AN INFORMAL ARITHMETICAL APPROACH TO COMPUTABILITY AND COMPUTATION, II

Z. A. Melzak

(received July 11, 1963)

11. In the first part of this paper [1] there was introduced a hypothetical computing device, the Q-machine. It was derived by abstracting from the process of calculating carried out by a man on his fingers, assuming an adequate supply of hands and the ability to grow fingers at will. The Q-machine was shown to be equal in computing power to a universal Turing machine. That is, the Q-machine could compute any number regarded as computable by any theory of computability developed so far. It may be recalled here that Turing machines were obtained by Turing [2] by abstracting from the process of calculating carried out by a man on some concrete 'symbol space' (tape, piece of paper, blackboard) by means of fixed but arbitrary symbols. Hence the contrast between the Q-machine and the Turing machines is that between arithmetical manipulation of counters and logical manipulation of symbols. In particular, one might say, loosely, that in a Turing machine, as in arithmetic, numbers are represented by signs whereas in the Q-machine, as on a counting frame, numbers represent themselves.

The programs of the Q-machine were written in terms of a ternary command scheme of the type

$$(1) \qquad \begin{array}{ccc} & A_a\ A_b\ A_c & \\ {}^{y}\swarrow & & \searrow^{n} \\ A_d\ A_e\ A_f & & A_g\ A_h\ A_i\ ; \end{array}$$

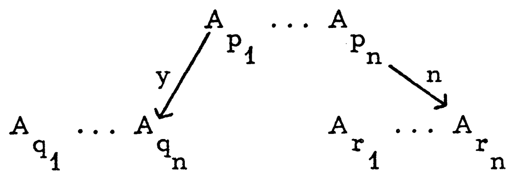
when that is executed, the contents p, q, r of the locations A_a, A_b, A_c become respectively $p-q\ \mathrm{sgn}(p-q)$, q, $r+q\ \mathrm{sgn}(p-q)$.

Canad. Math. Bull. vol. 7, no. 2, April 1964

https://doi.org/10.4153/CMB-1964-017-6 Published online by Cambridge University Press

Here sgn x is defined to be 1 if $x \geq 0$, and to be 0 otherwise.

The following generalization suggests itself: let n be a positive integer, $n \geq 2$, and let $f_i(x_1, \ldots, x_n)$, $i = 1, \ldots, n$, be n integer-valued functions, each one defined for all non-negative values of its integer arguments; now replace the scheme (1) by

(2) 

which is to be executed as follows: Let $c_1, \ldots, c_n$ be the contents of the locations $A_{p_1}, \ldots, A_{p_n}$; if

$$c_i + f_i(c_1, \ldots, c_n) \geq 0 , \qquad i = 1, \ldots, n ,$$

then the contents $c_1, \ldots, c_n$ are changed to $c_1 + f_1(c_1, \ldots, c_n)$, $\ldots$, $c_n + f_n(c_1, \ldots, c_n)$ respectively, and the next command to be carried out is $A_{q_1} \ldots A_{q_n}$. If for some i, $c_i + f_i(c_1, \ldots, c_n) < 0$ then the contents $c_1, \ldots, c_n$ are left unchanged and the next command to be carried out is $A_{r_1} \ldots A_{r_n}$. All previous conventions regarding subscripted locations A_{A_m}, program loops, sub-programs, stops etc. remain in force. Any such machine will be called a Q-machine (or a Q_n-machine, or the machine $Q_n(f_1, \ldots, f_n)$), n will be called its degree, and the functions $f_1, \ldots, f_n$ its transfer functions. Individual Q-machines will be distinguished by superscripts; thus Q_3^0 or $Q_3^0(f_1, f_2, f_3)$, with $f_1(c_1, c_2, c_3) = -f_3(c_1, c_2, c_3) = -c_2$, $f_2 = 0$, will be the old Q-machine of the first part of this paper. We make one final proviso: in (2) the commands may contain the store symbol S

https://doi.org/10.4153/CMB-1964-017-6 Published online by Cambridge University Press

once or more times, but not so as to involve the transfer of infinitely many counters; also, no command should be such that for any contents of the locations involved no counters get transferred. Thus the Q_3^0 commands $S\,S\,A_i$ or $A_i\,S\,A_j$ are excluded. When the first command in (2) is carried out, the original contents $c_1, \ldots, c_n$ are changed to

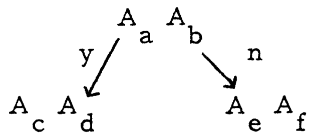
$$(3) \qquad c_i + f_i(c_1, \ldots, c_n) \prod_{j=1}^{n} \operatorname{sgn}\,[c_j + f_j(c_1, \ldots, c_n)], \quad i = 1, \ldots, n.$$

When $n_1 = n_2$ and $f_i = g_i$, $i = 1, \ldots, n_1$, then the Q-machines $Q_{n_1}(f_1, \ldots, f_n)$ and $Q_{n_2}(g_1, \ldots, g_n)$ are identical. However, it may happen that the transfer functions of two Q-machines of the same degree are not the same and yet their effects with any initial location-contents and for any program are the same; consider, for instance, Q_3^0 and the machine $Q_3^1(f'_1, f'_2, f'_3)$ with $f_1 = f'_1$, $f_3 = f'_3$ and $f'_2 = \operatorname{sgn}(c_2 - c_1 - 1)$. The point is that here f'_2 differs from f_2 only for $c_2 > c_1$, so that Q_3^1 and Q_3^0 are effectively the same.

Many questions may be raised now about the universality, loop types, structure etc. of the Q-machines and some of them will be considered here. The generalization from the old Q-machine to the present Q-machines is natural if one wishes to investigate the effect of allowing different basic operations in computing. For instance, Q_3^0 is essentially an adder while the machine $Q_5^0(f_1, \ldots, f_5)$, with $f_2 = f_3 = 0$, $f_1(c_1, c_2, c_3, c_4, c_5) = - c_2 - c_3 - c_2\,c_3$, $f_4(c_1, c_2, c_3, c_4, c_5) = c_2 + c_3$, and $f_5(c_1, c_2, c_3, c_4, c_5) = c_2\,c_3$, is essentially an adder-and-multiplier.

12. A Q-machine is universal if it can compute any computable number. In this section we describe the machine $Q_2^0(f_1, f_2)$ which has some claim to being the simplest of all

universal computing devices; here the transfer functions are $f_1(c_1, c_2) = - f_2(c_1, c_2) = - 1$. In words, there is a binary command scheme

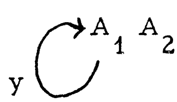


carried out by transferring one counter from the location A_a to A_b if A_a is not empty and proceeding to the command $A_c A_d$, and leaving the contents of A_a and A_b unchanged and then proceeding to the command $A_e A_f$ in case A_a is empty. It might be assumed that initially all the locations are empty since the Q_2^0 program $(S A_1)^p (S A_2)^q \ldots (S A_n)^r$ transforms the initial contents $0, \ldots, 0, \ldots$ to $p, q, \ldots, r, 0, \ldots$.
We note first two simple sub-routines for Q_2^0:

1) Transfer $T(A_1, A_2)$

initial contents: $p, q, 0, \ldots$

program:



final contents: $0, p + q, 0, \ldots$

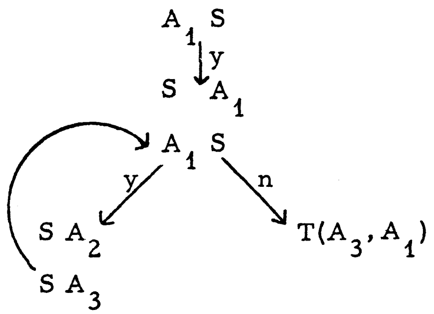
This program needs no reserved locations. The special case $T(A, S)$ reads 'empty A'.

2) Copy $C(A_1, A_2, A_3)$

initial contents: $p, q, 0, 0, \ldots$

https://doi.org/10.4153/CMB-1964-017-6 Published online by Cambridge University Press

program:



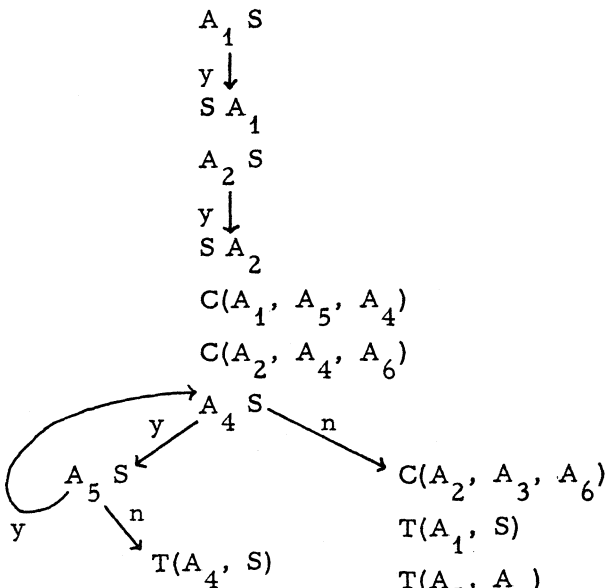
final contents: p, p+q, 0, 0, . . .

Now we have the

3) $Q^0_3 - Q^0_2$ equivalence program

initial contents: p, q, r, 0, 0, 0, . . .

program:



final contents: p - q sgn (p-q), q, r + q sgn (p-q), 0, 0, 0, . . .

This shows that the command $A_1 A_2 A_3$ can be simulated as a program on Q^0_2. Clearly the converse is also true: the command $A_1 A_2$ of Q^0_2 can be simulated as a Q^0_3 program consisting

https://doi.org/10.4153/CMB-1964-017-6 Published online by Cambridge University Press

of one instruction $A_1 A_3 A_2$, with the contents of A_3 equal to 1. Therefore any Q_3^0 program can be translated into its Q_2^0 equivalent and vice versa. Since Q_3^0 has already been shown to be universal, so is Q_2^0.

13. Consider the three already defined Q-machines Q_2^0, Q_3^0 and Q_5^0. They form a hierarchy: Q_2^0 can add 1, Q_3^0 can add two integers, Q_5^0 can add and multiply two integers. Define $A_1 A_2 0 A_4 A_5$ to be the Q_5^0 command $A_1 A_2 A_3 A_4 A_5$, subject to the condition that the location A_3 stays empty throughout. The effect of the command $A_1 A_2 0 A_4 A_5$ on the locations A_1, A_2 and A_4 is the same as that of the Q_3^0 command $A_1 A_2 A_4$. This leads to a generalization. Let $A_1 A_2 \ldots A_n$ be the command of a Q-machine and put

$$P(A_1 A_2 \ldots A_n) = A_1' A_2' \ldots A_n' ,$$

where each A_i' is either A_i itself or some fixed integer $p_i \geq 0$. In the latter case the location A_i retains its contents p_i throughout the program. Leaving out all inactive indices (referring to the locations to which fixed contents have been pre-assigned) we have

$$P(A_1 A_2 \ldots A_n) = A_{j_1} A_{j_2} \ldots A_{j_N} , \quad 1 \leq j_1 < j_2 < \ldots < j_N \leq n .$$

Now let $m \leq N$ and let $i_1, i_2, \ldots, i_m$ be an increasing subsequence of the sequence $j_1, j_2, \ldots, j_N$. This induces a Q_m-machine with the command $A_{i_1} A_{i_2} \ldots A_{i_m}$ which we shall call a contraction of the original Q-machine; the latter is called an extension of the former. We note that Q_5^0 is an extension of Q_3^0 and of Q_2^0 and Q_3^0 is an extension of Q_2^0, since

$$A_1 A_2 A_4 = A_1 A_2 0 A_4 0, \quad A_1 A_4 = A_1 1 0 A_4 0, \quad A_1 A_4 = A_1 1 A_4.$$

One Q-machine may be an extension or a contraction of several Q-machines. In fact, it is easy to show that any sequence $\left\{Q^k_{n_k}(f_{k1}, \ldots, f_{kn_k})\right\}$ $(k = 0, 1, \ldots)$ of Q-machines of bounded degrees, $n_k \leq N$, possesses a common extension $Q_{N+1}(F_1, \ldots, F_{N+1})$. For it may be assumed without loss of generality that $n_0 = n_1 = \ldots = N$ and then we have only to define

$$F_{N+1}(c_1, \ldots, c_{N+1}) = 0, \quad F_i(c_1, \ldots, c_N, k) = f_{ki}(c_1, \ldots, c_N), \; i = 1, \ldots, N$$

to have the desired extension property for $Q^k_{n_k} : A_1 A_2 \ldots A_N = A_1 A_2 \ldots A_N k$.

The following propositions are easily demonstrated: 1) any extension of a universal machine is universal; 2) the set of all Q-machines is uncountable and so is the set of all universal Q_n-machines for $n \geq 3$; 3) each finite or countably infinite set of Q-machines of degrees $\leq N$ has uncountably many common extensions which are universal and are of degree $N + 1$; 4) the binary relation $Q^a \leq Q^b$ (Q^b is an extension of Q^a) is a partial but not a total order on the set of all Q-machines; 5) defining $Q^a < Q^b$ by $Q^a \leq Q^b$ and $Q^b \not\leq Q^a$, we have uncountably many chains $Q^1 < Q^2 < \ldots$ starting with any given Q^1. In each case the uncountable multiplicity follows from the sufficient freedom of choice of values of one or more transfer functions.

14. The set of all Q-machines is inconveniently and somewhat unnaturally large, and we should like to limit it somehow. This limitation could be carried out in at least

https://doi.org/10.4153/CMB-1964-017-6 Published online by Cambridge University Press

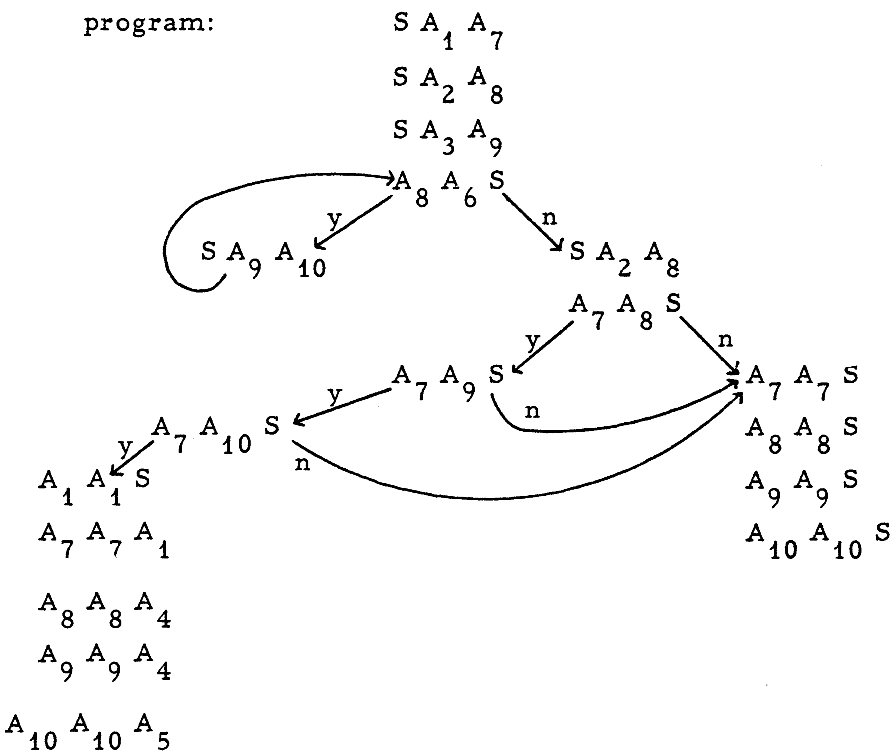
two essentially distinct ways. Proceeding 'internally', we may define a number of 'atomic arithmetical operations', such as the decision whether a finite set is empty or not, transferring one member from a non-empty set to another set, merging two finite sets, the decision whether one finite set has more members than another one etc., and then we may consider only those Q-machines whose commands are executable in terms of these arithmetical operations. This amounts to singling out certain basic functions and allowing only those Q-machines whose transfer functions are (allowable) compositions of the basic ones. The whole procedure may be suitably formalized and one obtains then something similar to the theory of recursive functions.

Proceeding 'externally', we choose a universal machine and we limit ourselves only to those Q-machines whose commands can be synthesized as programs of that universal machine. We shall follow this course and we shall take Q_2^0 as the basic universal machine. That is, in the remainder of this paper, unless the contrary is explicitly stated, we consider only those Q-machines with the command $A_1 A_2 \ldots A_n$, for which a Q_2^0 program $P = P(p_1, \ldots, p_M)$ exists, such that the command $A_1 \ldots A_n$ of Q_n, with the initial contents $p_1, \ldots, p_M, 0, \ldots$ leads to the same final contents as the Q_2^0 program P (with the initial contents all empty). On the other hand, for ease of description we can use instead of Q_2^0 either Q_3^0 or Q_5^0 or any other Q-machine which has been shown to be equivalent in the above sense to Q_2^0. The equivalence of Q_3^0 and Q_2^0 has been shown, the equivalence of Q_5^0 and Q_2^0 will follow from it and from the Q_5^0-Q_3^0 equivalence program given below. Since Q_5^0 is an extension of Q_3^0, it suffices to show that the command of Q_5^0 is obtainable as a Q_3^0 program.

https://doi.org/10.4153/CMB-1964-017-6 Published online by Cambridge University Press

initial contents: p, q, r, s, t, 1, 0, 0, 0, 0, ...

program:



final contents: the same as after the Q_5^0 command

$A_1 A_2 A_3 A_4 A_5$.

The set of all admissible Q-machines is now isomorphic to a subset $\mathcal{J}_2^0$ of the set of all Q_2^0 programs. $\mathcal{J}_2^0$ is countable and an explicit enumeration can be given. Recalling the conventions about the arrows, we can put every program $P \in \mathcal{J}_2^0$ in the form of a string $A_{a_1} A_{b_1}, A_{a_2} A_{b_2}, \ldots, A_{a_N} A_{b_N}$ of N successive pairs of the type $A_{a_i} A_{b_i}$; from each pair there issue two arrows marked 'y' and 'n' and leading to other pairs. We put $A_0 = S$ and we let in the last pair $a_N = b_N = 0$ interpreting S S as the command 'stop'. In each pair a_i and b_i are

https://doi.org/10.4153/CMB-1964-017-6 Published online by Cambridge University Press

non-negative integers or they may be themselves locations. Therefore each command in the string is uniquely determined by an ordered 6-tuple of non-negative integers $(p_i, q_i, r_i, s_i, t_i, u_i)$ in the following way: with $A_i A_j$ we associate $(i, 0, j, 0, h, k)$, with $A_{A_i} A_j$, $(i, 1, j, 0, h, k)$, with $A_i A_{A_j}$, $(i, 0, j, 1, h, k)$, and with $A_{A_i} A_{A_j}$, $(i, 1, j, 1, h, k)$.

In each case the fifth member k of the 6-tuple shows that the arrow marked 'y' leads to the k-th pair of the string and the sixth member h shows that the arrow marked 'n' leads to the h-th pair of the string. Now P becomes an ordered set of ordered 6-tuples $S_1, \ldots, S_N$; let $f : (n_1, n_2, n_3, n_4, n_5, n_6) \to n$ be a function mapping in a 1:1 fashion the set of all ordered 6-tuples of non-negative integers onto the non-negative integers themselves. With the program $P = (S_1, \ldots, S_N)$ we now associate the number $N_P = \prod_{i=1}^{N} p_i^{f(S_i)}$, where p_i is the i-th prime, starting with $p_1 = 2$. Of course, not every number is the number of a program, and a program with the number N_1 may be completely equivalent in its effects on all locations to a program with a different number N_2. If $P_1, P_2 \in \mathcal{P}_2^0$ we define the composite program $P_1 P_2$ by juxtaposition, with all the terminal arrows of P_1 leading to the initial command of P_2. Letting 1 stand for the empty program which does nothing, the set $\mathcal{P}_2^0$ becomes a countable semi-group with identity.

15. An interesting binary relation on the set of all Q-machines is obtained by letting $Q^a \leq Q^b$ if and only if the command of Q^a can be synthesized as a loopless program on Q^b. Let also $Q^a < Q^b$ if $Q^a \leq Q^b$ and $Q^a \not\geq Q^b$. For instance, we have $Q_2^0 < Q_3^0 < Q_5^0$; the reason for this is that the basic operations of these three machines (adding 1, adding

https://doi.org/10.4153/CMB-1964-017-6 Published online by Cambridge University Press

two integers, adding and multiplying two integers) are on different levels of a progressive hierarchy of the magnitude of the computed number. We derive now some sufficient conditions for two Q-machines, $Q^a = Q^a_n(f_1, \ldots, f_n)$ and $Q^b_m(g_1, \ldots, g_m)$, to ensure that $Q^a < Q^b$. A function $F(x)$ dominates Q^a if for every set of non-negative integer arguments $(c_1, \ldots, c_n)$

$$|c_i + f_i(c_1, \ldots, c_n)| < F(\max(c_1, \ldots, c_n)), \quad i = 1, \ldots, n .$$

For a positive integer k let $F_k(x)$ denote the k-th iterate of $F(x)$. We say that $G(x)$ majorizes $F(x)$ (and also that $G(x)$ majorizes Q^a) if for every fixed k there is x_0, such that $F_k(x) < G(x)$ provided that $x_0 < x$. It is well known that every function $F(x)$, no matter how fast its growth, possesses a majorizing function $G(x)$. For instance, we can take

$$G(x) = \sum_{j=1}^{\infty} a_j F_j(x) ,$$

where the sequence $\{a_j\}$ tends to 0 fast enough. Suppose now that a) Q^b is an extension of Q^a, and b) the first transfer function g_1 of Q^b is such that for suitable values $c_2, \ldots, c_m$ there occurs a transfer of at least $G(c_1)$ counters to some location, $G(x)$ being any function majorizing Q^a. Condition a) implies $Q^a \leq Q^b$. Observe next that any loop-free program of Q^a with a string of k consecutive commands, and with the initial contents $p_1, \ldots, p_N$, results in final contents none of which can exceed $F_k(\max(p_1, \ldots, p_N))$. On the other hand, a single command of Q^b can, with the same initial contents, produce final contents which exceed $F_k(\max(p_1, \ldots, p_N))$, no matter how large k may be. Therefore condition b) implies $Q^b \not\leq Q^a$. It follows that $Q^a < Q^b$. Since Q^a was arbitrary, we can carry on the process to obtain for any initial

https://doi.org/10.4153/CMB-1964-017-6 Published online by Cambridge University Press

$Q^1 = Q^a$ an infinite chain of the type $Q^1 < Q^2 < Q^3 < \dots$.

By the construction of this chain, for any n the Q^n program equivalent to the command of Q^{n+1} contains at least one loop, the Q^n program equivalent to the command of Q^{n+2} contains at least one loop-within-a-loop, and generally, the Q^n program equivalent to the command of Q^{n+k} contains at least one k-tuply imbedded loop. This shows that no matter which machine is used for computing, there will be numbers computing which calls for programs of arbitrarily high degree of loop imbedding.

16. For the purpose of this section we modify the definition of certain Q-machines so that they can handle negative integers. Consider first Q_3^0; its basic operations are addition of non-negative integers, restricted subtraction of integers (= formation of the difference $a - b$ when $a \geq b \geq 0$) and the conditional transfer which depends on the previous operation. Suppose now that instead of a single row of locations there is a double row

$$\begin{array}{l} A_1^+, A_2^+, \dots \\ A_1^-, A_2^-, \dots \end{array},$$

and let the contents of A_n be the difference (contents of A_n^+) - (contents of A_n^-). This amounts to the usual process of regarding integers as pairs of non-negative integers with the ordinary provision for pair equivalence: $(a, b) = a - b$ and $(a, b) = (c, d)$ if $a + d = b + c$. There is, as before, a single store S. The Q_3^0 command $A_1 A_2 A_3$ is now interpreted as the unbranched program

https://doi.org/10.4153/CMB-1964-017-6 Published online by Cambridge University Press

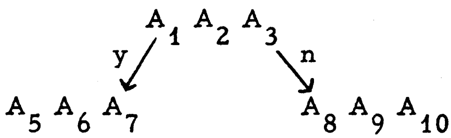
$$S\,A_2^-\,A_1^+$$

$$S\,A_2^+\,A_1^-$$

$$S\,A_2^+\,A_3^+$$

$$S\,A_2^-\,A_3^- \;.$$

Thus modified, Q_3^0 can handle negative integers. However, the ability to perform conditional transfer vanishes, since now subtraction is always possible. Here this ability can be easily restored: replace



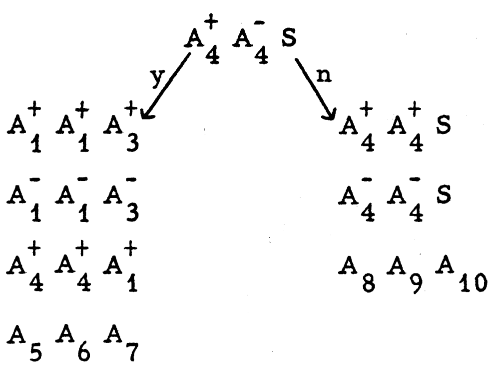
by the program

$$S\,A_1^+\,A_4^+$$

$$S\,A_2^-\,A_4^+$$

$$S\,A_2^+\,A_4^-$$

$$S\,A_1^-\,A_4^-$$



https://doi.org/10.4153/CMB-1964-017-6 Published online by Cambridge University Press

Let $\bar{Q}_3^0$ denote the above modification of Q_3^0 to handle negative integers. It is clear that with four rows of locations we can further modify Q_3^0 so that it can handle Gaussian integers.

In this section there will be proved a theorem, formulated in terms of the Q-machines, but having some independent interest. It is given here as an example of some uses of our theory. The question which led to it, was: what is the simplest way of computing?

In the first part of this paper we considered the sequence $\{p_n/q_n\}$ of fractions, $n = 0, 1, \ldots,$ with $p_0 > 0$, $q_0 > 0$ and with the recursion formula $p_{n+1} = p_n + 2q_n$, $q_{n+1} = p_n + q_n$. One has then $\lim_{n\to\infty} (p_n/q_n) = +2^{1/2}$, and an unbranched Q_3^0 program was given to compute each successive convergent fraction from the previous one by means of four additions of positive integers and no other operations. Since the addition of integers is one of the simplest possible operations, this raises the question: which (real) numbers can be computed by means of iterative schemes with a bounded number of additions of integers per iteration stage and no other operations?

The above formulation is not yet precise enough, and to emend it we define a real number x to be $\bar{Q}_3^0$ computable if there exists an unbranched $\bar{Q}_3^0$ program:

initial contents: $a_{10}, \ldots, a_{k0}, 0, \ldots$

program:

$$\begin{array}{ccc} A_{i_1} & A_{j_1} & A_{k_1} \\ \cdot & \cdot & \cdot \\ A_{i_p} & A_{j_p} & A_{k_p} \end{array}$$

which is non-terminating and which results in accumulating successively in certain two locations, say in A_1 and A_2, of two sequences of integers, $\{p_n\}$ and $\{q_n\}$, p_n and q_n being the contents after the n-th cycle, such that $\lim (p_n/q_n) = x$. The sequences $\{p_n\}$ and $\{q_n\}$ will be called admissible for x. Our question is now re-formulated: which numbers are $\bar{Q}_3^0$ computable? A partial answer is given in the following

Theorem. Any real $\bar{Q}_3^0$ computable number is algebraic. If x is a PV-number then any number of the form $(t_1 x + t_2)/(t_3 x + t_4)$, with the t_i's all integers, is $\bar{Q}_3^0$ computable.

Recall that the PV-numbers (Pisot-Vijayaraghavan numbers) are real positive algebraic integers greater than 1, all of whose other conjugates are less than 1 in absolute value. For their properties see [4].

The idea of the proof is to show first that every computation method for a $\bar{Q}_3^0$ computable number x is essentially the same as the above special case of the sequences $\{p_n\}$ and $\{q_n\}$ which are admissible for $+2^{1/2}$: there is a linear system of recurrence relations for a finite number of sequences $\{p_n\}$, $\{q_n\}$, ..., $\{z_n\}$, and x is the limit of a ratio, say (p_n/q_n). We observe first that since there are no subscripted locations A_{A_i}, only a finite number of locations is ever occupied throughout the program. It may be assumed therefore that the locations A_{k+1}, A_{k+2}, ... stay empty throughout.

Let a_{in}, $i = 1, \ldots, k$, be the contents of the i-th location after the n-th cycle. Since each successive command of the program results in location contents which are linear combinations of the contents before the command, we have

$$\text{(4)} \qquad a_{i\,n+1} = \sum_{j=1}^{k} c_{ij}\, a_{jn}, \quad i = 1, \ldots, k;\ a_{10}, \ldots, a_{k0} \text{ given integers.}$$

The integers c_{ij} are constants independent of n since there are no subscripted locations. Solution of the system (4) is obtained by assuming that

$$(5) \qquad a_{in} = \sum_{s=1}^{k} b_{is} \lambda_s^n , \qquad i = 1, \ldots, k; \; n = 0, 1, \ldots ;$$

on substituting (5) into (4) and equating the coefficients of each λ_s^n one finds that $\lambda_1, \ldots, \lambda_k$ are the eigenvalues of the matrix (c_{ij}). Therefore the procedure is justified provided that the eigenvalues $\lambda_1, \ldots, \lambda_k$ are all distinct. Otherwise one has instead of (5)

$$(6) \qquad a_{in} = \sum_{s=1}^{K} B_{is}(n) \lambda_s^n , \qquad i = 1, \ldots, k; \; n = 0, 1, \ldots ;$$

here λ_s is an eigenvalue of multiplicity n_s, $B_{is}(n)$ is a polynomial in n of degree $n_s - 1$ and $\sum_{s=1}^{K} n_s = k$. The coefficients of the polynomials B_{is} are obtained on substituting (6) into (4) and equating the coefficients of $n^p \lambda_s^n$ for each pair p, s. In either case, whether the eigenvalues are simple or not, the coefficients b_{is} or the coefficients of the polynomials B_{is} are obtainable by algebraic processes from the integers c_{ij} and a_{i0} and the eigenvalues λ_i, which are algebraic numbers. Therefore these coefficients are themselves algebraic. If x is a $\bar{Q}_3^0$ computable number then by definition $x = \lim_{n\to\infty} (a_{in} / a_{jn})$, so x is either some ratio b_{is}/b_{js} or the ratio of the leading coefficients in certain two polynomials B_{is} and B_{js}. In either case x is algebraic.

Suppose next that x is $\bar{Q}_3^0$ computable with the admissible sequences $\{p_n\}$ and $\{q_n\}$. Therefore, if the t_i's are integers, the number $(t_1 x + t_2)/(t_3 x + t_4)$ is also $\bar{Q}_3^0$ computable with the admissible sequences $\{t_1 p_n + t_2 q_n\}$ and $\{t_3 p_n + t_4 q_n\}$. It remains to show that a PV-number is $\bar{Q}_3^0$ computable. The integers c_{ij} in (4) are arbitrary - let them be chosen so that the given PV-number x is one of the eigenvalues λ_s of the matrix (c_{ij}). Then all the other eigenvalues are less than x in absolute value, so that

$$x = \lim_{n \to \infty} (a_{1\,n+1} / a_{1n}) = \lim_{n \to \infty} [(\sum_{s=1}^{k} b_{is} \lambda_s^{n+1})/(\sum_{s=1}^{k} b_{is} \lambda_s^{n})]$$

and hence the sequences $\{a_{1\,n+1}\}$ and $\{a_{1n}\}$ are admissible for x. Thus x is $\bar{Q}_3^0$ computable.

One can similarly consider the $\bar{Q}_5^0$ computable numbers. It is first necessary to modify Q_5^0 to operate with negative numbers; this can be arranged without difficulty, as for Q_3^0. Since only unbranched programs will be considered, there is no need to restore the conditional transfer (although this could be done easily). We define a real number to be $\bar{Q}_5^0$ computable in the same way as before, replacing $\bar{Q}_3^0$ by $\bar{Q}_5^0$. Since $\bar{Q}_5^0$ can add and multiply, instead of the linear combinations of (4) we have now polynomial combinations

$$(7) \qquad a_{i\,n+1} = P_i(a_{1n}, \ldots, a_{kn}), \ i = 1, \ldots, k; \ n = 0, 1, \ldots ,$$

and the coefficients of the polynomials P_i are constants independent of n since there are no subscripted locations A_{A_i} in the program. It follows that our $\bar{Q}_5^0$ computable numbers coincide with the rationally recursive numbers introduced and investigated in [3]. Among other open questions posed in [3] there are: 1) is every rationally recursive number computable?,

https://doi.org/10.4153/CMB-1964-017-6 Published online by Cambridge University Press

and 2) can one exhibit a computable number which is not rationally recursive (the existence of such is easy to prove)?

From the equation 'rationally recursive' = '$\bar{Q}_5^0$ computable' it follows at once that the answer to 1) is yes. Starting with the remarks at the end of the previous section, it is not hard to exhibit a computable number which is not $\bar{Q}_5^0$ computable. For instance, let

$$2_{(1)}(x) = 2^x, \quad 2_{(n+1)}(x) = 2^{2_{(n)}(x)}, \quad (k!)_{(1)} = k!, \quad (k!)_{(n+1)} = [(k!)_{(n)}]!$$

and define

$$F(x) = \sum_{n=1}^{\infty} 2_{(n)}(x) / (n!)_{(n)} ;$$

then for every positive integer m the number $F(m)$ is computable but not $\bar{Q}_5^0$ computable.

The author wishes to thank the referee for improvements, corrections and suggestions.

REFERENCES

1. Z.A. Melzak, An Informal Arithmetical Introduction to Computability and Computation, Can. Math. Bull., vol. 4, no. 3, Sept. 1961.

2. A.M. Turing, On Computable Numbers with an Application to the Entscheidungsproblem, J. Lond. Math. Soc., Spring 1936.

3. H.S. Shapiro, Rational Recurrence Formulae, Comm. Pure Appl. Math., vol. 12, no. 3, Aug. 1959.

4. J.W.S. Cassels, An Introduction to Diophantine Approximation, Cambridge Univ. Press, 1957.

The University of British Columbia