Proceedings of the Symposium on Mathematical Theory of Automata
New York – April 1962
Polytechnic Press
of the polytechnic Institute of Brooklyn
1963

1963

ON A SIMPLE SOURCE FOR NON-COMPUTABLE FUNCTIONS*

Tibor Rado**
Department of Mathematics
The Ohio State University, Columbus, Ohio

In a paper published in the Bell System Technical Journal, I showed how a series of simply and clearly defined non-computable functions can be constructed, using only the principle that if E is a non-empty, finite set of non-negative integers, then E has a (unique) largest element. The purpose of the present paper is two-fold. First, some even more primitively defined non-computable functions (derived from the same principle) will be exhibited. Next, the following question will be considered. In binary digital computers with fixed word-length, the sort of phenomenon referred to above does not arise. Accordingly, there arises the possibility of constructing a model of arithmetic in which the "principle of the largest element" yields computable functions only. Comments on progress in this direction will be made to suggest plausible approaches.

I. INTRODUCTION

The following simple fact is of constant use in mathematics: if E is a non-empty, finite set of non-negative integers, then E has a largest element. For easier reference, let us denote by max E the non-negative integer which is the largest element of the non-empty, finite set E of non-negative integers (we leave the symbol max E undefined if E fails to satisfy the conditions just stated). Throughout mathematics, an integer x identified as $x = \max E$ (where E is known to satisfy the conditions described above) is considered as *well defined.* Assume, in addition, that E itself is given as the set of all those non-negative integers for which a given computable (that is, general recursive) function f(x) has the value zero (in symbols: $E = \{x \mid f(x) = 0\}$,), where we know that the equation $f(x) = 0$ has at least one and at most a finite number of zeros. Then we are given an algorithm to compute f(x), for each non-negative integer x, and hence to determine effectively whether or not x is an element of the set E. Under these conditions, the set E would be considered, under current mathematical standards, as exceptionally well defined,

* The results reported in this paper were obtained, in part, by research supported by the U.S. Army Research Office (Durham) under Contract DA-33-019-ORD-2114 and grant DA-ARO(D)-31-124-G53 to the Ohio State University Research Foundation, and in part by the Battelle Memorial Institute.

** Visiting professor at the Center for Advanced Studies, Wesleyan University, Middletown, Connecticut, during the Spring 1962 semester.

Presented at the *Symposium on Mathematical Theory of Automata,* Polytechnic Institute of Brooklyn, April 24, 25, 26, 1962

1963

and the integer max E would appear as well-defined too.

In a recent note, [1] the writer gave very simple examples of non-computable functions, using constructions which involve only integers given in the form max E; in addition, the sets E occurring in the constructions were exceptionally well defined. In this paper some further comments will be presented along these lines. Let us note that our main objective is to observe the phenomenon of non-computability in its simplest form, so that we can use the insight we achieve to see better what tasks we can delegate to computers. Actually, the comments to be presented here originated with the writer's studies relating to the optimal design of automatic systems, and specifically with efforts to use computers to the limit of their capabilities for this purpose.

II. COMMENTS ON TERMINOLOGY

We shall use as illustration the logical game called the "Busy Beaver" game in reference 1, merely recalling details needed here. We operate with Turing machines with the alphabet 0, 1 and a potentially both-ways infinite tape, in the sense of Kleene. [2] To explain certain unessential but quite convenient modifications of Kleene's terminology, we consider a particular instance of such a Turing machine (Fig. 1). In Fig. 1, C_1, C_2, C_3 stand for card 1, card 2, card 3, respectively, where we use card instead of state (or internal configuration); experience shows that internal configuration, for example, may discourage mathematicians unfamiliar with computers. On each card, the first column, from the left, is the alphabet column. The next column is the *overprint* by column. The next column shows the shift instruction, where 0 is the code for left shift, and 1 is the code for right shift. The rightmost column shows the subscript of the next card called after the shift, where a 0 is the code for Stop. We deviate from Kleene by not allowing a center shift (no shift at all); this is irrelevant, yet quite convenient for certain purposes. Otherwise, we follow Kleene's definitions; in particular, we adopt his concept of a function computed by a Turing machine (of the format just described).

C_1	
0	102
1	113

C_2	
0	111
1	102

C_3	
0	112
1	100

Figure 1

The particular 3-card Turing machine shown above is quite remarkable in a way; indeed, it is a current world's champion in the Busy Beaver 3-card classification (see reference 1 for further details). Competition in the 3-card classification of the Busy Beaver game (briefly, the BB-3 contest) is governed by the following rules:

i) The contestant writes his own 3 cards, conforming to the format prescribed above.

ii) He then submits his 3-card machine M, as well as a positive integer s (the shift-number) to a qualified representative of the International Busy Beaver Club.

Thus an entry in the BB-3 contest is a pair (M, s), where M is a 3-card machine and s is a positive integer. The umpire first checks whether the entry (M, s) is *valid*. To judge this, he starts the entry (M, s) with its card C_1 on a potentially both-ways infinite all-zero tape (that is, all the squares of the tape contain zeros). He then operates M, persisting through not more than s shifts. For the entry (M, s) to be valid, M must stop after exactly s shifts (the reason for requiring submission of the shift-number s is that it is sometimes quite hard to see whether a 3-card machine M will ever stop if started with card 1 on an all-zero tape). If the entry (M, s) is found valid, then the number of 1-s on the tape (at the time when M stops) is the score $\sigma(M, s)$ of the entry (M, s). For example, if the reader tests the 3-card machine shown above, he will find that it is a valid BB-3 entry with shift-number $s = 13$ and score 6 (see reference 1). In fact, this particular entry is one of several known 6-scorers in the BB-3 contest, and a current BB-3 champion, since no entry has been presented yet with a score exceeding 6.

Of course, for each positive integer n, the BB-n contest is defined in exactly the same manner. For each positive integer n, let $\Sigma(n)$ be the maximum possible score in the BB-n contest. It is shown in reference 1 that $\Sigma(n)$ is an exceptionally well-defined positive integer (in the sense of Section 1). For each positive integer n, let S(n) be the maximum possible shift-number s that may occur in a valid BB-n entry (see above). It is shown in reference 1 that, for every n, S(n) is an exceptionally well-defined positive integer.

It is trivial that $\Sigma(1) = 1$, $S(1) = 1$. It is quite an interesting problem to determine $\Sigma(2)$; with considerable help from the IBM 7090 (and considerable work on a computer program) it was found that $\Sigma(2) = 4$; but S(2) is yet unknown. As regards $\Sigma(3)$, the above 3-card machine shows that $\Sigma(3) \geq 6$ and $S(3) \geq 13$, but the precise values of $\Sigma(3)$, S(3) are unknown.* As regards $\Sigma(3)$, the ductory seminars, the determination of $\Sigma(3)$ and S(3) was attempted in terms of computer programs which became quite elaborate, without any definitive results. Dr. C. Y. Lee of the Bell Telephone Laboratories contributed the observation that $\Sigma(100)$ is greater than 10 raised to the 10 raised to the 10 raised to 50, 000, showing that while it is quite hard to reach a respectable score for low card numbers,

* Since this was written, the conjectures $\Sigma(3) = 6$, $S(3) = 21$ were confirmed by Mr. Shen Lin who continued the writer's efforts to find further behavior patterns to identify run-aways. This work will appear in the Bell System Technical Journal; it constitutes part of the doctoral thesis of Mr. Lin, a student of the writer.

fantastic scores may be expected if one considers larger but still quite moderate card numbers. In any case, even though skilled mathematicians and experienced programmers attempted to evaluate $\Sigma(3)$ and $S(3)$, there is no evidence that any presently known approach will yield the answer, even if we avail ourselves of high-speed computers and elaborate programs. As regards $\Sigma(4)$, $S(4)$, the situation seems to be entirely hopeless at present.

III.

Motivated by these circumstances, the writer found that $\Sigma(n)$, $S(n)$ (as functions of n) are in fact non-computable (not general recursive); the proof turned out to be surprisingly primitive. However, this result does not preclude the possibility of discovering algorithms that will yield the values of $\Sigma(n)$, $S(n)$ for particular, individual values of n; in fact, as noted above, we do know that $\Sigma(2) = 4$. Indeed, some rather puzzling results in this direction are readily available. Let us first note the inequalities

$$\Sigma(n) \leq S(n) \leq (n+1)\,\Sigma(5n) \cdot 2^{\Sigma(5n)} \tag{1}$$

which hold for every positive integer n.[1]

Let us now consider a fixed positive integer n. Then $\Sigma(n)$ is an exceptionally well-defined (finite) positive integer, and there exists at least one n-card machine $\widetilde{M}_n$ with the score $\Sigma(n)$. Now, denote by M^* the 1-card machine of Fig. 2. Next, denote by M_n^* the $(n+1)$ card machine represented by $M^* M_n$ (see Kleene[2] for notations). Keeping n fixed, let us put (onto the originally all-zero tape) $x+1$ consecutive 1-s to represent, in the sense of Kleene[2] and also of Davis,[3] the non-negative integer x, and let us start M_n^* under the rightmost 1 (with its first card). Then, proceeding to the left, M_n^* will first overprint the 1-s with zeros; then it shifts to the right, and $\widetilde{M}_n$ takes over. Since we know that starting from an all-zero tape, $\widetilde{M}_n$ will eventually stop printing $\Sigma(n)$ 1-s on the tape, it follows that M_n^* computes, in the sense of Davis, a function $b_n(x)$ which has the constant value $\Sigma(n)$. As regards the reference to Davis, let us note that he operates with Turing machines which differ slightly from ours (they are sets of "quadruples," while ours are sets of "quintuples"), but this is evidently irrelevant.

C_1
0 010
1 001

Figure 2

Thus we see that $\Sigma(n)$ is computable for each individual n, as the value of a constant function $b_n(x)$. We proceed presently to point out a consequence of the inequalities (1): namely, that $S(n)$ is also computable for each individual positive integer n. Let us fix n. We consider then $\Sigma(5n)$, for this fixed value of n. As we just

observed, $\Sigma(5n)$ is computable (by the machine M_{5n}^*). The computation of $S(n)$, for the fixed positive integer n, may be outlined as follows. Let K_n be the class of the n-card machines. The number of these machines is finite; in fact there are $[4(n+1)]^{2n}$ such machines which can be readily enumerated in a systematic, effective way. Let $M \in K_n$. We start M, with its card 1, on an all-zero tape and persist in operating it through not more than

$$L_n = (n+1)\,\Sigma(5n) \cdot 2^{\Sigma(5n)} \tag{2}$$

shifts. Now the inequalities (1) mean (since $M \in K_n$) that if M stops at all, then it stops after a certain number $s \leq L_n$ of shifts, where L_n is defined in Eq. (2). Hence, if M fails to stop within L_n shifts, it is identified as a "run-away." If M does stop within a certain number $s \leq L_n$ of shifts, then we note s; and in this manner we obtain a non-empty, finite set of positive integers consisting of the shift-numbers s of the *stoppers* in the class K_n; $S(n)$ is then simply the largest number in this set. This intuitive description can be readily translated into a program, as the experienced reader will easily see; in fact, we readily obtain the result that there exists a Turing machine that computes a constant function $c_n(x)$ whose value is $S(n)$. Actually the argument just sketched amounts to the fact that (in view of the inequalities (1)) $S(n)$ is computable *from* $\Sigma(n)$ and $5n$, in the sense of Kleene.[2]

IV.

As stressed by authors in the field of computable functions, the formal concept of computability (see Davis, reference 3, page 10, definition 2.5, and Kleene, reference 2, p. 360) is meant to capture the meaning of the vague, intuitive concept of "effective calculability." Now it is clear from the history of mathematics (as well as of the sciences in general) that perfect correspondence between an intuitive concept and its formal counterpart cannot be expected generally. For example, the nowhere differentiable continuous function of Weierstrass caused great surprise among mathematicians, for the evident reason that this sort of phenomenon was not anticipated in the intuitive concept of continuity.

Let us now consider the relationship between the formal concept of computability and the intuitive concept of effective calculability, in the light of the comments above on $\Sigma(n)$ and $S(n)$. As pointed out earlier in this paper, sustained efforts were made by skilled mathematicians and skilled programmers to calculate $\Sigma(3)$ and $S(3)$ effectively. It was found immediately that the heart of the issue is catching the 3-card run-away machines (never-stoppers). A first evident thought was that a run-away will somehow go into a "loop," somehow exhibit an oscillatory or periodic behavior. Actually, in one of the programs for calculating $\Sigma(3)$, about a dozen routines were included for the purpose of looking for certain types of behavior which (if observed) implied that the particular Turing machine processed was a run-away. In this manner, out of an initial number of about 17, 000, 000 3-card machines, only a few thousand were still undecided. However, the writer feels that all mathematicians and programmers who tried to calculate $\Sigma(3)$ or $S(3)$ agree that there is no evidence whatever that $\Sigma(3)$ or $S(3)$ will ever be "effectively calculated." As regards $\Sigma(4), \Sigma(5), \ldots, \Sigma(100)$, to remain in the range of reasonable integers n, the situation is entirely hopeless at this time.

Accordingly, we seem to be in the presence of a conflict between an intuitive concept (effective calculability) and its formal counterpart (computability). As pointed out earlier, perfect correspondence cannot be expected in general; furthermore, as the history of mathematics shows, such conflicts may be extremely fruitful if treated constructively. In the present context, the following comments may be relevant.

As pointed out above, the (formal) non-computability of the function $\Sigma(n)$, for example, is directly traceable to the definition of $\Sigma(n)$ for each individual n as the largest element of a non-empty, finite set of non-negative integers (where the set involved is, in fact, exceptionally well defined). Now, if one phrases this type of definition in terms of logical formulas, it is seen that we are faced with logical expressions referred to as of type $\forall\exists\forall$, in view of the order in which the universal and existential quantifiers $\forall$, $\exists$ appear. It is well known that logical expressions of this type correspond, generally, to unsolvable decision problems. Hence, if $\Sigma(n_0)$, for example, is computable for some particular n_0, then one would expect that the reason should be some *particular feature* exhibited by machines with n_0 cards, rather than a general theorem applicable to every n. This expectation is perhaps strengthened by the observation that in the extensive researches on solvable cases of decision problems of the type $\forall\exists\forall$, the issue is the identification of *special features* which make a particular decision problem of this type solvable.[4] Let us also recall the halting problem (see Davis[3]), not only is this problem undecidable for all Turing machines, but, as shown by Davis, there exists an *individual* Turing machine whose halting problem is undecidable. Hence one would rather expect that in a similar manner there may exist an individual positive integer n_0 for which $\Sigma(n_0)$, $S(n_0)$ are non-computable.

V.

In any case, the writer feels that in view of the efforts expended

on the calculation of $\Sigma(3)$, for instance, it is rather unrealistic to accept the *mere existence* of a Turing machine that computes $\Sigma(3)$ as evidence that $\Sigma(3)$ can be "effectively calculated." This realistic attitude is based, in part, on the writer's experiences in actual computer work (involving large and logically intricate programs concerned with the optimal design of automatic systems). In fact, he feels that, in his work with such programs, he profited greatly from the efforts expended (along with others) to subdue somehow the exasperatingly elusive BB-3 problem, even though this problem seems to be merely a nice exercise in a course for beginners.

A very simple and very direct answer to the questions raised here may very well be that the writer misinterpreted the definition of "computability" as stated, for example, by Kleene (reference 2, p. 360) or Davis (reference 3, page 10, definition 2.5). In a way, this would be a very gratifying outcome. Indeed, the BB-n problem would appear then as an instance of non-computability in its perhaps most primitive form, and hence as a potential source of new insights regarding the extent to which computers can relieve the human mind of monotonous tasks, setting it free to exercise its powers on the highest levels.

In conclusion, it should be noted that a detailed (and technically quite involved) study of the issues raised above is in progress. The purpose of the informal presentation in this note is to call attention to certain questions which seem to deserve further consideration.

REFERENCES

1. T Rado, "On Non-Computable Functions," *Bell Syst. Tech. J.*, Vol. 41, No. 3 (May 1962).
2. S.C. Kleene, *Introduction to Metamathematics* (New York: Van Nostrant, 1952).
3. M. Davis, *Computability and Unsolvability* (New York: McGraw-Hill 1958).
4. J. Suranyi, *Reduktionstborie des Entscheidungsproblems* (Hungarian Academy of Sciences, 1959).