KYBERNETIKA ČÍSLO 5, ROČNÍK 5/1969

Recursive Functions Computable within $C\bar{f}\log\bar{f}$

Jiří Hořejš

The class **L** of functions $f(x)$ computable on classicial (i.e. one-tape, off-line) Turing Machines within $C\overline{f(x)}\log\overline{f(x)}$ steps is dealt (with $\overline{f(x)} = \max(x, f(x), 1)$). **L** is shown to be the simplest class of functions beyond the abilities of finite automata and it turns out to be sufficiently rich to comprise polynomials, to be closed under addition, multiplication and exponentiation, but not under substraction and composition.

INTRODUCTION

We shall adopt the model of "*classical*" Turing machines, wellknown from the literature (see e.g. [D]). Every machines is provided with one both-side infinite tape and is designed for computing the values of a number–theoretic function. As we shall restrict ourselves to (total, computable – i.e. recursive) functions of one variable only (a lot of results can be, however, easily generalized, which is left to the reader), we shall modify the model of [D] by assuming the argument x recorded by means of x (rather than $x + 1$) succesive 1's. Thus a machine Z computes a total function $f(x)$ mapping the set N of all non-negative integers into itself iff, whenever the word 1^x (i.e. $11 \ldots 1$, x-times) is written down on the tape, the head is placed on the leftmost symbol of this word, and the computation is started, it will finally halt, leaving $f(x)$ symbols "1" printed (not necessarily in adjancet squares) on the tape.

Let $T(x)$ (or, more precisely, $T_Z(x)$ or $T_f(x)$) denote an upper bound for the number of steps in which Z computes $f(x)$; i.e. let the length of computation performed by Z when this starts from the instantaneous description $q_0 1^x$ do not exceed $T(x)$, for $x = 0, 1, 2, \ldots$ We shall say that in this case $f(x)$ is *computable within* $T(x)$. It will be of our main interest the class **L** of all such functions f, for which $T_f(x) = C\overline{f(x)}\log f(x)$,*

* We shall not explicitly round off the non-integer values of $\log x$; this is supposed to be performed automatically whenever needed. Also, due to the use of the multiplicative constants, the base of logarithms need not be specified. Moreover we put $\log 0 = \log 1 = 1$.

where $\bar{f}(x) = \max(x, f(x), 1)$. Namely, we shall show that the class **L** comprises in some sense the "simplest" functions (theorem 1); on the other hand, **L** turns out to be sufficiently rich (theorems 2, 3) though not closed under all usual operations (theorem 5). The consideration of more comprehensive classes is postponed to another work.

An important role is played by the notion of the *crossing sequence* (c.seq.) at a border between two adjacent squares of the tape, this being defined as a (finite) sequence of internal states (of a machine Z computing a function f) in which the machine crosses succesively the given border. Main properties of crossing sequences are described and utilized both in [H] and in [T] and we shall assume that the reader is familiar with at least one of these papers.

The present work aims to add some results in the direction of the mentioned papers for the Turing machines (and their tasks) which seem still to be used most frequently.

§ 1.

A function $f(x)$ is called *quasi-periodic*, if there is a number $k \geqq 1$ and numbers $x_1, \ldots, x_k, d_1, \ldots, d_k, r_1, \ldots, r_k, x_i < x_{i+1}$ for $i = 1, \ldots, k - 1$, such that

$$(1) \qquad f(x_i + nd_i) = f(x_i) + nr_i \quad (i = 1, \ldots, k)$$

and every $x > x_k$ can be written in the form $x = x_i + nd_i$ for at least one $i \in \{1, \ldots, k\}$ and suitable n. For $d_i = d$ and $r_i = 0$ $(i = 1, \ldots, k)$ we obtain an ultimately periodic function, for $k = 1$ an ultimately linear step-function.

Quasi-periodic functions are rather special. In spite of their apparently complex structure, they are easily manageable; they can even be processed by a special sort of finite automata, e.g. in the following way: Let a quasi-periodic function $f(x)$ from the definition be given; consider a finite automaton A with input sequences of the form 1^x; A controls the input sequences (i.e. it is permitted to delay accepting the input 1's when necessary). The values $f(0), \ldots, f(x_k)$ are remembered in the internal memory of A and are handled separately in a suitable way. For $x \geqq x_k$, the resulting $f(x)$ is placed on one of k output tapes (which serve for recording the result only, not as a memory). This is enabled by the following algorithm, which describes the activity of A: After accepting first x_i input signals, A prints $f(x_i)$ output 1's on the i-th tape, while every next d_i steps, r_i another 1's are added on the same tape $(i = 1, \ldots, k)$. For every $x \geqq x_k$, $f(x)$ is thus written on (at least) one output tape, the number i of which can be (for given x) specified according to the relation $x = x_i + nd_i$ (i.e. by the d_i which was just counted).

The following theorem shows thus the relative "triviality" of functions which can be computed "faster" than in $C\overline{f(x)} \log \overline{f(x)}$:

Theorem 1. *If $f(x)$ is computable within $T(x)$, where*

$$\lim \frac{T(x)}{f(x) \log f(x)} = 0\,,$$

then $f(x)$ is quasi-periodic.

Proof. Let $f(x)$ be computed by a machine Z. Consider the segment S_x of the tape, where 1^x is initially written, as pictured on fig. 1. It is $|S_x| = x$ (for every word or segment of a tape u, $|u|$ denotes the length of u). Let us denote by L_x or R_x the

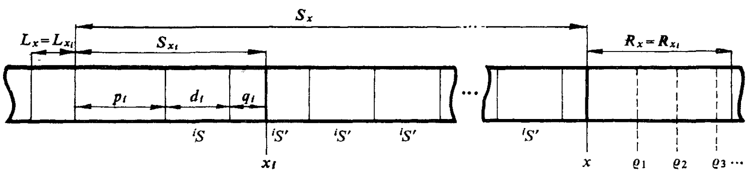


Fig. 1.

actually used parts of the tape to the left or to the right of S_x, respectively (i.e. every square of L_x and R_x will be visited at least once during the computation of Z). L_x or R_x may turn out to be empty.

First of all we prove:

(i) For every x it holds: all c.seqs. inside R_x are mutually different (it is understood that we speak about c.seqs at different borders): if there were – for some x – two of them, say ϱ_1 and ϱ_2, such that $\varrho_1 = \varrho_2$, then there would exist other borders with c.seqs $\varrho_3, \varrho_4, \ldots$ such that $\varrho_i = \varrho_1$ and $\varrho_{i+1} - \varrho_i = \varrho_2 - \varrho_1$ $(i = 2, 3, \ldots)$ and this would mean that Z would never halt, which is in contradiction with the totality of $f(x)$. ($\varrho_i - \varrho_j$ denotes the number of squares between borders, at which ϱ_i and ϱ_j are considered.)

(ii) For every x it holds: all c.seqs inside L_x are mutually different; this is seen in the same way as (i).

(iii) If $f(x)$ is not quasi-periodic then for infinitely many x there are at least $x/3$ mutually different c.seqs in S_x.

To prove (iii), consider the following construction. Denote by G the set of all x such that in S_x there occur (during the computation of $f(x)$ by Z) at least two equal c.seqs.

Define succesively:

$$x_1 = \min \{y \mid y \in G\}\,.$$

Let ${}^1S$ be a subsegment of S_x such that:

(α) the c.seqs at the ends of ${}^1S$ are equal (during the computation of $f(x_1)$), but inside ${}^1S$, there is no pair of equal c.seqs;

(β) there exists no other segment with the property (α) with its left end to the left of ${}^1S$. Such ${}^1S$ evidently exists as soon as x_1 does; denote $|{}^1S| = d_1$ and define

$$X_1 = \{x_1 + nd_1 \mid n = 0, 1, 2, \ldots\}\,, \quad Y_1 = X_1\,,$$

$$x_2 = \min\{y \mid y \in \mathsf{G},\, y \notin Y_1\}\,.$$

Let ${}^2S$ be a subsegment of S_{x_2} such that

(α) the c.seqs at the ends of ${}^2S$ are equal (during the computation of $f(x_2)$), but inside ${}^2S$, there is no pair of equal c.seqs;

(β) there exists no other segment with the property (α) with its left end to the left of ${}^2S$. Denote $|{}^2S| = d_2$ and put

$$X_2 = \{x_2 + nd_2 \mid n = 0, 1, 2, \ldots\}\,,$$

$$Y_2 = Y_1 \cup X_2,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\,,$$

$$x_i = \min\{y \mid y \in \mathsf{G},\, y \notin Y_{i-1}\}\,.$$

Let ${}^iS$ be a subsegment of S_{x_i} such that

(α) the c.seqs at the ends ${}^iS$ are equal (during the computation of $f(x_i)$), but inside ${}^iS$, there is no pair of equal c.seqs;

(β) there exists no other segment with the property (α) with its left end to the left of ${}^iS$. Denote $|{}^iS| = d_i$ and put

$$X_i = \{x_i + nd_i \mid i = 0, 1, 2, \ldots\}\,,$$

$$Y_i = Y_{i-1} \cup X_i\,,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

Finally, put $\mathscr{X} = \{x_i\}$ and $Y = \cup Y_i$. Distinguish now the following cases:

(1) $\mathscr{X}$ may turn out to be empty. In this case G is empty and (iii) trivially holds, as for all x all c.seqs in S_x are mutually different.

(2) $\mathscr{X}$ is nonempty, but finite. Three subcases are considered:

(a) $Y = N$. Let $\mathscr{X} = \{x_1, \ldots, x_k\}$. In this case every $x \in N$ can be expressed in the form $x = x_i + nd_i$ for suitable $i \in \{1, \ldots, k\}$, and $f(x) = f(x_i + nd_i) = f(x_i) + nr_i$, where r_i denotes the number of 1's left - after the computation of $f(x)$ ends - inside (every copy of) the segment ${}^iS$ (see fig. 1, where ${}^iS'$ denotes the copies of ${}^iS$). $f(x)$ is thus quasiperiodic.

(b) $N - Y$ is finite; considerations of (a) apply to every $x > x_0$ for some x_0, showing that $f(x)$ is quasi-periodic again (note that in (1), the x's need not be the minimal ones with the stated properties).

(c) $N - Y$ is infinite. Hence, there are infinitely many x, for which only mutually different c.seqs occur in S_x during the computation of $f(x)$, so that (iii) is satisfied again.

(3) $\mathscr{X}$ is infinite. Now we shall show that for all $x_i \in \mathscr{X}$, there are at least $x_i/3$ mutually different c.seqs in S_{x_i}; this will conclude the proof of (iii).

Consider an S_{x_i} (fig. 1); denote by q_i (p_i) the number of squares between the right (left) end of ${}^iS$ and the right (left) end of S_{x_i}. Logically, there are two possibilities:

(a) $q_i < d_i$ so that either $p_i \geqq x_i/3$ or $d_i = x_i/3$. But both to the left of ${}^iS$ and inside ${}^iS$ only mutually different c.seqs occur (see (α) and (β) clauses above) so that (iii) is established;

(b) $q_i \geqq d_i$; consider $x'_i = x_i - d_i$; as $q_i \geqq d_i$, it is $x'_i \in \mathrm{G}$ and due to the definition of x_i further $x'_i \in Y_j$ for some $j < i$, so that the leftmost pair of equal c.seqs of $S_{x'_i}$ is precisely that of S_{x_j}. Because $S_{x'_i}$ and S_{x_i} have the leftmost pair of equal c.seqs common again, we conclude that S_{x_i} and S_{x_j} have the same ones and hence $d_i = d_j$ and $x_j \in Y_j$; the contradiction with the definition of x_i shows that the case (b) cannot take place.

Having (iii) established, we summarize, that whenever $f(x)$ is not quasiperiodic, there are infinitely many x such that the number of mutually different c.seqs in S_x is proportional to $|S_x| = x$. The same holds – even unconditionally – for L_x, R_x.

Put $l(x) = |L_x| + |R_x| + |S_x|$ and $m(x) = \max(|L_x|, |R_x|, |S_x|)$. Trivially, $3m(x) \geqq l(x)$. With respect to the proved inequality of c.seqs it is (see [T]) $T_Z(x) \geqq K'\, m(x)\,.\,\log m(x)$ for suitable $K' \geqq 0$ and infinitely many x as soon as $f(x)$ is not quasiperiodic. Hence, for non-quasiperiodic functions

$$T(x) \geqq K\, l(x) \log l(x) \tag{3}$$

and because $l(x) \geqq \overline{f(x)}$, we have $T(x) \geqq K \overline{f(x)} \log \overline{f(x)}$ for suitable $K > 0$ and infinitely many x, so that

$$\lim \frac{T(x)}{\overline{f(x)} \log \overline{f(x)}}$$

cannot equal zero. The contradiction concludes the proof.

Theorem 1 shows that the class **L** splits in two subclasses, one of which is formed by functions, for which the upper bound $C\bar{f} \log \bar{f}$ is overestimated – but these functions can be but quasiperiodic – and the other by functions the computation of which utilizes the allowed time limit fully. Especially these are dealt with in what follows. Let us, however, remark to the end of this paragraph that neither of the functions from **L** can use too much of the tape. Namely, using (3) we prove:

Lemma 1. *Let $l(x)$ denote the length of the tape, actually used during the computation of $f(x), f(x) \in \mathbf{L}$. Then $l(x) \leqq L\overline{f(x)}$ for suitable L and all x.*

(This reminds partially the definition of linear bounded automata by Myhill [M].)

Proof. For a function f which is not quasiperiodic this follows from

$$T_f(x) \geqq K\, l(x) \log l(x) \quad \text{(cf. (3))}$$

and

$$T_f(x) = C f(x) \log f(x) \quad (\text{definition of } \mathbf{L})\,.$$

In the case of a quasiperiodic function f we proceed analogously, but only L_x and R_x are utilized and instead of (3) the inequality $T_f(x) \geqq K(l(x) - x) \log (l(x) - x)$ is obtained. It suits, however, as well.

Thus, the resulting 1's cannot be dispersed on the tape too infrequently.

§ 2.

Now the question arises, what can really be done within $C\bar{f} \log \bar{f}$, which functions belong to $\mathbf{L}$ and which do not. We shall try to give some characterizations in this direction. First we prove few closure properties of $\mathbf{L}$.

Theorem 2. (a) $f(x) \equiv x \in \mathbf{L}$,

(b) $f(x) = K \in \mathbf{L}$ $(K \in N)$,

(c) *if $f(x)$, $g(x) \in \mathbf{L}$ then $f(x) + g(x) \in \mathbf{L}$,*

(d) *if $f(x)$, $g(x) \in \mathbf{L}$ and $f(x)$, $g(x) \leqq f(x)\, g(x)$ for all x, then $f(x)\, g(x) \in \mathbf{L}$.*

Note. The additional assumption $f(x)$, $g(x) \leqq f(x)\, g(x)$ is not too restrictive. It is satisfied if $f(x) = g(x) = 0$ whenever $f(x)\, g(x) = 0$ (especially for $f(x) = x^m$, $g(x) = x^n$, $m, n \geqq 0$). Yet it may be weakened a little more by supposing that it holds for almost all x.

Corollary. *Polynomials over N (with nonnegative coeficients!) belong to $\mathbf{L}$.*

Proof. (a) and (b) is evident. Before the proof of (c) and (d), we shall prove three lemmas, which express the properties of the "unary" coding of numbers, underlying following considerations. Let us note the fact that an arbitrary working alphabet is at our disposal; it is often useful to consider the tape divided into a fixed number of tracks along the length of the tape, and, correspondingly, the term: "a symbol is written on the tape (in the square)" may mean "a symbol is written in a track of the tape (in a place where a track meets a column of the tape)".

Lemma 2. ("Encoding" lemma). *Let S denote a marked (by suitable markers placed on both ends) segment of the tape, such that it contains precisely s symbols "1" $(s \geqq 0)$ and the head is scanning some square of S. Then the words $\bar{s}$, representing a binary code of s can be written down on a prescribed (by pointing out its first or last square) place in S within $E\, |S| \log |S|$ steps.*

The proof is not difficult and for the special case, when all squares contain the symbol 1, was already described elsewhere. Let us therefore mention the rough idea only. The head moves repeatedly through S from its leftmost symbol to its rightmost symbol and back, crossing out in any such movement from the left to the right every even 1; if the last 1 (the rightmost one) is not crossed out according to this rule, it is crossed subsequently and the symbol 1 is attached to the $\tilde{s}$ which is being

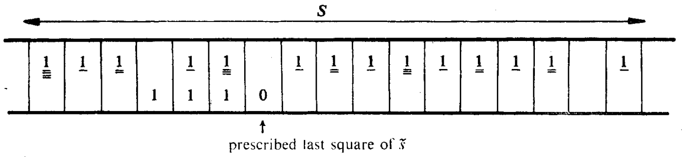


Fig. 2.

formed; in the other case, the symbol 0 is added. The process goes on as far as there are some of the original 1's on the tape; it is repeated at most $\log_2 |S|$ times and every single move lasts $2|S|$ steps; the assertion follows. Fig. 2 illustrates the case $s = 14$, $|S| = 17$.

From lemma 1 and lemma 2 it follows:

Lemma 3. *If $f(x) \in \mathbf{L}$, then within $C\overline{f(x)} \log \overline{f(x)}$ steps the result of computation of $f(x)$ can be printed on the tape (not only in the form of $f(x)$ symbols "1" spread over the tape but even) in the form $\widetilde{f(x)}$.*

Lemma 4. ("Decoding" lemma). *Let a marked word $\tilde{s}$ be written down on the tape and the head scan its leftmost symbol equal to 1. Then the marked word 1^s can be printed on the tape within $D . s \log s$ steps such that the left (the right) ends of both words coincide.*

Proof. Let a "loop" consist of the performance of the following two activities:

(1) shift of a given (marked) word $\tilde{p}$ (over the alphabet $\{0, 1\}$) one square to the right and inserting 1 in the released square;

(2) transformation of the newly placed word $\tilde{p}$ into the marked word $\tilde{q}$, where $q = p - 1$, $|\tilde{q}| = |\tilde{p}|$. At the end, the word $\tilde{q}$ is considered as "given". One loop can be performed in $2|\tilde{p}|$ steps (see fig. (3)). Now let us perform successive loops, starting with the word $\tilde{s}$ (and the head scanning its leftmost symbol) and repeating them until the word $\widetilde{s'}$ is obtained with $s' = 0$. In this case the head is s squares far from its initial position and this number of 1's remains between its original and present position. The time spent equals $2|\tilde{s}| . s$ steps, which does not exceed $Ds \log s$, for $s \geqq 2^{|\tilde{s}|-1}$.

Let us return to the proof of (c). According to the lemma 3, $\widetilde{f(x)}$ can be printed on the tape within $C_f \overline{f(x)} \log \overline{f(x)}$ steps, and $\widetilde{g(x)}$ can be printed within another $C_g \overline{g(x)} \log \overline{g(x)}$ steps for suitable C_f and C_g. Both words can be supposed to have common right end and to be written on different tracks of the tape (see fig. 4). During the next $C \max\left(|\widetilde{f(x)}|, |\widetilde{g(x)}|\right)$ steps the word $\widetilde{f(x) + g(x)}$ can be obtained (simply

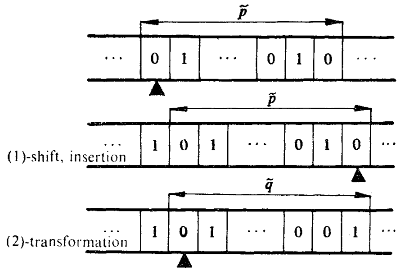
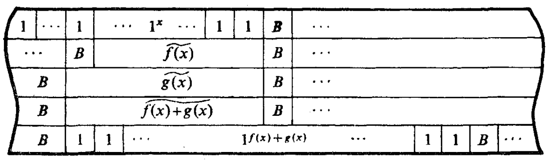


Fig. 3.

by means of the binary adding algorithm) and during next $D(f(x) + g(x))$ . . $\log(f(x) + g(x))$ steps, according to the lemma 4, the word $1^{f(x)+g(x)}$ results. (In the fig. 4 this word is printed only in one track of the tape.)



1 ··· 1 ··· 1^x ··· 1 1 B ···
··· B $\widetilde{f(x)}$ B ···
B $\widetilde{g(x)}$ B ···
B $\widetilde{f(x)+g(x)}$ B ···
B 1 1 ··· $1^{f(x)+g(x)}$ ··· 1 1 B ···

Fig. 4.

The total time spent by a machine following the outlined algorithms and computing thus the function $f(x) + g(x)$ does not exceed

$$C_f \overline{f(x)} \log \overline{f(x)} + C_g \overline{g(x)} \log \overline{g(x)} + C \max\left(|\widetilde{f(x)}|, |\widetilde{g(x)}|\right) + D(f(x) + g(x)) \log (f(x) + g(x)) \leqq \leqq (C_f + C_g + C + D)\left(\overline{f(x) + g(x)}\right) \log \left(\overline{f(x) + g(x)}\right), \tag{4}$$

because $\overline{f(x)} \leqq \overline{f(x) + g(x)}$, $\overline{g(x)} \leqq \overline{f(x) + g(x)}$, $|\widetilde{f(x)}| \leqq \overline{f(x)} \leqq \overline{f(x) + g(x)} \leqq \leqq \overline{(f(x) + g(x))} \log \overline{(f(x) + g(x))}$ and similarly for $|\widetilde{g(x)}|$, and $f(x) + g(x) \leqq \leqq \overline{f(x) + g(x)}$. Thus, $f(x) + g(x) \in \mathbf{L}$.

The proof for (d) proceeds analogously, it takes only a bit more time to obtain word $\widetilde{f(x)\,g(x)}$. The well known algorithm for binary multiplication is to be slightly adapted here: the words, resulting from successive shifts of one operand are added as soon as they are created because only a limited number of tracks of the tape is at our disposal (see fig. 5, where $f(x) = 11$, $g(x) = 46$). The needed time for it does not exceed $C|\widetilde{f(x)}|\,(|\widetilde{f(x)}| + |\widetilde{g(x)}|)$ (in the parantheses, the length of the product

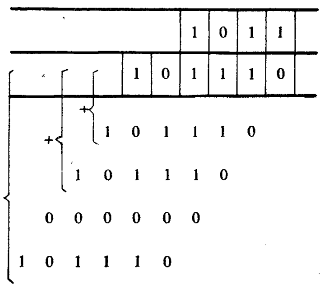


Fig. 5.

is expressed). Taking C_f, C_g and D in the sense of the previous part, the inequality (4) now converts to:

$$(5)\qquad C_f\overline{f(x)}\log\overline{f(x)} + C_g\overline{g(x)}\log\overline{g(x)} + C\,|\widetilde{f(x)}|\,(|\widetilde{f(x)}| + |\widetilde{g(x)}|) +$$
$$+ D(f(x)\,g(x)\log(f(x)\,g(x)) \leqq (C_f + C_g + C' + D)\overline{f(x)\,g(x)}\log\overline{f(x)\,g(x)}$$

for suitable C' because $f(x) \leqq \overline{f(x)\,g(x)}$ and $\overline{g(x)} \leqq \overline{f(x)\,g(x)}$ (here the assumption $f(x),\ g(x) \leqq f(x)\,g(x)$ is utilized); further $|\widetilde{f(x)}| \leqq C_1\log f(x) \leqq C_1\overline{f(x)}$, $|\widetilde{g(x)}| \leqq$ $\leqq C_2\log g(x)$ and hence $C|\widetilde{f(x)}|\,(|\widetilde{f(x)}| + |\widetilde{g(x)}|) \leqq CC_1\overline{f(x)}\,(C_1\log f(x) + C_2\,.$ $.\,\log g(x)) \leqq C'\overline{f(x)}\,(\log f(x) + \log g(x)) \leqq C'\overline{f(x)}\log(f(x)\,g(x)) \leqq C'\overline{f(x)\,g(x)}\,.$ $.\,\log\overline{f(x)\,g(x)}$ and finally, $f(x)\,g(x) \leqq \overline{f(x)\,g(x)}$.

A generalization of the idea involved in the proof of lemma 4 will enable us to prove the part (a) of the following theorem, which in turn shows that the class $\mathbf{L}$ is not exhausted by polynomials:

Theorem 3. (a) *If* $f(x),\ g(x) \in \mathbf{L}$ *and* $f(x) > 1$ *and* $g(x) \geqq 1$ *for all* x, *then* $f(x)^{g(x)} \in \mathbf{L}$;

(b) *if* $f(x),\ g(x) \in \mathbf{L}$ *and* $g(x) \geqq x$ *for every* x, *then* $g(f(x)) \in \mathbf{L}$.

Note. The requirement $f(x) > 1$ and $g(x) \geqq 1$ is (analogously as similar assumption in theorem 2) not too restrictive and it may be weakened again by assuming that it holds for almost all x.

Proof. (a) According to the lemma 3, $\widetilde{f(x)}$ can be printed within $t_1(x) \leqq C_1 \overline{f(x)}$. . $\log \overline{f(x)}$ and $\widetilde{g(x)}$ can be printed within $t_2(x) \leqq C_2\, \overline{g(x)} \log \overline{g(x)}$ for suitable C_1 and C_2. Together with $\widetilde{f(x)}$, the word $1^{|\widetilde{f(x)}|}$ can be easily generated and within another $t_3(x)$ steps the word $u = \widetilde{|\widetilde{f(x)}|}$ can be written down (on a suitable track

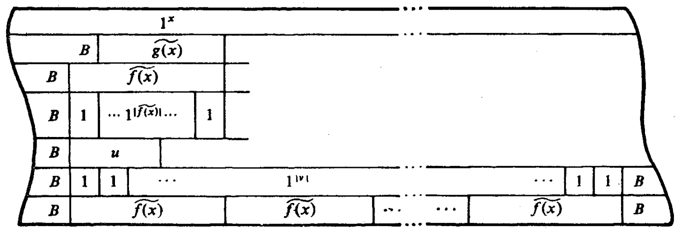


Fig. 6.

as seen from fig 6), where $t_3(x) \leqq C_3 |\widetilde{f(x)}| \log |\widetilde{f(x)}|$ for suitable C_3 according to lemma 2. The word u gives a binary representation of the length of the binary representation of the value $f(x)$. Our next task is to construct the word $v = \widetilde{f(x)}^{g(x)}$ (i.e. $\widetilde{f(x)}\widetilde{f(x)}, \dots, \widetilde{f(x)}, \widetilde{f(x)}$ concatenated $g(x)$-times), which represents the value $f(x)^{g(x)} - 1$ in a $f(x)$-adic system, the numerals of which are $00 \dots 1$, $00 \dots 10$, $00 \dots 11, \dots, \widetilde{f(x)}$ (these numerals – as words in the alphabet $\{0, 1\}$ – have the same length; we can suppose that their ends are suitably marked so that by a juxtaposition of them, a representation of a number in the $f(x)$-adic system is obtained). Let us find out the length of v: $|v| = |\widetilde{f(x)}| \,.\, g(x)$. According to theorem 2 part (d) (note that both $g(x)$ and $|\widetilde{f(x)}|$ belong to **L**, the first according to the assumption, the second due to the estimation of $t_3(x)$ above) the word $1^{|v|}$ can be obtained within $C'\, g(x)\, |\widetilde{f(x)}| \log(g(x) \,.\, |\widetilde{f(x)}|)$. As $|\widetilde{f(x)}| < C'' \log f(x)$, the word $1^{|v|}$ can be printed within $t_4(x) \leqq C_4\, g(x) \log f(x) \log [g(x) \log f(x)]$.

The word $1^{|v|}$ being established, v itself can be constructed by succesive adding of copies of the word $\widetilde{f(x)}$ to the right of the initially written $\widetilde{f(x)}$ (the process is repeated as far as the right end of $1^{|v|}$ is not reached). Because an arbitrary word can be copied in a time proportional to the square of its length, this process can be completed within $t_5(x) \leqq C'''|\widetilde{f(x)}|^2 . g(x) \leqq C_5\, g(x) \log^2 f(x)$ steps. By this construction of the word v, the first stage of the computation ends; let us call it preparatory stage (only this stage is pictured on fig. 6).

Now let a "loop" (cf. the proof of lemma 4) consist of the performance of the following two activities over two words: v and a given word p, which both are placed in the same columns of the tape (but, of course, in different tracks), both consisting of $g(x)$ adjacent *numerals*, i.e. words in $\{0, 1\}$ of the length $|\widetilde{f(x)}|$:

(1) shift of the words v and p one square right and inserting 1 in the released square (now in the whole square – auxiliary division into tracks vanishes for this square);

(2) transformation of the newly placed word p as follows: the numerals of p are succesively scanned from the right to the left; those which are of the form $00 \ldots 01$ are replaced by $\widetilde{f(x)}$ (which is, for this purpose, at disposal in the word written above p), while the first one from the right not of this form is considered as binary code of a number, which is in turn diminished by 1. Other numerals remain during the loop untouched. The resulting word is considered as "given". One loop can be performed in no more than $C^*|v| = C^*|\widetilde{f(x)}|\, g(x) \leqq C^{**}\, g(x) \log f(x)$ steps.

The proper computation (after the preparatory stage) consists now of succesive performances of loops, starting with $p = v$ and ending with $p = 00 \ldots 01^{g(x)}$. The described construction ensures that exactly $f(x)^{g(x)}$ steps will be performed and this number of 1's will be left on the tape (the other auxiliary 1's may be left, as they are written in some tracks of the tape only, and do not represent the "true" 1's). With respect to the estimation above, this part of computation can be accomplished within $t_6(x) \leqq C_6\, f(x)^{g(x)}\, g(x) \log f(x)$ steps.

The proof of the part (a) of our theorem will be concluded as soon as it will be shown that the total time spent, $\sum_{i=1}^{6} t_i(x) \leqq C\, \overline{f(x)^{g(x)}} \log \overline{(f(x)^{g(x)})} = C\, \varphi(x)$. This will be done by showing that for all $i = 1, \ldots, 6$: $t_i(x) \leqq C_i\, \varphi(x)$ and putting $C = \sum_{i=1}^{6} C_i$.

(1) $t_1(x) \leqq C_1 \overline{f(x)} \log \overline{f(x)} \leqq C_1\, \varphi(x)$ because $f(x) \leqq f(x)^{g(x)}$ (as $g(x)$ is supposed to be $\geqq 1$);

(2) $t_2(x) \leqq C_2\, \overline{g(x)} \log \overline{g(x)} \leqq C_2\, \varphi(x)$ because $g(x) \leqq f(x)^{g(x)}$ (as $f(x)$ is supposed to be $\geqq 2$);

(3) $t_3(x) \leqq C_3 |\widetilde{f(x)}| \log |\widetilde{f(x)}| \leqq C_3 \overline{f(x)} \log \overline{f(x)} \leqq C_3\, \varphi(x)$;

(4) $t_4(x) \leqq C_4\, g(x) \log f(x) \log [g(x) \log f(x)] \leqq C_4\, \varphi(x)$, because $g(x) \log f(x) \leqq$ $\leqq f(x)^{g(x)}$ (recall that the case $f(x) = 0$, which could be suspected of making troubles because we put $\log 0 = 1$, cannot appear, as $f(x) \geqq 2$);

(5) $t_5(x) \leqq C_5\, g(x) \log^2 f(x) = C_5 \log f(x) \log f(x)^{g(x)} \leqq C_5\, \varphi(x)$, because $\log f(x) \leqq f(x)^{g(x)}$;

(6) $t_6(x) \leqq C_6 f(x)^{g(x)} g(x) \log f(x) = C_6 f(x)^{g(x)} \log f(x)^{g(x)} \leqq C_6\, \varphi(x)$.

(b) According to lemma 3 and 4, $1^{f(x)}$ can be printed within $C_f\, \overline{f(x)} \log \overline{f(x)}$ steps. Put $y = f(x)$. Starting from 1^y, the word $1^{g(y)}$ can be similarly obtained within another $C_g\, \overline{g(y)} \log \overline{g(y)}$ steps. Let us compare $\overline{g(y)}$ with $\overline{g(f(x))}$ $(= g \circ (f(x))$; it is $\overline{g(y)} = \max(y, g(y), 1) = \max(f(x), g(f(x)), 1)$ while $\overline{g(f(x))} = \max(x, g(f(x)), 1)$. We see that in the assumed case, when

$$(6) \qquad g(f(x)) \geqq f(x)\,,$$

it is $\overline{g(y)} \leqq \overline{g(f(x))}$ and the total time spent does not exceed $C_f\, \overline{f(x)} \log \overline{f(x)} + C_g\, \overline{g(f(x))} \log \overline{g(f(x))}$, which in turn – due to (6) again – is less or equal to $2C_g \overline{g(f(x))} \log \overline{g(f(x))}$. This concludes the proof.

§ 3.

Now we give some negative results, especially by showing that the class **L** is not closed under substraction and composition. Let us stress that the additional assumptions required for the theorems 2, part (d) and (3), part (a) hold, are not substantial and could be removed by suitable changes in definitions of corresponding operations, which would not destroy the character of the operations. On the other hand, it does not seem possible to "save" in such a way the operations of substraction and nonrestricted composition (cf. theorem 3, part (b)); their anticlosure properties seem to be inherent.

We begin by applying the diagonal method to construct a function not belonging to **L**; its special form is utilized subsequently.

Theorem 4. *There exists a total computable function* $f(x)$, $f(N) = \{0, 1\}$, *such that* $f(x) \notin$ **L**.

Proof. Let $Z_1, Z_2, Z_3, \ldots$ be a sequence of all classical Turing machines and let

$$(7) \qquad Z'_1 = Z^1_1\,,\quad Z'_2 = Z^2_1\,,\quad Z'_3 = Z^1_2\,,\quad Z'_4 = Z^3_1\,,\quad Z'_5 = Z^2_2\,,\quad Z'_6 = Z^1_3, \ldots$$

represent an effective enumeration, in which every of the machines from the preceding sequence occurs infinite many times: Z_i as $Z^1_i, Z^2_i, Z^3_i, \ldots$

Consider a machine Z, which computes a function $f(x)$ in the following way: for a given j it produces (i.e. prints in a suitable track of its tape) the code (e.g. the set of quadruples) of the machine $Z'_j = Z^K_i$, which in turn computes some function $f_j(x)$. Z subsequently simulates the activity of Z'_j in its computation of the value $f_j(j)$ and this simulation lasts until

either (a) Z'_j would stop leaving on its tape (simulated of course by a suitable part of the tape of Z again) $f_j(j)$ symbols "1" or (b) the number of steps of Z_j' would exceed $Kj \log j$; this value is computed by Z beforehand and after the simulation of a step of Z_j' this number is decreased by 1 so that the comparison can be effectively performed.

In the case (a), Z finds out, whether $f_j(j) \neq 0$; if so, Z puts $f(j) = 0$ (i.e. erases all 1's from the tape); otherwise it puts $f(j) = 1$. Thus, $f(j) \neq f_j(j)$.

In the case (b) there are two subcases (which are, however, treated by Z in the same way):

either (ba) $f_j(x)$ is not computable within $K\overline{f_j(x)} \log \overline{f_j(x)}$ (this does not automatically imply $f_j \notin \mathbf{L}$ as f_j can be computable within $K'\overline{f_j(x)} \log \overline{f_j(x)}$ for suitable $K' > K$),

or (bb) $f_j(x)$ is computable within $K\overline{f_j(x)} \log \overline{f_j(x)}$, but in this case necessarily $\overline{f_j(x)} > j$ (for $K\overline{f_j(x)} \log \overline{f_j(x)}$ to be greater than $Kj \log j$). In both subcases, Z puts $f(j) = 0$, which ensures that $f(j) \neq f_j(j)$ (note that in the case (bb) $f_j(j) > j > 0$!).

The construction ensures $f(x) \notin \mathbf{L}$; indeed, every function from $\mathbf{L}$ is computed by a suitable machine (even by infinite number ones) from the list (7). But the function $f_j(x)$ computed by Z_j' differs from $f(x)$ at least in one point: $f(j) \neq f_j(j)$ for all j. By construction, $f(x) = 0$ or 1 for all x.

Utilizing the just proved result, the following one can be derived:

Theorem 5. *There exist functions* $f_1(x)$, $g_1(x)$, $f_2(x)$, $g_2(x) \in \mathbf{L}$ *such that*:

(a) $f_1(x) \dot{-} g_1(x) \notin \mathbf{L}$,

(b) $g_2(f_2(x)) \notin \mathbf{L}$,

$$(x \dot{-} y = x - y \text{ for } x \geqq y,\ x \dot{-} y = 0 \text{ for } x < y)\,.$$

Proof. Let us take a total computable function $h(x) \notin \mathbf{L}$, $h(x) = 0$ or 1 (theorem 4). Further, consider a function $H(x)$, which determines the number of steps, in which a fixed machine Z_h (computing h) will compute the value $h(x)$. $H(x)$ is of course computable as well. Consider the machine Z_H, which computes $H(x)$ in the following way: Z_H simulates the computation of Z_h in one of its tracks, but after every one simulated step it marks the position of the simulated head, performs some additional movement – let us call it a "loop" (to be described immediately) – and returns to the marked square to perform the simulation of the next step of Z_h. The mentioned additional movement enables to note the required number of steps of Z_h: in the loop in which the n-th step of Z_h is simulated, the head (of Z_H) runs to the right until the n-th square of the tape is reached and marked (the squares being counted from the zero square, which was scanned by the machine at the beginning); then it retuns to the left until the $(-n)$-th square is reached and marked, changes the direction again and finally comes to the marked square (see fig. 7; the symbols] and [mark the n-th and $(-n)$-th square, respectively, to enable proper behaviour of Z_H during the next loop). In this way the distance of the right (as well as of the left) marked square from the 0'th square determines the number of hitherto simulated steps of Z_h. Note that the simulated head always remains within the markers. The simul-

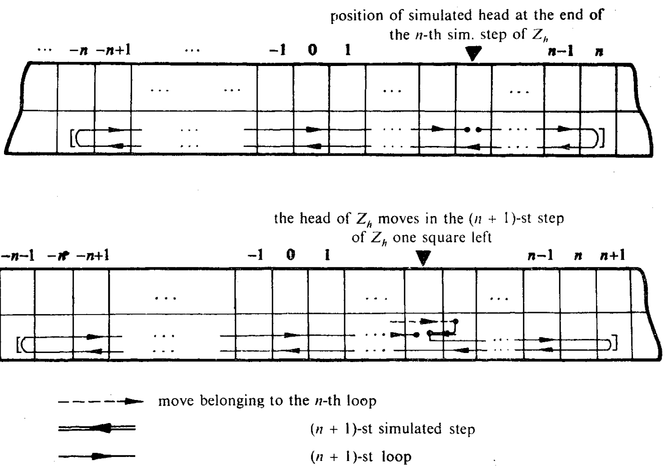

position of simulated head at the end of the n-th sim. step of Z_h
… $-n$ $-n+1$ … -1 0 1 $n-1$ n
the head of Z_h moves in the $(n+1)$-st step of Z_h one square left
$-n-1$ $-n$ $-n+1$ -1 0 1 $n-1$ n $n+1$
move belonging to the n-th loop
$(n+1)$-st simulated step
$(n+1)$-st loop

Fig. 7.

ation of the n-th step of Z_h requires $4n$ steps of the machine Z_H, so that during the simulation of first n steps of Z_h the number of steps performed by Z_H makes

$$\sum_{i=1}^{n} 4i = 2n(n+1)\,. \tag{8}$$

Consider now the function $g_1(x) = 2^{H(x)}$; we shall prove that $g_1 \in \mathbf{L}$ (note that theorem 3 (a) cannot be applied directly as it is not ensured $H(x) \in \mathbf{L}$). $g_1(x)$ can be computed in the following way: first, $1^{H(x)}$ is established, which requires $2H(x)\,.\,(H(x) + 1)$ steps according to (8). Having $1^{H(x)}$ written down on the tape, $g_1(x)$ can be computed – now according to theorem 3 – within next $C\,.\,2^{H(x)} \log 2^{H(x)}$ steps; thus the total time spent does not exceed $2H(x)\,(H(x) + 1) + C\,.\,2^{H(x)}\,.\,H(x) \log 2 \leqq C_g\,.\,2^{H(x)} \log 2^{H(x)}$ steps for suitable C_g, so that really $g_1(x) \in \mathbf{L}$ (g_1 grows so "rapidly" that the preparatory computation of $H(x)$ can not influence the fact $g_1(x) \in \mathbf{L}$ even that $H(x)$ itself need not belong to $\mathbf{L}$).

Consider further the function

$$f_1(x) = 2^{H(x)} + h(x)\,; \tag{9}$$

again, $f_1(x) \in \mathbf{L}$. f_1 is computed in the same manner as g_1 with the only exception that $h(x)$ (which can be easily remembered by the machine, as $h(x) = 0$ or 1) is added to the result. Evidently $f_1(x) \in \mathbf{L}$.

The proof of (a) is now at hand, as $f_1(x) \dot{-} g_1(x) = f_1(x) - g_1(x) = h(x) \notin \mathbf{L}$.

To prove the part (b), put $f_2(x) = f_1(x)$ and note that $f_2(x)$ is even (odd) if $h(x) = 0$ ($h(x) = 1$) (see (9)). Let further $g_2(x) = 0$ for x even and $g_2(x) = 1$ for x odd. Evidently, $g_2(x) \in \mathbf{L}$. But $g_2(f_2(x)) = h(x) \notin \mathbf{L}$, what was to be proved.

Acknowledgment. I wish to express my gratitude to P. Strnad for his helpful criticism.

(Received November 25th, 1968.)

REFERENCES

[D] Davis M.: Computability and Unsolvability, New York 1958.

[H] Hennie F. C.: One-tape, Off-line Turing computations. Inf. Control *8* (1965), 553—578.

[M] Myhill J.: Linear bounded automata. Wright Air Development Division Ohio, Report (1960), 60—22.

[T] Трахтенброт Б. А.: Тьюринговы вычисления с логарифмическим замедлением. Алгебра и логика *3* (1964), 33—48.

Rekurzivní funkce vyčíslitelné v $C\bar{f}\log\bar{f}$

JIŘÍ HOŘEJŠ

V práci je uvažován klasický model Turingova stroje pro výpočet funkce $f(x)$ jedné proměnné zobrazující množinu přirozených čísel do sebe (viz např. [D]). Hodnota argumentu x resp. funkční hodnota $f(x)$ je zapsána pomocí x resp. $f(x)$ jednotek. Vyšetřují se funkce, které mohou být vyčísleny vhodným Turingovým strojem uvažovaného typu během $C\overline{f(x)}\log\overline{f(x)}$ kroků stroje, kde $\overline{f(x)} = \max(x, f(x), 1)$. Ukazuje se, že třída **L** funkcí této vlastnosti je v jistém smyslu nejjednodušší třída funkcí ležících mimo schopnost konečných automatů (věta 1 a definice quasi-periodických funkcí), že však je již dosti bohatá; obsahuje např. polynomy, je uzavřená vzhledem k operacím součtu, součinu, umocňování, (věty 2, 3), ne však vzhledem k odčítání a superpozici (věta 5). Při důkazu pozitivních tvrzení se tu využívá lemat ukazujících na možnost poměrně „rychlého" převodu mezi g-adickými zápisy čísel, negativní tvrzení se opírají o metodu přechodových posloupností (crossing sequence z [H] resp. „sled" z [T]) a diagonální metodu.

Doc. RNDr Jiří Hořejš, CSc., Katedra matematických strojů University J. E. Purkyně, Janáčkovo nám. 2a, Brno.