a formula of 251 characters (including a *) would be stored three times with a waste factor of 247/1000 or about 25%. This waste factor could be decreased by the use of 2 or more banks of delay lines and some switching equipment.

The above-described memory would feed a decoding function switch with five inputs and 32 outputs. These outputs would in turn feed a gated counter, operator function switch, buffer, shifting register, and stop box,

all of which would be the electronic analogues of the circuits shown in Figure 1. Certain minor variations in the design are necessary, e.g., one due to the fact that stars are used to fill otherwise empty memory positions. It is possible, using available electronic techniques, to construct these circuits so that they would keep up with the megacycle pulse rate of the circulating memory. Such a machine would completely evaluate a formula containing 25 variables and slightly over 250 characters in length in about three hours.

BIBLIOGRAPHY

1. Edmund C. Berkeley, Giant Brains, John Wiley and Sons, Inc., New York, 1949, Chapt. 9.

2. D. M. McCallum and J. B. Smith, "Mechanized Reasoning," Electronic Engineering, April, 1951.

3. D. M. McCullum and J. B. Smith, "Feedback Logical Computers," Electronic Engineering, December, 1951.

4. W. R. Abbott, "Computing Logical Truth with the California Digital Computer," Mathematical Tables and Other Aids to Computation, 5(1951) 120-128.