KYBERNETIKA ČÍSLO 4, ROČNÍK 5/1969

On Optimum Time Bounds for Recognition of Some Sets of Words by On-line Turing Machines

PAVEL STRNAD

In the paper optimum time bounds (up to a multiplicative constant) on recognition of some sets of words by on-line Turing machines are given. The activity of Turing machines which recognize the sets within these time bounds is described.

INTRODUCTION

In this paper we consider multitape on-line Turing machines in the sense of [1]. An on-line Turing machine begins its computation in a fixed state s_0 with all tapes completely empty. (The input symbols of the machine form the input alphabet, the output symbols form the output alphabet and symbols used on tapes form the internal alphabet of the considered machine.)

The machine produces an output symbol as an answer to an input symbol. This answer need not be immediate, it may come after several steps of the activity of the machine. The machine does not accept any further input symbol during that activity (during the elaboration of an answer). The $(i + 1)$-st input symbol is accepted by the machine at the time which follows the time of the production of the i-th output symbol.

If the output alphabet of the machine is $\{0, 1\}$, the machine may alternatively be thought of as recognizing the set of those input sequences for which the last symbol in the corresponding output sequence is a 1.

In the whole paper the term "machine" always means an on-line Turing machine.

The activity at a given step of the computation of the machine depends on the symbols read by the heads on the tapes, on the state of the control unit of the machine and on the input symbol (at the steps at which an input symbol is accepted). At every step of its activity the machine can rewrite the symbols under the heads on the tapes, shift every of its heads one square to the right or to the left, change its state and produce an output symbol. There is one head on every tape.

The machine M recognizes (represents) a set A of words over an alphabet X if

1. the input alphabet of the machine M is $Y \supset X$ and the output alphabet of M is $\{0, 1\}$;

2. for every word w over the alphabet Y it holds:
$w \in A$ if and only if the machine M assigns to the input word w an output word of the length $|w|$ and the last symbol of that output word is a 1. ($|w|$ is the length of the word w.)

Let us denote $\tau_M(w)$ the time (i.e. the number of steps) of the processing of the input word w by the machine M and let $\tau_M(n) = \max_{|w|=n} \tau_M(w)$. We call $\tau_M(n)$ the computation time of the machine M.

Definition. Let $T(n)$ be an arithmetic function. We say that a set B is $T(n)$-recognizable if the following conditions are satisfied:

1. There is a multitape on-line Turing machine X and a constant C such that

(a) X recognizes B,
(b) $\tau_X(n) < C\,T(n)$ for all n.

2. For any on-line Turing machine Y which recognizes B there is a constant K such that for almost all values of n the inequality $\tau_Y(n) > K\,T(n)$ holds.

Remark. The constant K from the part (b) of the definition depends on the machine Y (on the number of its tapes and on the number of the symbols of its internal alphabet). In concrete cases it is possible to decrease this constant by an increase of the number of the tapes or by an increase of the number of the symbols of the internal alphabet.

1. TIME BOUNDS ON THE RECOGNITION OF THE SET A

Let us consider the alphabet $\{0, 1, *\}$. A word over this alphabet of the length $i + 1$ which consists of 0's and 1's and which is terminated by * will be called an i-block ($i = 1, 2, \ldots$). A sequence of 2^i i-blocks will be called an i-base.

Let us define the sets $A^{(i)}$: $w \in A^{(i)}$ if and only if there exists $k \geqq 1$ such that w is a sequence of $2^i + k$ i-blocks and the last i-block of the word w is equal to one of the initial 2^i i-blocks of the word w.

Then $A = \bigcup_{j=1}^{\infty} A^{(j)}$ is the set A from [1].

Hennie proved [1] that for any on-line Turing machine M that recognizes the set A the inequality $\tau_M(n) > n^2/(48t \log s \log^2 n)$ holds for almost all values of n (t is the number of tapes and s is the number of symbols of the internal alphabet of the machine M). Throughout this paper log means $\log_2$.

First we shall describe a machine N which recognizes the set A and for which the inequality $\tau_N(n) < C \,.\, n^2/\log^2 n$ holds for each n, where C is an appropriate constant.

(Hennie [1] states the possibility of a recognition of the set A by a machine using that computation time, but without a detailed proof.) As a consequence, the set A is $n^2/\log^2 n$-recognizable (according to our definition of $T(n)$-recognizability).

The description of the activity of the machine N. The machine N has 8 tapes. The input alphabet of the machine N is $\{0, 1, *\}$. The activity of the machine N is divided into several stages.

The first stage begins with accepting the first input. Within this stage

(a) the machine determines the length of the first block and marks this length on the tape T_4. In this way the machine obtains the information to which set $A^{(i)}$ the given word could belong. During its further activity the machine still compares the lengths of the blocks of the input word with the length of the first block. If some block has a different length the machine will give out a 0 as an answer to every subsequent input symbol.

(b) As soon as the machine finds out the length $i + 1$ of the first block, it begins to mark, on the two tapes T_5 and T_6, groups of $1, 2, 4, \ldots, 2^{i-1}$ squares. The machine simultaneously transfers these groups onto the tapes T_2 and T_3. It is possible to do this within $\sum_{t=0}^{i-1} 2^t = 2^i - 1$ steps, in the following way: Let 2^j squares on the tape T_5 and groups of $1, 2, 4, \ldots, 2^j$ squares on the tapes T_2 and T_3 be already marked. The head on the tape T_5 goes from the left end of the group of 2^j marked squares to the right end of that group and then returns to its initial position. That activity takes $2 \cdot 2^j = 2^{j+1}$ steps; simultaneously, the head on each of the tapes T_2, T_3, T_6 marks a group of 2^{j+1} squares. Afterwards the heads on the tapes T_5 and T_6 interchange their roles and the head on the tape T_6 passes twice through the group of 2^{j+1} marked squares; the heads on the tapes T_2, T_3, T_5 simultaneously mark groups of the 2^{j+2} squares, e.t.c. The total number of groups which are marked on the tapes T_2, T_3 is i, the machine compares it with the record on the tape T_4. On the tape T_2 the groups of marked squares will be separated (the last symbol of every group will be indicated), on the tape T_3 the groups will not be separated. The length of the record on each of the two tapes T_2 and T_3 is $2^i - 1$.

(c) From the beginning of its activity the machine successively writes down on the tape T_7 input symbols. On the tape T_8 the machine marks the number of the accepted and recorded blocks. We shall consider only the case $i \geqq 2$ because $A^{(1)}$ can be recognized by a finite automaton.

(d) As soon as the machine finishes the activity described in (b), it continues working in this way: it stops writing on the tape T_8 and, on the tape T_3 it successively erases one symbol for every further accepted block. Simultaneously, the machine erases on the tape T_3 the same number of symbols as are written down on the tape T_8. As soon as the head on the tape T_3 erases all its record, $2^i - 1$ blocks are accepted. The machine then accepts two blocks more and writes them down on the tape T_7. Up to this moment the machine has accepted $2^i + 1$ blocks, i.e. $(2^i + 1)(i + 1)$ input

symbols and has written them on the tape T_7. The machine has produced $2^i(i+1)+i$ output symbols 0's. (During the first stage of its activity the machine receives an input symbol at each step of the computation, the output symbol being all the time 0.)

The second stage begins with the $((2^i+1)(i+1)+1)$-st step of the computation. During this stage the machine will not receive any input symbol and will not give out any output symbol. During the second stage the machine encodes the blocks written on the tape T_7. First the head on the tape T_7 returns to the begin of its record. The head on the tape T_1 begins its operation on a marked square of the tape and encodes successively 2^i i-blocks from the tape T_7 as follows: if the first symbol of the block is 1, the head makes a shift 1 square to the right; if on the second place of the block there is 1, the head makes a shift 2 squares to the right, ...; if on the t-th place of the block there is 1, the head makes a shift 2^{t-1} squares to the right. If on the t-th place of the block there is 0, the head keeps its position (i.e. no shift is done). The shifts 2^{k-1} $(k = 1, 2, \ldots, i)$ squares to the right are realized using the groups recorded on the tape T_2. As soon as in the process of encoding of the block the head on the tape T_7 comes to a square on which * is written, the head on the tape T_1 marks its position (by a special symbol of the internal alphabet) and then returns to the left to the marked beginning square. Simultaneously the head on the tape T_2 returns to its starting square and the head on the tape T_7 begins to process symbols of the next block and the whole procedure is repeated until 2^i i-blocks are coded. To two different blocks there correspond two different total shifts to the right, the coding is one-to-one. No shift (i.e., the beginning square) is assigned to the i-block $\underbrace{00\ \ldots\ 00}_{i}*$ the maximum shift is $2^i - 1$ squares and it corresponds to the i-block $\underbrace{11\ \ldots\ 11}_{i}*$.

After having coded the i-base (the initial 2^i i-blocks) the machine controls the (2^i+1)st block (i.e., it verifies whether this block is equal to any block from the base or not). The head on the tape T_1 moves similarly as when coding the block but now it does not mark the square but finds out whether it is yet marked or not; the head then returns to the beginning square and the machine gives out the output symbol 1 (or 0). By this last step of computation the third stage begins.

The third stage. To the input the symbols of the next blocks come and the head on the tape T_1 codes them again by shifts to the right. In the step in which the input symbol is * the machine finds out whether the square under the head on the tape T_1 is marked or not, the head on the tape T_1 returns to the beginning square and the machine gives out the output symbol 1 (or 0).

The Estimation of the Computation Time of the Machine N

The words from $A^{(1)}$ are recognizable by a finite automaton, consequently we shall consider the set $\bigcup_{j=2}^{\infty} A^{(j)}$ only. If $i+1$ is the length of the first block received by the

machine, then the processing of an input word of the length $n < (i + 1)(2^i + 1)$ takes n steps. Let the input word be of the length $n \geqq (i + 1)(2^i + 1)$. Then there exists an integer k such that the inequalities $(i + 1)(2^i + k) \leqq n < (i + 1)(2^i + k + 1)$ hold. Then for the computation time of the machine N we have:

$\tau_N(n) < i + 1$ ... this number of steps corresponds to the coding of the length of the first block of the input word on the tape T_4;

$+ 2^i - 1$ ... the realization of the record on the tapes T_2 and T_3;

$+(2^i + 1)(i + 1)$ ... the record of the initial $2^i + 1$ i-blocks on the tape T_7;

$+2^i(2 \cdot 2^i)$ ... the coding of the initial 2^i i-blocks on the tape T_1;

$+(k + 1)(2 \cdot 2^i)$ ... the coding of the next $k + 1$ i-blocks on the tape T_1.

Therefore (for $i \geqq 2$)

$$\tau_N(n) < 2 \cdot 2^i \cdot (2^i + k + 1) + 3 \cdot i \cdot 2^i \leqq 4 \cdot 2^i \cdot (2^i + k) =$$
$$= 4 \cdot 2^i \cdot (i + 1)(2^i + k)/(i + 1) .$$

If we compare the bounds for n we see that

$$\tau_N(n) < 4 \cdot 2^i \cdot n/(i + 1) < 4 \cdot n \cdot 2^i/i .$$

To each given value of n it is possible to find an integer m such that the inequalities $(m + 1) \cdot 2^m \leqq n < (m + 2) \cdot 2^{m+1}$ hold. Surely it will be $m \geqq 1$. Further $\log n < \log(m + 2) + m + 1 < 3m$, because $m \geqq i \geqq 2$; consequently $1/m < 3/\log n$. As the function $2^i/i$ is increasing for $i \geqq 2$, the proved inequalities yield

$$2^i/i \leqq 2^m/m < 3 \cdot 2^m/\log n < 3 \cdot (m + 1) \cdot 2^m/(m \log n) < 9n/\log^2 n .$$

Therefore $\tau_N(n) < 4n \cdot 9n/\log^2 n = C \cdot n^2/\log^2 n$ holds.

As a consequence of the Hennie's estimation and the construction of the machine N just described, the set A is $n^2/\log^2 n$ – recognizable (see Definition).

2. TIME BOUNDS FOR RECOGNITION OF SOME SUBSETS OF THE SET A

For simplicity of writing, let us introduce the denotations $\log^{(k)} x$ and $\exp^{(k)} x$ (where k is a nonnegative integer) in the following sense:

$$\log^{(0)} x = \exp^{(0)} x = x ,$$
$$\log^{(k)} x = \log_2(\log^{(k-1)} x) , \quad \exp^{(k)} x = 2^{\exp^{(k-1)} x} \quad \text{for} \quad k \geqq 1 .$$

I

In this chapter we shall determine-similarly as for the set A – for some other sets the time which is necessary and sufficient (up to a multiplicative constant) for them to be recognized by an on-line Turing machine. So we obtain a hierarchy of sets which have different time complexity. These sets are subsets of the set A. Each of them is a set of words all the i-blocks of which fulfil some restrictive condition. If r is a real number then $[r]$ denotes an integer such that $[r] \leqq r < [r] + 1$. The sets are as follows:

$A_{p/q}$ – the beginning $[(q - p)\, i/q]$ symbols of an i-block of a word from $A_{p/q}$ are 0's, the other symbols are arbitrary, i.e. 0's or 1's; p, q are positive integers, $0 < p/q \leqq 1$;

$A^k_{u,v}$ – the number of 1's of each i-block of a word from $A^k_{u,v}$ is just $u + 1$ and u of 1's are on the places (within the block) of the type $\exp^{(k)} p^v$, where $p = 1, 2, \ldots, [(\log^{(k)} i)^{1/v}]$. The set $A^k_{u,v}$ is defined for $k = 0, 1, 2, \ldots$; $u = 0, 1, 2, \ldots$; $v = 1, 2, 3, \ldots$; $v + k > 1$.

Theorem 1. *The set $A_{p/q}$ is $(n/\log n)^{1+p/q}$-recognizable.*

Remark. With respect to the definition of $T(n)$-recognizability the statement of Theorem means the following:

(a) If an on-line Turing machine M recognizes the set $A_{p/q}$, then for almost all values of n the inequality $\tau_M(n) > c_M(n/\log n)^{1+p/q}$ holds, where C_M is a constant depending on the number of tapes and on the number of symbols of the internal alphabet of the machine M.

(b) There exists a constant C and an on-line Turing machine $\overline{N}$ which recognizes the set $A_{p/q}$ and for which the inequality $\tau_N(n) < C(n/\log n)^{1+p/q}$ holds for each n.

Proof. (a) This part of the proof is similar to the Hennie's proof for the set A [1]. We shall consider two i-bases as "different" if and only if there exists an i-block which is a part of one i-base and is not a part the other. It is possible to choose $2^{2^{i-[(q-p)i/q]}} - 1$ pairwise different i-bases from $A_{p/q}$. Denote by a_j $(j = 1, 2, \ldots, 2^{2^{i-[(q-p)i/q]}} - 1)$ the i-bases which correspond to a certain fixed choice of pairwise different i-bases. Further let b_j $(j = 1, 2, \ldots, 2^{2^{i-[(q-p)i/q]}} - 1)$ be arbitrary i-words (an i-word is a word consisting of a finite number $k = 0, 1, 2, \ldots$ of i-blocks) those consist of i-blocks, satisfying the condition imposed on the i-blocks in the set $A_{p/q}$. First we prove two lemmas.

Lemma 1. *If an on-line Turing machine L recognizes the set $A_{p/q}$, then for each $i \geqq 1$ there exists an integer j $(1 \leqq j < 2^{2^{i-[(q-p)i/q]}})$, and an i-block d so that at processing the word $a_j b_j d$ the machine L needs a time (the number of steps) larger than v_i for the processing of the last i-block d, where*

$$v_i = (2^{i-[(q-p)i/q]} - \log Q - t \log s - 1)/(2t \log s)\,;$$

here Q is the number of internal states, t is the number of tapes, s is the number of symbols of the internal alphabet of the machine L.

Proof. 1. Let $Qs^t > 2^{2^{i-[(q-p)i/q]-1}}$. Then $\log Q + t \log s > 2^{i-[(q-p)i/q]} - 1$, therefore $2^{i-[(q-p)i/q]} - \log Q - t \log s - 1 < 0$. It follows that $(2^{i-[(q-p)i/q]} - \log Q - t \log s - 1)/(2t \log s) < 0$.

2. Let $Qs^t > 2^{2^{i-[(q-p)i/q]-1}}$. We introduce the concept of an x-configuration of the given Turing machine. This x-configuration includes the machine state, the symbol under the head on each tape and x symbols to the left and to the right from it (altogether $2x + 1$ symbols on each tape). Therefore there are at most $Qs^{t(2x+1)}$ different x-configurations of a given machine. Let x_i be the greatest integer satisfying $Qs^{t(2x_i+1)} \leqq 2^{2^{i-[(q-p)i/q]-1}} < 2^{2^{i-[(q-p)i/q]}} - 1$. Certainly $x_i \geqq 0$, by assumption 2. Passing to logarithms we obtain $\log Q + 2x_i t \log s + t \log s \leqq 2^{i-[(q-p)i/q]} - 1$. From the last inequality and from the fact that x_i was the greatest integer j satisfying the inequality $Qs^{t(2j+1)} \leqq 2^{2^{i-[(q-p)i/q]-1}}$, the inequality $0 \leqq x_i \leqq (2^{i-[(q-p)i/q]} - \log Q - t \log s - 1)/(2t \log s) < x_i + 1$ follows. As $Qs^{t(2x_i+1)} < 2^{2^{i-[(q-p)i/q]}} - 1$, the number of i-words $a_j b_j$ (for $j = 1, 2, \ldots, 2^{2^{i-[(q-p)i/q]}} - 1$) is greater than the number of different x_i-configurations of the considered machine. Thus there exist two i-words $a_r b_r$ and $a_s b_s$ (for $r \neq s$) such that for the input words $a_r b_r$ and $a_s b_s$ the x_i-configurations of the machine are in the both cases equal at the time at which the last output symbol is given out. But the bases a_r, a_s are different, thus there exists an i-block d which is a part of one and is not a part of the other base. Assume that the time of processing the block d is shorter than $x_i + 1$ steps, i.e., that during processing the block d by the machine no head leaves the squares of the initial x_i-configuration. The machine then works in the same way when processing the last i-block d of the input words $a_r b_r d$, $a_s b_s d$. Thus it either receives both of the words $a_r b_r d$, $a_s b_s d$ or rejects them both simultaneously. This is a contradiction, because the i-block d is a part of one of the bases a_r, a_s and is not a part of the other. Thus the time of processing the i-block d of the input word $a_r b_r d$ (and simultaneously also the time of processing the i-block d of the input word $a_s b_s d$) is greater than or equal to $x_i + 1 > v_i$.

Q.E.D.

Lemma 2. *If an on-line Turing machine L recognizes the set $A_{p/q}$ and a_j $(j = 1, 2, \ldots, 2^{2^{i-[(q-p)i/q]}} - 1)$ are pairwise different i-bases then there exist a positive integer $k < 2^{2^{i-[(q-p)i/q]}}$ and an i-word h such that*

(a) *the i-word h consists of 2^i i-blocks satisfying the condition imposed on the i-blocks in the set $A_{p/q}$.*

(b) *In the course of the processing of the word $a_k h$ by the machine L the time of processing of each of the 2^i i-blocks of the word h is larger than v_i* (see Lemma 1).

Proof. Let us have pairwise different i-bases a_j $(j = 1, 2, \ldots, 2^{2^{i-[(q-p)i/q]}} - 1)$.

Then according to Lemma 1 there exist positive integers r, s and an i-block d_1 so that at processing the words $a_r d_1$, $a_s d_1$ the processing of the last i-block d_1 lasts longer than v_i. Let us take i-words a'_j $(j = 1, 2, \ldots, 2^{2^{i-[(q-p)i/q]}} - 1)$, where $a'_r = a_r d_1$, $a'_s = a_s d_1$ and $a'_j = a_j$ for all other values of j, and repeat Lemma 1 with them. We obtain the same number of i-words a''_j again, etc. The whole procedure will be finished when we obtain a group of i-words, at least one of which has the length $2^{i+1}(i+1)$; i.e., it consists of 2^{i+1} i-blocks. The i-word thus obtained evidently has the properties stated in Lemma 2.

Q.E.D.

Now we return to the proof of the part (a) of Theorem 1.

Let us denote $n_j = 2^{j+1}(j+1)$. For each positive integer n there exists a positive integer i such that $n_i = 2^{i+1}(i+1) \leqq n < 2^{i+2}(i+2) = n_{i+1}$. Then $\tau(n) \geqq \tau(n_i)$ and according to Lemma 2

$$\tau(n) \geqq \tau(n_i) > 2^i \cdot v_i = 2^i(2^{i-[(q-p)i/q]} - \log Q - t \log s - 1)/(2t \log s);$$

Q, t, s are constants, $p/q > 0$, then

$$\tau(n) > 2^i \cdot 2^{i-[(q-p)i/q]}/(3t \log s) \geqq 2^{2i-(q-p)i/q}/(3t \log s)$$

for large values of i, since $2^{i-[(q-p)i/q]} \geqq 2^{i-(p-q)i/q}$.

It is easy to change this last inequality into

$$\tau(n) > 2^{i(1+p/q)}/(3t \log s) = (2^{i+2}(i+2)/(2^2 \cdot (i+2)))^{1+p/q}/(3t \log s);$$

then

$$\tau(n) > n^{1+p/q}/((2^2(i+2))^{1+p/q} \cdot 3t \log s) \geqq (n/(i+2))^{1+p/q}/(48t \log s).$$

Passing to logarithms of the bounds within which n lies we obtain $i + 1 + \log(i+1) \leqq \log n$, thus $i+2 \leqq \log n$ for almost all values of i. Altogether for almost all values of n the inequality $\tau(n) > c_M(n/\log n)^{1+p/q}$ holds, where $C_M = 1/(48t \log s)$.

Q.E.D.

The proof of the part (b) depends on the description of the activity of the machine $\overline{N}$. The activity of the machine $\overline{N}$ is similar to the activity of the machine N recognizing the set A (see § 1), so we describe the activity of $\overline{N}$ only briefly. In the first stage of the activity the machine first marks the length of the first block. During its whole activity the machine compares the lengths of the blocks with the length of the first block. On one of the tapes the machine writes down symbols from the input; altogether in the first stage the machine records $2^i + 1$ i-blocks (i.e. it writes down $(2^i + 1) \cdot (i+1)$ symbols). During its activity in the first stage the machine produces $2^i(i+1) + i$ output symbols 0. The machine computes the number $[(1 - p/q)\, i]$; the initial $[(1 - p/q)\, i]$ symbols of every i-block of a word from $A_{p/q}$ must be 0's

and with every accepted i-block the machine verifies this condition. In the second stage the machine codes 2^i i-blocks of the base. The number of squares needed for coding an arbitrary i-block of a word from $A_{p/q}$ is $2^{i-[(1-p/q)i]}$, the coding is similar as in the machine N. In the third stage of the activity of the machine $\bar{N}$ the activity is similar to the activity of the machine N from § 1 again.

The estimation of the computation time of the machine $\bar{N}$

In the activity of the machine $\bar{N}$ there are only two essential differences in comparison with the activity of the machine N. The first is the computation of $[(1 - p/q)\, i]$. This computation can be realized in Ti steps, where T is a constant. For almost all values of i it is possible to realize this computation in the first stage of the activity of the machine. The second difference concerns the coding (and the control) of i-blocks. To the coding of the i-blocks of words in $A_{p/q}$, $2^{i-[(1-p/q)i]}$ squares are sufficient on one of the tapes. Then for the computation time of the machine $\bar{N}$ the inequality

$$\tau_{\bar{N}}(n) < i + 1 + 2^i - 1 + (2^i + 1)(i + 1) + 2^i(2 \,.\, 2^{i-[(1-p/q)i]}) + \\ + (k + 1)(2 \,.\, 2^{i-[(1-p/q)i]},$$

holds (the relations between n, i, k are introduced in § 1, where the estimation of the computation time of the machine N was made). By similar arrangements as in § 1 we have $\tau_{\bar{N}}(n) < 8n \,.\, 2^{(p/q)i}/i$ and further $\tau_{\bar{N}}(n) < C(n/\log n)^{1+p/q}$. By this, the proof of Theorem 1 is finished.

Theorem 2. *The set $A^k_{u,v}$ is $n(\log^{(k+1)} n)^{u/v}$-recognizable.*

Proof. See Remark after Theorem 1; we again divide the proof into two parts.

(a) This part of the proof is similar to the proof of the part (a) of the Theorem 1. Let us denote α the number of different i-blocks of the word from $A^k_{u,v}$. Then

$$\alpha = (i - [(\log^{(k)} i)^{1/v}]) \binom{[(\log^{(k)} i)^{1/v}]}{u}.$$

It is possible to choose $2^\alpha - 1$ pairwise different i-bases from $A^k_{u,v}$. The two lemmas from the proof of Theorem 1 can be formulated in this way:

Lemma 1′. *If an on-line Turing machine L recognizes the set $A^k_{u,v}$ then for each $i \geqq 1$ there exist an integer j $(1 \leqq j < 2^\alpha)$ and an i-block d so that at processing the word $a_j b_j d$ the machine L needs for the processing of the last i-block d a time (the number of steps) larger than v_i, where*

$$v_i = (\alpha - \log Q - t \log s - 1)/(2t \log s);$$

here Q is the number of internal states, t is the number of tapes, s is the number of symbols of the internal alphabet of the machine L.

Lemma 2'. *If an on-line Turing machine L recognizes the set $A^k_{u,v}$ and a_j $(j = 1, 2, \ldots, 2^z - 1)$ are pairwise different i-bases then there exist a positive integer $k < 2^z$ and an i-word h so that*

1. *the i-word h consists of 2^i i-blocks satisfying the condition imposed on the i-blocks in the set $A^k_{u,v}$.*
2. *In the course of the processing of the word $a_k h$ by the machine L the time of processing of each of the 2^i i-blocks of the word h is larger than v_i* (see Lemma 1').

The proofs of these two lemmas are similar to the proofs of the lemmas from the proof of Theorem 1; we shall not repeat them. Now we come to the proof of the part (a) of Theorem 2. Let us denote $n_j = 2^{j+1}(j+1)$. For each positive integer n there exists a positive integer i such that $n_i = 2^{i+1}(i+1) \leqq n < 2^{i+2}(i+2) = n_{i+1}$ holds. Then $\tau(n) \geqq \tau(n_i)$ and, according to Lemma 2',

$$\tau(n) \geqq \tau(n_i) > 2^i v_i = 2^i(\alpha - \log Q - t \log s - 1)/(2t \log s) .$$

Q, t, s are constants, $v + k > 1$, therefore for large values of i the inequality $\tau(n) > 2^i \cdot \alpha/(3t \log s)$ holds. Further, for large values of i the inequality $\alpha > i(\log^{(k)} i)^{u/v} : (2^{u+1}(u!))$ holds, consequently $\tau(n) > i \cdot 2^i(\log^{(k)} i)^{u/v}/(3.2^{u+1} \cdot t(u!) \log s)$. For large values of i also the inequality $i > (\frac{1}{3}) \log n$ holds, thus the last inequality can be rewritten in the form $\tau(n) > K_L\, n(\log^{(k+1)} n)^{u/v}$, where the constant K_L depends on the machine L.

(b) This part of the proof of Theorem 2 consists in the description of the activity of a concrete machine that recognizes the set $A^k_{u,v}$ and on the estimation of its computation time. Let us divide the activity of this machine into stages, similarly as we did when describing the activity of the machine N that recognizes the set A.

In the first stage of its activity the machine writes down $2^i + 1$ i-blocks (i.e. $(2^i + 1)(i + 1)$ input symbols) on one of the tapes and gives out $2^i(i+1) + i$ output symbols 0. Further, in the first stage, the machine determines, within an i-block, the places of the type $\exp^{(k)} p^v$ for $p = 1, 2, \ldots, [(\log^{(k)} i)^{1/v}]$. With every i-block, the machine finds out whether just u of 1's are on these places and whether in the whole i-block just $u + 1$ of 1's are contained. In the first stage, on one of the tapes, the machine marks $(i - [(\log^{(k)} i)^{1/v}])$ groups of squares, the groups being pairwise separated by auxiliary symbols. Each from these groups contains $\binom{[(\log^{(k)} i)^{1/v}]}{u}$ squares.

In the second and the third stage the machine codes i-blocks. The machine codes every i-block of a word from the set $A^k_{u,v}$ on the tape on which the groups are made. This is done as follows. The 1 (just one) which is not on a place of the type $\exp^{(k)} p^v$

can be on the $i - [(\log^{(k)} i)^{1/v}]$ places. In accordance with the position of this 1 the machine chooses one from the $i - [(\log^{(k)} i)^{1/v}]$ groups. Within this group of squares it is possible to code an arbitrary u-tuple of 1's, which are positioned on places of the type $\exp^{(k)} p^v$, because this group contains $\binom{[(\log^{(k)} i)^{1/v}]}{u}$ squares. Let us bring the head on the left-most square of this group. Let us consider the $(\exp^{(k)} 1^v)$-th symbol of the i-block. If it is a 0, the head shifts $\binom{[(\log^{(k)} i)^{1/v}] - 1}{u - 1}$ squares to the right, if it is a 1, the head stays on its position (i.e. no shift is done). Then the machine considers the $(\exp^{(k)} 2^v)$-th symbol, etc. If on the $(\exp^{(k)} j^v)$-th place within the considered i-block there is a 1, the head stays on its position. If on this place there is a 0 and on the places $\exp^{(k)} x^v$ (for $x < j$) there are altogether y of 1's ($y \leqq u - 1$), then the head shifts $\binom{[(\log^{(k)} i)^{1/v}] - j}{u - y - 1}$ squares to the right. If on the places of the type $\exp^{(k)} x^v$ (for $x < j$) there are altogether u of 1's, then the head stays on its position. In this way it is possible to assign unambiguously to every i-block of a word from $A^k_{u,v}$ a certain square; the machine marks this square. When the coding of the base is finished the machine verifies whether the following individual i-blocks are equal to any block of the base or not. The shifts of $\binom{a}{b}$ squares (a, b are positive integers, $a < [(\log^{(k)} i)^{1/v}]$, $b < u$) to the right are realized by using the record that the machine made during the first stage. The time needed for production of these records is shorter than $\binom{[(\log^{(k)} i)^{1/v}] + 1}{u}$ for almost all values of i. The coding of an arbitrary i-block lasts at most $2(i - [(\log^{(k)} i)^{1/v}]) . \binom{[(\log^{(k)} i)^{1/v}]}{u}$ time units. The finding of the places of the type $\exp^{(k)} p^v$ (for $p = 1, 2, \ldots, [(\log^{(k)} i)^{1/v}]$) lasts at most $(k + 1)\, i$ time units. The realization of the system of groups of squares lasts at most $(i - [(\log^{(k)} i)^{1/v}]) . \binom{[(\log^{(k)} i)^{1/v}]}{u}$ time units.

The estimation of the computation time of the machine R

If $i + 1$ is the length of the first block accepted by the machine R then the processing of the input word of the length $n < (i + 1)(2^i + 1)$ lasts n time units. Let the input word be of the length $n \geqq (i + 1)(2^i + 1)$. Then there exists an integer m such that $(i + 1)(2^i + m) \leqq n < (i + 1)(2^i + m + 1)$. According to the description of the activity of the machine R for its computation time we have:

$\tau_R(n) < (2^i + 1)(i + 1)$ ... this number of steps corresponds to the recording of the initial $2^i + 1$ i-blocks;

$+(k+1)\,i$ ... the finding (for the given value of i) of all places of the type $\exp^{(k)} p^v$ for $p = 1, 2, \ldots, [(\log^{(k)} i)^{1/v}]$;

$+\binom{[(\log^{(k)} i)^{1/v}]+1}{u}$ ... the unary coding of the possible shifts by the coding of i-blocks;

$+\binom{[(\log^{(k)} i)^{1/v}]}{u}(i - [(\log^{(k)} i)^{1/v}])$

... the realization of the system of the groups;

$+2(i - [(\log^{(k)} i)^{1/v}])\binom{[(\log^{(k)} i)^{1/v}]}{u} \cdot 2^i$

... the coding of 2^i i-blocks of the base;

$+2(i - [(\log^{(k)} i)^{1/v}])\binom{[(\log^{(k)} i)^{1/v}]}{u}(m+1)$

... "control" (i.e., the verification whether the given i-block is equal to any block of the base or not) of the following $m+1$ i-blocks.

Altogether, we have the inequality

$$\tau_R(n) < (2^i + k + 2)(i+1) + \binom{[(\log^{(k)} i)^{1/v}]+1}{u} + 2(i - [(\log^{(k)} i)^{1/v}])\binom{[(\log^{(k)} i)^{1/v}]}{u}(2^i + m + 2).$$

For large values of i (in dependence on the given values of u, v, k) the inequality

$$\tau_R(n) < (2^i + k + 2)(i+1) + 2(i - [(\log^{(k)} i)^{1/v}])\binom{[(\log^{(k)} i)^{1/v}]}{u}(2^i + m + 3)$$

holds, thus for a suitable constant K and for almost all values of i

$$\tau_R(n) < K(i - [(\log^{(k)} i)^{1/v}])(2^i + m)\binom{[(\log^{(k)} i)^{1/v}]}{u}.$$

Further

$$\tau_R(n) < K(i/(i+1))([(\log^{(k)} i)^{1/v}])^u (2^i + m)(i+1);$$

from that last inequality it follows that $\tau_R(n) < Kn(\log^{(k)} i)^{u/v}$. Because of the restrictive conditions on the value of n we have that $i < \log n$ holds, consequently $\tau_R(n) < Kn(\log^{(k+1)} n)^{u/v}$.

Q.E.D.

We shall describe still another class of subsets of the set A and determine the corresponding time functions. Let p, q, s, v_i (for $i = 1, \ldots, s$) be positive integers and u_i, k_i (for $i = 1, 2, \ldots, s$) nonnegative integers such that $v_i + k_i > 1$ ($i = 1, 2, \ldots$ $\ldots, s$), $0 < p/q < 1$. Let us denote $A^{(p,q,s,u_i,v_i,k_i)}$ such a subset of the set A that every i-block of a word from A fulfils the following conditions. Let us suppose the i-block divided into q parts of the same length (we shall omit details concerning the case that i is not divisible by q). Symbols in the last p parts of the block will be arbitrary, i.e. 0's or 1's. Let us divide the remainder of the block (i.e., its $q - p$ initial parts) into s portions of the same length. The first portion will contain just u_1 of 1's on the places (with respect to the first portion) of the type $\exp^{(k_1)} p^{v_1}$ for $p = 1, 2, \ldots$; the second portion will contain just u_2 of 1's on the places (with respect to the second portion) of the type $\exp^{(k_2)} p^{v_2}$ for $p = 1, 2, \ldots$; $\ldots$; the s-th portion will contain just u_s of 1's on the places (with respect to the s-th portion) of the type $\exp^{(k_s)} p^{v_s}$ for $p = 1, 2, \ldots$ For the just described set the following Theorem 3 holds; this theorem will not be proved here because its proof is similar (but technically more difficult) to the proofs of the both preceding theorems.

Theorem 3. *The set* $A^{(p,q,s,u_i,v_i,k_i)}$ *is* $T(n)$ *– recognizable, where*

$$T(n) = (n/\log n)^{1+p/q} \prod_{j=1}^{s} (\log^{(k_j+1)} n)^{u_j/v_j} .$$

Remark. By a similar procedure as in the case of the set $A^{(p,q,s,u_i,v_i,k_i)}$ it is possible to describe, for every function of the type

$$T(n) = n^{p/q} \prod_{k=1}^{t} (\log^{(k)} n)^{i_k/j_k} \quad (1 \leqq p/q < 2;\ 0 \leqq i_k/j_k ;$$

p, q positive integers; i_k, j_k nonnegative integers for $k = 1, 2, \ldots, t$), a set A_T (a subset of the set A) such that

$$A_T \quad \text{is} \quad T(n)\text{-recognizable} .$$

(Received October 22nd, 1968.)

REFERENCE

[1] F. C. Hennie: On-line Turing machine computations. IEEE Transactions on Electronic Computers *EC*-15 (February 1966), 1, 35—44.

O optimálních časových odhadech rozeznávání jistých množin slov Turingovými stroji typu on-line

PAVEL STRNAD

Budiž $T(n)$ funkce tvaru

$$n^{p/q} \prod_{k=1}^{t} (\log^{(k)} n)^{i_k/j_k}$$

($1 \leqq p/q < 2$; $0 \leqq i_k/j_k$; p, q přirozená čísla; i_k, j_k celá nezáporná čísla pro $k = 1, 2, \ldots, t$).

Pak existuje množina slov A_T taková, že

a) je-li M Turingův stroj typu on-line, který rozeznává A_T, potom platí $\tau_M(n) > C_M\, T(n)$ pro skoro všechna n; C_M je konstanta, která závisí na stroji M a $\tau_M(n)$ je časová funkce stroje M;

b) lze popsati Turingův stroj typu on-line M_T, který rozeznává množinu A_T a pro jehož časovou funkci platí pro všechny hodnoty n $\tau_{M_T}(n) < C\, T(n)$; C je vhodná konstanta.

Pavel Strnad, Vysoká škola strojní a textilní v Liberci, Studentská 5, Liberec.