$$\Delta y = c_1(\Delta y'),\ \Delta t = c_2(\Delta t') \tag{117}$$

and if equation 116 holds, the numerical solutions likewise will be the same at corresponding mesh points. Thus, for a finite plate, the thickness should be broken up into the same number m of space intervals, and the period into the same number n of time intervals, to obtain this numerical agreement. The value of these conclusions in saving expensive calculations and tests is manifest.

On the other hand, if for a case where the finite-difference solution could be made identical, with Δt and Δy unchanged, the numerical solutions need not agree in the manner described. They will differ, however, in the same manner as the solutions for the same physical problem, but with different Δy and Δt.

In only one case does Table I show results for two runs having constant f/b^2, H_m, and the same numbers m and n. These are the 5th and 13th entries, and they do show the correct 1:2 loss ratio, namely 731:1,462 watts per meter^2. Had these dimensional considerations occurred to us earlier, it would have saved calculating the 13th entry altogether.

The dimensional analysis can even be extended to include stretching modifications in the shape of the B-H curve, by replacing equation 1 with

$$H' = f(B') \tag{118}$$

where

$$H = c_4 H',\ B = c_5 B' \tag{119}$$

and where c_4 and c_5 are constants. Substitution into equation 6 shows that, in the primed variables, the form of the equation is preserved provided that

$$\frac{c_3 c_4 c_2}{c_1^2 c_5} = 1 \tag{120}$$

While solutions that have not been carried to complete convergence are always open to suspicion, we believe that the calculated power loss ratios shown in Table I reflect not so much lack of convergence as discrepancies resulting from the degree of approximation inherent in the different mesh sizes employed. This opinion is based upon extensive comparison of the finite-difference approximation solutions with the exact solutions in the linear (constant μ) case.

Admittedly, the initial computer programming, "debugging," and numerical analysis experimentation were time-consuming, and millions of calculations were involved for each case considered. Yet, actual production runs were relatively short and inexpensive. Average computer time on the IBM *7090* was about 1 minute per run.

To be sure, the expense of numerical computing machine procedures cannot be matched against "limiting nonlinear" theory for high saturation, nor are they intended to replace experimental tests and empirical formulas summarizing the results of such tests. Yet we believe the numerical solution of Maxwell's nonlinear equations, with the aid of computing machines, will prove increasingly useful in the future. It is potentially more accurate than the limiting nonlinear theory, especially when the saturation is not so high as to justify the replacement of the curve in equation 1 by two horizontal straight lines; it is applicable to a variety of boundary conditions corresponding to equations 10 through 17, and for different types of B-H curves.

The main reason for including the comparison of the computed solutions with test results was to establish confidence in the numerical method. It can now be applied to other, more difficult, cases for which empirical equations and approximate theories are not available.

An Approach to Artificial Intelligence

R. W. HOUSE
MEMBER IEEE

T. RADO

Summary: In this paper, attention is called to certain problems which are extremely primitive conceptually, and yet exhibit certain features quite relevant for study in the field of artificial intelligence. These issues are presented with the belief that realistic study of them can provide approaches to artificial intelligence. Turing machines are used to present the problems. By this means, the concept of optimality is introduced. The paper then addresses the problem of discovering optimal machines using programmable methods. The requirement for proofs of methods is emphasized. Apparent ways of limiting the difficulties encountered are shown to be deceptive. Finally, the effect achieved by changing the required format is illustrated.

Paper **S-142**, recommended by the AIEE Computing Devices Committee and the Transmission and Distribution Committee and approved by the AIEE Technical Operations Department for presentation at the IEEE Winter General Meeting, New York, N. Y., January 27–February 1, 1963. Manuscript submitted October 26, 1962; made available for printing February 25, 1963.

R. W. HOUSE is with Battelle Memorial Institute, and T. RADO is with The Ohio State University and Battelle Memorial Institute, both in Columbus, Ohio.

Some of the work referred to herein was supported in part by Battelle Memorial Institute and by the United States Army Research Office (Durham), under Grant *DA-ARO(D)-31-124-G53* with The Ohio State University Research Foundation.

THE COMMENTS presented in this paper are intended to assist in clarifying basic concepts and in selecting realistic objectives in the field of artificial intelligence. In this field, many research objectives outlined by various workers are viewed with considerable pessimism by a significant number of other workers. Explicitly stated reasons for this pessimism range from apparent conflicts with established facts in mathematical logic on the one hand to the apparent impossibility of implementation in terms of actual computers on the other hand. The present writers feel that much of the disagreement stems from vagueness of terminology as regards both basic concepts and basic objectives.

In this paper, attention is called to certain problems which are extremely primitive conceptually and yet exhibit certain features quite relevant for study in the field of artificial intelligence. The authors' motivation in presenting these issues is their belief that the realistic study of artificial intelligence may be expected to yield novel methods and insights which should be of decisive importance in many fields.

Turing Machines

For conciseness, definiteness, and conceptual simplicity, we shall use Turing machines with the alphabet 0, 1 to illustrate various fundamental points. An excellent presentation of Turing machines is available in Kleene's book.[1] Hence, we restrict ourselves here to indicate some notational conventions we shall use for convenience. Fig. 1 illustrates a Turing machine with ALPHABET 0, 1. The symbols C_1, C_2, etc., mean card 1, card 2, etc. We use the term "card" rather than the usual terms *internal configuration* or *internal state* since the latter terms may discourage persons unfamiliar with the insides of computers.

On each card, the left-most column contains the ALPHABET 0, 1. The next column to the right contains the OVERPRINT BY instruction; the next column to the right contains the SHIFT instruction, where 0 is the code for a LEFT SHIFT and 1 is the code for a RIGHT SHIFT. The right-most column contains the subscript of the card that is called to assume control after the shift, with the understanding that a 0 (zero) in this column is the code for the STOP instruction. Note that we do not allow a CENTER SHIFT; thus, the machine must shift right or left after the execution of an OVERPRINT BY instruction. This deviation from Kleene's terminology is quite convenient in many situations. We assume the reader will work with the ma-

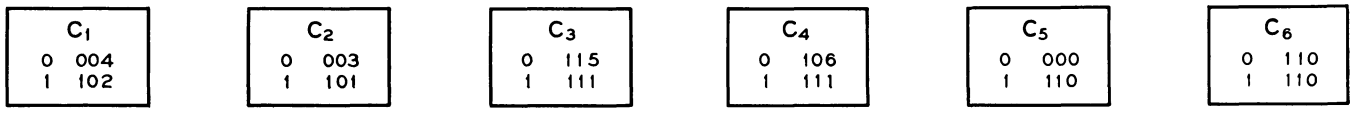


Fig. 1 (above). A 6-card machine

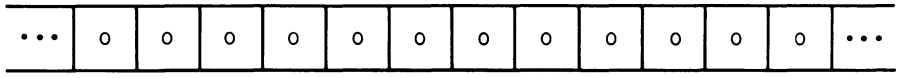


Fig. 2 (right). All-zero tape

chines presented until he understands their operation.

In the sequel, machine means a Turing machine with the 0, 1 ALPHABET, given in terms of any finite number of cards in the format just described. Thus, Fig. 1 shows a particular 6-card machine. A machine operates on a potentially both-ways infinite tape divided into squares, each of which contains a 0 (zero) initially (all-zero tape), as shown in Fig. 2.

INPUT

An input for a machine is obtained by indicating where the machine should start and by replacing the 0's by 1's in any finite number of squares. In particular, the number of these squares may be equal to zero; the input is then the all-zero tape. Furthermore, the 1 squares need not be consecutive. Fig. 3 shows two possible inputs containing at least one 1.

The three dots at both ends in Fig. 3 indicate that all the squares to the right and left contain 0's, while the arrow indicates the starting position. The machine always starts with its card 1 in control. Fig. 4 shows the operating record of the 6-card machine in Fig. 1 for the input on the right side of Fig. 3. The number and subscript below the square indicate the card number and the content of the square before the overprint action effected by the card, respectively.

OUTPUT

In the case illustrated in Fig. 4, the machine of Fig. 1 stops after five shifts, and the output consists of the input and two additional 1's as shown, the machine stopping in the position indicated by the 0 for STOP in the part of the figure below the tape. It is instructive to see what happens if the machine of Fig. 1 receives the input of Fig. 5. The operating record for this case is shown in Fig. 6.

Now the machine of Fig 1 *never stops;* after the third shift, it oscillates back and forth on two adjacent squares. The expression for this is that it "goes into a loop."

A machine is actually a plan of operation, spelled out in a standardized form. If the machine and the input including the starting position are given, then the sequence of events—the operating record—is completely determined. An interesting exercise is to discover the number of C-card machines for each positive integer C. If we denote by Nl the number of C-card machines, then it is easy to see that

$$N_c = [4(C+1)]^{2C} \tag{1}$$

For example:

$$N_1 = 8^2,\ N_2 = 12^4,\ N_3 = 16^6,\ N_4 = 20^8,$$
$$N_5 = 24^{10},\ N_6 = 28^{12} \tag{2}$$

Evidently, N_c grows very fast as C increases. But even for small card-numbers, N_c is quite large. For instance

$$N_3 = 16^6 = 16{,}777{,}216 \tag{3}$$

$$N_4 = 20^8 = 25{,}600{,}000{,}000 \tag{4}$$

$$N_5 = 24^{10} = 63{,}403{,}380{,}965{,}376 \tag{5}$$

Thus we see that N_C is of astronomical size, even for low values of C.

A Primitive Problem

Consider the positive integers, 1, 2, 3, 4, 5, 6, etc. The integers 1, 3, 5, etc., are ODD; 2, 4, 6, etc., are EVEN. Human beings of normal intelligence learn easily to identify any given positive integer as ODD or EVEN, or, as we shall say, to determine PARITY. Let us now return, to achieve conceptual simplicity, to the stone age in a sense. According to ample archeological evidence, the caveman kept business records in the form of notches carved on sticks. To avoid using heavy equipment, let us consider a record, or a count, in the form of a finite sequence of consecutive 1's on a tape as explained in the preceding section. For conciseness, we shall call an input consisting of a finite or nonzero number of consecutive 1's an INPUT STRING. Fig. 7 shows two INPUT STRINGS.

Let us agree that the starting position for INPUT STRINGS is always the right-most 1 of the string, as indicated by the arrow in Fig. 7.

An INPUT STRING is termed ODD or EVEN according as the number of 1's in the string is ODD or EVEN. Thus the INPUT STRING on the left in Fig. 7 is ODD, while the string on the right is EVEN.

The primitive problem we want to consider is how to formulate a plan to determine the PARITY of any given INPUT STRING. To make the problem definite, we require that a report should be presented in the format shown in Figs. 8 and 9.

In Fig. 8, the INPUT STRING is ODD; for this case, we require (in addition to the input string) that the report consist of a 1 printed on the second square to the left of the INPUT STRING. For an EVEN INPUT STRING, as illustrated in Fig. 9, we require that the report consist of the INPUT STRING and of two 1's, printed into the second and third squares to the left of the INPUT STRING. Furthermore, we require that the plan should be stated in the form of a machine (Turing machine with the binary ALPHABET 0, 1), which is to yield for *every* INPUT STRING the correct report in the format just explained. In addition, the machine should stop in the position indicated by ** in Figs. 8 and 9, respectively.

After some experimentation, the reader may easily set up a machine that meets these requirements. Let us call such a machine a P-machine, standing for PARITY MACHINE. The reader may readily convince himself that the machine of Fig. 1 is a P-machine by assigning a few INPUT

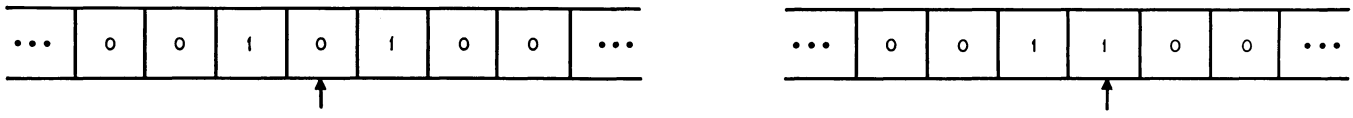


Fig. 3. Special inputs

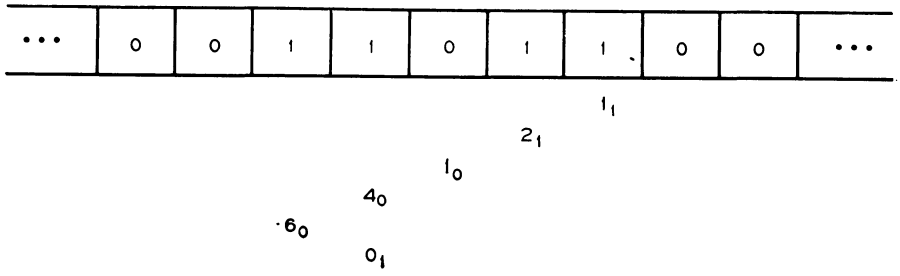


Fig. 4. Operating record

STRINGS and executing the instructions indicated in the cards of Fig. 1. Actually, the plan spelled out in this figure consists of two parts. First, the PARITY is determined, using the idea involved in the "She loves me, she loves me not" method. Then the report is made up in the required format.

An Optimality Problem

After some experimentation with the 6-card *P*-machine, the reader will observe that the second lines (1-lines) of cards 3, 4, 5, and 6 are not used at all in determining the PARITY of INPUT STRINGS. This raises the question whether it is possible to set up a *P*-machine with fewer than six cards.

For convenient reference, we denote by c_p the smallest possible number of cards that a *P*-machine may have. Since we have actually exhibited a 6-card *P*-machine, it is clear that

$$c_p \leqq 6 \qquad (6)$$

DEFINITION

A *P*-machine is *C*-optimal (optimal with respect to the number of cards used) provided that the number of its cards is equal to c_p.

OPTIMALITY PROBLEM

To find a *C*-optimal *P*-machine presents an optimality problem, typical in many ways of those encountered in the theory and use of automatic systems, including computers. Furthermore, various basic difficulties, relevant for artificial intelligence, are revealed in a concrete manner, uncluttered by technical and conceptual complications. Accordingly, the writers feel justified in discussing this deceptively primitive problem in some detail.

The approach that suggests itself in view of the inequality, equation 6, is to survey the class of machines with 5 cards to see if there exists a 5-card *P*-machine. The number of 5-card machines is astronomical; see equation 5. Hence, it is evident that we need some plan to carry out the search for a 5-card *P*-machine. Now it would seem that such a plan should be directly available. Indeed, somebody did succeed in finding a 6-card *P*-machine —the *P*-machine of Fig. 1. The discoverer must have followed some plan, since the number of 6-card machines is even more astronomical than the number of 5-card machines.

It is entirely reasonable to assume that this plan should be a valuable guide in searching for a 5-card *P*-machine. In fact, the interested reader may easily justify this plausible assumption. We observed that the 6-card *P*-machine first determines the PARITY of the INPUT STRING, using the idea of "She loves me, she loves me not." Then it prepares the report in the required format. After some experimentation, the reader should become aware of short cuts in accomplishing these objectives. In any case, the authors had little trouble in finding by this method the 5-card *P*-machine shown in Fig. 10, in which the hyphens indicate that the second lines (1-lines) of cards 3 and 4 are not used in determining PARITY.

While the reader may be elated by discovering his own 5-card *P*-machine, he will surely realize that the optimality problem is not yet solved. Indeed, the discovery of a 5-card *P*-machine enables us to replace the inequality 6 by the better inequality

$$c_p \leqq 5 \qquad (7)$$

But we still do not know whether $c_p < 5$; in fact, the presence of two unused lines in Fig. 10 suggests that there exists a 4-card *P*-machine. Furthermore, it is plausible that we shall discover such a machine by observing the way the 5-card *P*-machine of Fig. 10 does the job, and noticing after a while some short cut. In any case, this is what the authors tried to do; but when they finally found the 4-card *P*-machine shown in Fig. 11, something new was added to the idea of just looking for short cuts.

The reader who hand-checks this 4-card *P*-machine for a few INPUT STRINGS will note that the machine performs an apparently useless step; namely, it first prints a 1 in the square separating the INPUT STRING from the report (1 or 11), and then overprints this 1 by a 0 just in the nick of time. This illustrates a very basic phenomenon in the design and use of machines of any kind. At first, human beings try to design machines which simulate what human beings would do; this stage may be called the "anthropomorphic stage." However, this approach may be quite inefficient. Evidently, an automobile walking on two mechanical legs would be far less efficient than one rolling on wheels.

Briefly, optimal design and use of machines generally requires a complete breakaway from the initial anthropomorphic approach; more picturesquely, we may say that we have to dehumanize our approach if we want to obtain optimal design. On a very modest scale, this point is illustrated by the printing of a "useless" 1 in the course of the operation of the 4-card *P*-machine of Fig. 11. This step makes no sense from the

Fig. 5. Input for Fig. 6

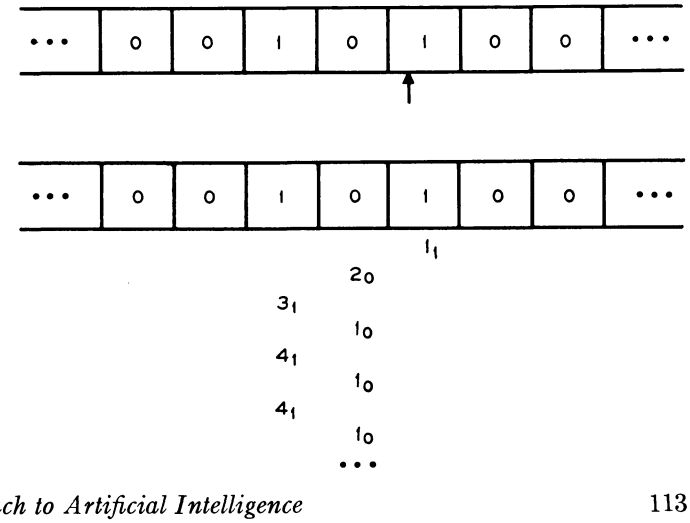


Fig. 6. Operating record for new input

Fig. 7. Two INPUT STRINGS

anthropomorphic point of view; rather, it takes advantage of the particular manner in which Turing machines operate. In the present case, the reader will soon realize that the design of card 3 in Fig. 11 represents a dehumanizing idea.

This philosophical discourse is meant to clarify the situation we have to face now in trying to solve our optimality problem. Our discovery of a 4-card *P*-machine enables us to sharpen the inequality 7 to the inequality

$$c_p \leqq 4 \qquad (8)$$

However, the presence of an unused line, indicated by hyphens in Fig. 11 suggests that there may exist a 3-card *P*-machine; in any case, this possibility *must* be investigated to solve completely our optimality problem.

The authors regret that they could neither find a 3-card *P*-machine, nor prove the nonexistence of same. Accordingly, some plan must be devised to settle this point in a definitive manner. In the absence of any plausible approach based on our experience in finding successively the 6-card, 5-card, 4-card *P*-machines, a natural thought is to delegate the task of searching for a 3-card *P*-machine to a computer program. Many would agree that this objective is quite simple conceptually and technically; yet we run into a number of basic difficulties in this relatively simple undertaking.

Equation 3 stated that the number of 3-card machines is

$$N_3 = 16{,}777{,}216$$

While quite large, this number certainly is not astronomical, even for actually existing computers. Therefore, a computer search cannot be dismissed as unrealistic on its face. In fact, this idea is quite analogous to the basic problem of finding the optimal design of a system in terms of available or assigned components. Using this analogy, a computer program to achieve our present objective should consist of three parts; it should:

Part 1. Generate, in a systematic manner, the description of all the 16,777,216 machines with three cards, in a form usable by a digital computer.

Part 2. For each one of these machines, simulate by the computer the operating procedure of the machine for INPUT STRINGS as previously defined.

Part 3. For each machine, decide and report in some prescribed format whether or not the machine is a *P*-machine.

Parts 1 and 2 present no difficulty. Indeed, jointly with younger associates, Parts 1 and 2 were programmed for several computers, in connection with problems arising in the Busy Beaver logical game; see reference 2. However, Part 3 presents difficulties of a high order. Let M^* be an individual 3-card machine. To qualify as a *P*-machine, certainly M^* must come to a stop after a while for every INPUT STRING and, in addition, it must report PARITY correctly. Thus our computer program should decide, as part of the total job, whether or not M^* will stop if presented with any INPUT STRING. Now this is an even more exacting task than the famous halting problem; see reference 3. This is known to be generally undecidable for essentially primitive and yet quite deep logical reasons. This negative result is just one of a whole class of results in modern mathematical logic which relate to undecidability, unsolvability, and noncomputability.[3] The interested reader may find some very primitive instances for these phenomena.[2] For our present purposes, we merely note the following implication: There is no assurance that there exists a computer program that is capable of executing Part 3.

This frustrating implication should not be interpreted to mean that our problem will never be solved. Indeed, the same frustrating implication applies if we want to find out whether or not there exists a 6-card, 5-card, or 4-card *P*-machine. Yet we *did* answer these three questions affirmatively, although, significantly, *not by a programmed process*, but rather by reliance upon the mysterious faculty of the human mind called *intuition*—more precisely, intuition strengthened and refined by experience.

The basic question regarding artificial intelligence as it seems to us is whether nonprogrammed mental processes derived from intuition can be spelled out explicitly in the form of a program or algorithm whose execution requires no understanding or insight and hence can be delegated to a machine. In many cases, this is possible; in others, as previously explained, this may be impossible for logical reasons.

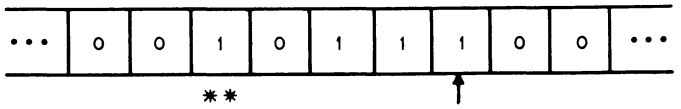
To clarify this point further, let us return to the discovery of the 4-card *P*-machine. Suppose we ask, after one such machine has been discovered, the following question: Just how many 4-card *P*-machines are there? To answer this question in terms of a computer program, we would again need a program to solve an aggravated form of the halting problem for 4-card machines; there is no evidence that such a program exists at all. Another pertinent observation is the following: The discovery of a 4-card *P*-machine was accomplished by using intuition rather than a computer program. However, once such a machine has been discovered, the *execution* of its instructions requires no insight, understanding or intelligence; the execution can be delegated to a computer. Observe that, by going on tape, a computer has at its disposal potentially infinite storage.

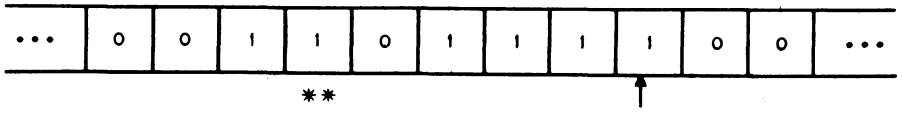
Truncated Problems

We repeat that a *P*-machine should report PARITY correctly for *every* INPUT STRING. Since there are infinitely many INPUT STRINGS, the question is whether the difficulties outlined are due to the requirement that a machine should do a certain job correctly in infinitely many

Fig. 8 (right). Format for reporting ODD string

Fig. 9 (below). Format for reporting EVEN string





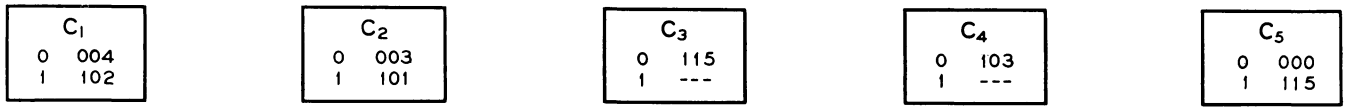


Fig. 10. A 5-card P-machine

Fig. 11. A 4-card P-machine

cases. Some may feel that such a requirement is unrealistic, and hence the comments presented in the preceding sections are essentially irrelevant, at least as far as engineering implementation is concerned.

This objection certainly deserves serious consideration; therefore, we might modify our optimality problem as follows: Let us assign a positive integer L, and define a P-L-machine as a machine that reports PARITY correctly for all INPUT STRINGS which contain not more than L 1's; that is, INPUT STRINGS of length not exceeding L. Let us say that a machine is a C-optimal P-L-machine if the number C of its cards is as small as possible, and c_{PL} is the smallest possible card-number for all P-L-machines.

The truncated optimality problem requires the determination of a C-optimal P-L-machine. Since the P-machines of the preceding sections are evidently P-L-machines for every choice of L, the discussion in those sections shows that we have to decide whether or not there exists a 3-card P-L-machine. Our task seems to be quite simple in principle. Let M^* be an individual 3-card machine; we have to decide whether it reports PARITY correctly for all INPUT STRINGS of length not exceeding the fixed number L. Since there are 16,777,216 machines with 3 cards, we have therefore to answer

$L \times 16{,}777{,}216$ questions

Large as this number may be, it is still finite, apparently giving us a finite problem, but one with basic difficulties. If we consider an individual 3-card machine M^* and an individual INPUT STRING i^*, to decide whether M^* reports the PARITY of i^* correctly, we should be able to decide whether M^* will stop after a while if presented with the INPUT STRING i^*. After a little experimentation, the reader will find that in some cases it is quite hard to answer this question. And, judging by our own experience, he will be at a loss to formulate a finite set of criteria which would automatically determine whether the machine will ever stop.

This problem can be truncated further by specifying the number of shifts allowed as a function of X, the length of the INPUT STRING. For example, the maximum number of shifts allowed might be $X+10$. The cases to be investigated then would be all three card machines with INPUT STRINGS of length not exceeding L and, for operating records, not exceeding $X+10$ shifts. This problem can be solved using the methods referred to in the next paragraph.

Suppose that, just to gain experience, we simplify the situation by merely asking whether a given 3-card machine will ever stop if started (with its card 1) on an all-zero tape. This particular question has been studied extensively by the authors in connection with the subject of sequential circuits. Many computer programs were written to answer this question; these programs grew larger and larger as more and more criteria for stoppers were covered. These programs were the results of co-operative efforts of experienced mathematicians and skilled programmers, and were run on some of the finest existing computers. Yet this extremely primitive-looking problem was still unsolved when this paper was presented, and probably most of the participants in the studies felt that perhaps it would always remain so. But since then, this problem has been solved by T. Rado and one of his graduate students, S. Lin.

Finite Problems and Proofs

The view is often expressed that finite problems are solvable, subject to limitations of a purely technological character. To illustrate a basic point in this connection, let us recall some facts from the field of optimal design of multiple-input multiple-output 2-level networks. There are now available various methods for this problem of optimal design; the heart of these methods is some way to reduce the fantastically astronomical number of possibilities to a search area of manageable size. However, even systems of modest size, say systems with six input lines and five output lines, may turn out to be exasperating "monsters"; after all available methods have been used, the search area is still hopelessly large. Accordingly, research efforts are directed toward further reductions of the search area.

A very basic point emerges in this connection: How can we be sure that the optimal case is still contained in the reduced search area? Evidently, this cannot be accomplished by actual inspection, due to the unmanageable number of cases. Hence, for every method to reduce the search area, there must be given a proof—an argument based on logical reasoning—that the baby has not been thrown out with the bathwater. The point is that even in dealing with finite problems, we may have to validate our procedure by mathematical proof, since actual inspection of all possibilities is out of the question. We stress this need for mathematical proofs, not merely as a

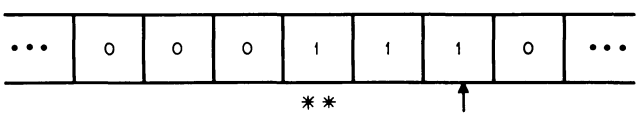


Fig. 12A. ODD report

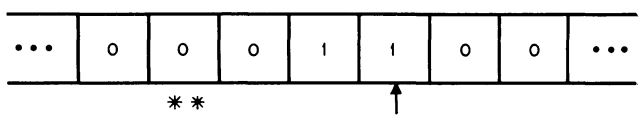


Fig. 12B. EVEN report

Fig. 13. A 2-card P*-machine

matter of annoying necessity, but also for the reason that adequate understanding and potential improvements of a method are best achieved as experience shows by searching and discovering a proof that the method works. In other words, mathematical proofs are important aids in research, as demonstrated by the experience of centuries.

Change of Format

Returning to our optimality problem for P-machines, we recall that it is still unsolved, since we were unable to determine whether there is a 3-card P-machine. Indeed, as far as evidence now available is concerned, our optimality problem will perhaps never be solved.

A very basic point emerges now. In formulating our optimality problem, we specified carefully the *input format* and the *output format.* As regards the output format, the essential feature was that the machine should report parity in a prearranged form or *code.* Now evidently we could choose some other code for the report. To be specific, let us modify the concept of a P-machine by modifying the report code (output format) as illustrated by Figs. 12(A) and 12(B).

If the INPUT STRING is ODD, then the machine should stop under left-most 1 of the INPUT STRING, as indicated in Fig. 12(A); if EVEN, the machine should stop under the second 0 to the left of the INPUT STRING as in Fig. 12(B). For clarity, let us call a machine that reports PARITY in this format a P^*-machine. Other requirements remain as before. Our new optimality problem is then to find a C-optimal P^*-machine. Since we merely changed the required format of the output by changing the code in which the machine is to report PARITY, there is no *a priori* reason to expect that the modified problem is less prohibitive than the original. Yet, the new problem is in fact trivial. Indeed, the reader will readily convince himself that the 2-card machine of Fig. 13 is a P^*-machine. Since evidently a 1-card machine cannot be a P^*-machine, the machine in Fig. 13 is a C-optimal P^*-machine, solving our optimality problem.

The fact that a change of format in formulating a problem may be of enormous importance is of course well known. In a general way, in the sciences and in technology, a change of format consists of modifying the theoretical model used. The heliocentric theory of our solar system is a famous example of the far-reaching consequences of an appropriate change of the theoretical model; the ever-accelerating progress in the sciences and in technology in modern times is due essentially to the constant search for more and more fruitful theoretical models. Our purpose in illustrating this basic point by the modest example of the P^*-machines is to suggest that the search for appropriate formulations of concepts and objectives in the field of artificial intelligence may be equally rewarding.

Conclusions

The examples discussed in this paper show that a number of basic theoretical and practical difficulties relevant for research in artificial intelligence arise even in problems that are truly primitive conceptually and technically. Furthermore, as suggested by our remarks on change of format, the selection and precise wording of basic concepts may be a decisive factor in achieving progress.

As the history of sciences and technology reveals, many of the finest accomplishments were adjudged as forever unattainable by leading experts of some earlier period. Accordingly, the rather frustrating picture presented here should not be construed to mean that research on artificial intelligence is a sterile enterprise. In the light of the lessons of history, the task of the present generation is to appreciate the enormous theoretical and practical difficulties involved, and to strengthen and refine our technical skill by selecting problems that are uncluttered by irrelevant complications. The authors believe that the study of conceptually simple and sufficiently representative problems has led the way in the past to significant advances.

References

1. INTRODUCTION TO METAMATHEMATICS (book), **S. C. Kleene.** D. Van Nostrand Company, Inc., Princeton, N. J., 1952.

2. ON NON-COMPUTABLE FUNCTIONS, **T. Rado.** *Bell System Technical Publications, Monograph 4199* July 1962, pp. 1–10.

3. COMPUTABILITY AND UNSOLVABILITY (book), **M. Davis.** McGraw-Hill Book Company, Inc., New York, N. Y., 1958.

Some Techniques of Optimization

ABRAHIM LAVI
MEMBER IEEE

Summary: Some common analytical and numerical techniques of finding the maximum or a minimum of multivariable functions are presented. The analytical methods given are Lagrange's method of undetermined multipliers, linear and dynamic programming; the numerical methods given are the univariate, factorial, steepest-ascend or -descend, and direct search. Wherever possible, examples are worked out to illustrate the applicability and the limitations of the method as well as the mechanics of obtaining a solution.

Paper 63-335, recommended by the AIEE Systems Science Committee and approved by the AIEE Technical Operations Department for presentation at the IEEE Winter General Meeting, New York, N. Y., January 27–February 1, 1963. Manuscript submitted October 31, 1962; made available for printing December 5, 1962.

ABRAHIM LAVI is with the Carnegie Institute of Technology, Pittsburgh, Pa.

THE fast development of our technology in the past decade has caused an awareness on the part of engineers and scientists that the traditional sharp dividing lines among the disciplines of engineering, science, and mathematics are more fictitious than real. The emergence of multidiscipline and interdisciplinary activities in government, in industry, and lately, in our universities, are the by-product of these phenomena. It is only natural that such activities should be identified, and hence, the names "system engineering" and "systems sciences."

There have been various definitions of system engineering[1,2] and there is not really room for more. Two of the attributes of the "systems" of interest here are complexity and cost. It is, therefore, mandatory that such complex and expensive entities be utilized to the limit of their capacity; that is, they should be optimized with respect to certain criteria. Such optimization may be the minimization of cost, the maximization of reliability, or the shortening of the manufacturing