COMPUTATIONAL COMPLEXITY OF ONE-TAPE TURING MACHINE COMPUTATIONS

J. Hartmanis

TECHNICAL REPORT
No. 68-3

January 1968

Department of Computer Science
Cornell University, Ithaca, N.Y. 14850

COMPUTATIONAL COMPLEXITY OF ONE-TAPE TURING MACHINE COMPUTATIONS

J. Hartmanis
Department of Computer Science
Cornell University
Ithaca, N.Y.

ABSTRACT. This paper is concerned with the quantitative aspects of one-tape Turing machine computations. It is shown, for instance, that there exists a sharp time bound which must be reached for the recognition of non-regular sets of sequences. It is shown that the computation time can be used to characterize the complexity of recursive sets of sequences and several results are obtained about this classification. These results are then applied to the recognition speed of context-free languages and it is shown, among other things, that it is recursively undecidable how much time is required to recognize a non-regular context-free language on a one-tape Turing machine. Several unsolved problems are discussed.

This research has been supported in part by National Science Foundation Grant GP-6426.

PRELIMINARIES

All through this paper we are concerned with the quantitative aspects of one-tape, off-line Turing machine computations. We assume that the Turing machine is used as a recognizer of sequences over some finite alphabet I. The set of all finite sequences over the alphabet I is denoted by I^* and the length of a sequence

$$w = x_1 x_2 \dots x_k \ , \quad x_i \text{ in } I \ , \ 1 \leq i \leq k \ ,$$

is denoted by $l(w)$; in this case

$$l(w) = k.$$

The null sequence is designated by Λ and thus

$$l(\Lambda) = 0.$$

The tape of a Turing machine M is unbounded on the right and the input string

$$w = x_1 x_2 \dots x_k$$

(in I^*) is written on the first k tape squares and the remaining tape is blank at the start of the computation.

A set of finite sequences A over I, $A \subseteq I^*$, is <u>accepted</u> or <u>recognized</u> by the Turing machine M if and only if M stops for all inputs w in I^* and accepts the input if it is in A and rejects it if it is not in A by entering the accepting or rejecting state, respectively.

Next we define three quantitative measures of the complexity of one-tape, off-line Turing machine computations.

1. The number of operations performed by a Turing machine in processing (rejecting or accepting) an input string is our measure of time. Let $T(n)$ be a computable function from non-negative integers into non-negative integers,

   $$T: \mathcal{J} \to \mathcal{J}.$$

   Then we say that a set A is $T(n)$-<u>recognizable</u> if and only if there exists a Turing machine M which accepts A and which processes every input of length n in $T(n)$ or fewer operations.

2. The amount of tape used by the Turing machine is our measure of memory. Let $L(n)$ be a computable function,

   $$L: \mathcal{J} \to \mathcal{J}.$$

Then the set A is said to be L(n)-*recognizable* or *recognized* with L(n)-tape if, and only if, there exists a Turing machine M which accepts A and which processes every input of length n using no more than L(n) tape squares.

3. The last complexity measure is based on the number of times M crosses boundaries between tape squares. Let R(n) be a computable function,

$$R: \mathcal{J} \to \mathcal{J} .$$

Then the set A is said to be R(n)-*recognizable* if, and only if, there exists a Turing machine M which accpts A and which processes every input of length n without crossing any boundary between adjacent tape squares more than R(n) times. (For related work see [1, 2, 3].)

We say that the Turing machine M *defines* T(n) if, and only if, for any input of length n, the machine M uses no more than T(n) operations and for some input of length n uses exactly T(n) operations.

Similarly, the machine M <u>defines</u> L(n) or R(n) if, and only if, M never uses more than L(n) tape squares or R(n) crossings between tape squares for inputs of length n, and for some input length n use exactly L(n) tape squares or makes exactly R(n) crossings of some boundary.

We now give a short description of crossing sequences which form the main analytic tool of this paper and which have been studied before [4] and recently in [1]. For a Turing machine M and tape t we associate with every boundary between adjacent tape squares of t an ordered sequence of states

$$s(1),\ s(2),\ \ldots,\ s(n);$$

in which the i-th state, $s(i)$, is the state the machine is in on the i-th crossing of this boundary during the computation performed by M when started on tape t ; we refer to this sequence as a crossing sequence. If

$$t = w_1 w_2$$

then the crossing sequence generated by M on the boundary between w_1 and w_2 is designated by

$$C(w_1;w_2).$$

It is seen that the crossing sequence $C(w_1; w_2)$ completely describes the information which is carried across the boundary between w_1 and w_2 by M in its computation. Thus we can easily show the next result which states that any tape segment between identical crossing sequences can be removed without affecting the computation on the remaining tape [1].

Lemma 1. If

$$t = w_1 w_2 w_3 w_4 w_5$$

and for M

$$C(w_1 w_2; w_3 w_4 w_5) = C(w_1 w_2 w_3; w_4 w_5),$$

then

$$C(w_1;\ w_2\ w_4\ w_5) = C(w_1;\ w_2\ w_3\ w_4\ w_5)$$

and
$$C(w_1\ w_2\ w_4;\ w_5) = C(w_1\ w_2\ w_3\ w_4;\ w_5).$$

Note that the machine M changes its state before it moves and therefore we see that if M accepts (rejects)

$$t = w_1\ w_2\ w_3\ w_4\ w_5$$

it will accept (reject)

$$t = w_1\ w_2\ w_4\ w_5.$$

RECOGNITION OF NON-REGULAR SETS

It is known [5] that if a set A is recognized by a one-tape, off-line Turing machine which does not write on its tape, then A is a regular set and can be recognized by a finite automaton or, equivalently, by a Turing machine which scans the input segment only once.

Recently Hennie [1] extended this result and showed that if M accepts A and for some constant k

$$R(n) \leq k \text{ or } T(n) \leq kn$$

then the set A is regular.

We will now show that there exist sharp bounds on how fast R(n) and T(n) have to grow for the recognition of non-regular sets on one-tape Turing machines. The first two theorems were obtained by Trachtenbrot [4] and independently by the author.

Theorem 1. If A is R(n)-recognizable and

$$\lim_{n\to\infty} \frac{R(n)}{\log n} = 0$$

then A is a regular set.

Proof. We will show that if

$$\lim_{n\to\infty} \frac{R(n)}{\log n} = 0$$

then

$$R(n) \leq k$$

for some constant k, and therefore by Hennie's result the set A is regular.

Let M recognize A and let M define R(n). (Thus R(n) is the longest crossing sequence generated by M on inputs of length n.) If R(n) is not bounded then there exists an infinite sequence of integers

$$0 < n_1 < n_2 < n_3 < \dots$$

such that

$$R(n_i) > R(n), \text{ for } n_i > n.$$

Thus M generates a crossing sequence of length $R(n_i)$, for the first time, on an input sequence of length n_i.

We now show that on the input segments t_i of length n_i, on which $R(n_i)$-long crossing sequences are generated, no crossing sequence can be generated more than twice. To see this, let

$$t_i = w_1 w_2 w_3 w_4, \; w_2 , w_3 \neq \Lambda$$

and assume that M on

$$t = w_1 w_2 w_3 w_4 w_b$$

generates

$$C(w_1 ; w_2 w_3 w_4 w_b) = C(w_1 w_2 ; w_3 w_4 w_b) =$$

$$C(w_1 w_2 w_3 ; w_4 w_b).$$

Then by the previous lemma we know that when M is started on

$$t' = w_1 w_3 w_4 w_b \quad \text{or} \quad t'' = w_1 w_2 w_4 w_b$$

it generates a crossing sequence of length $R(n_i)$ on the input segment

$$w_1\, w_3\, w_4 \quad \text{or} \quad w_1\, w_2\, w_4.$$

But since the segments are shorter than t_i we have a contradiction with the fact that a crossing sequence of length $R(n_i)$ is reached for the first time on an input segment of length n_i.

Thus on the input segments $t_1, t_2, \ldots,$ of length $n_1 < n_2 < \ldots,$ respectively, no crossing sequence is generated more than twice. We now use this fact to compute how fast $R(n_i)$ has to grow.

If M has Q states ($Q \geq 2$) then the number of different crossing sequence of length at most r is given by

$$\sum_{i=0}^{r} Q^i = \frac{Q^{r+1}-1}{Q-1} \leq Q^{r+1}.$$

Now if on a segment of length n_i no crossing sequence can be generated more than twice, then we must have

$$2Q^{R(n_i)+1} \geq n_i.$$

Taking logarithms to base Q on both sides of the inequality we get that

$$R(n_i) + 1 + \log 2 \geq \log n_i$$

whence

$$R(n_i) \geq \log n_i - 2.$$

But then

$$\lim_{n\to\infty} \frac{R(n)}{\log n} \neq 0.$$

Thus

$$\lim_{n\to\infty} \frac{R(n)}{\log n} = 0$$

implies that $R(n) \leq k$ and therefore A is regular.

<u>Corollary</u> 1. If a Turing machine M recognizes a non-regular set A and defines $R(n)$ then

$$\sup_{n\to\infty} \frac{R(n)}{\log n} > 0.$$

<u>Proof</u>. Follows directly from the theorem.

Next we show that there are non-regular sets which are recognizable with

$$R(n) = 2\ [\log_2 n].$$

(The symbol $[x]$ denotes the smallest integer k such that $k \geq x$.)

Lemma 2. There exists a one-tape Turing machine M which started on

$$t = 1^n w_b$$

computes with

$$R(n) = 2[\log_2 n]$$

the binary representation of n (i.e. M stops after writing the binary expansion of n on its tape).

Proof. The machine M sweeps from left to right over the segment of ones and marks off the first, third, fifth, etc. unmarked squares, then returns and repeats this process until all squares of this segment are marked off. It is seen that in the binary expansion

$$n = \sum_{i=0}^{k} a_i 2^i$$

[illegible]

the coefficient a_i is 1 if the rightmost unmarked square is marked off on the ith sweep; a_i is 0 otherwise. Thus to compute $a_k a_{k-1} \ldots a_1$ the machine just records the a_i according to the above procedure. Since this process is completed in $[\log_2 n]$ sweeps, we see that there exists an M which computes the desired binary expansion with

$$R(n) = 2[\log_2 n].$$

From the above proof it follows (as was already shown in [1]) that

$$n = 2^k$$

if and only if the process is finished in k sweeps and the rightmost one is marked off on the last sweep. Thus the non-regular set

$$\{1^{2^k} \mid k = 1, 2, \ldots\}$$

is

$$R(n) = 2 [\log n] \text{ - recognizable.}$$

Next we show that there also exists a sharp time bound which must be reached or exceeded for non-regular computations. This result and the following corollary were first obtained by Trachtenbrot [4] and independently by the author.

<u>Theorem</u> 2. If A is T(n)-recognizable and

$$\lim_{n\to\infty} \frac{T(n)}{n \log n} = 0$$

then A is regular.

<u>Proof</u>. Observe that the computation time is given by the sum over the length of all crossing sequences generated in the computation. Furthermore, the proof of Theorem 1 showed that

if A is not regular then there are infinitely many input segments

$$t_1, t_2, t_3, \ldots \text{ of length } n_1 < n_2 < n_3 < \ldots,$$

respectively, on which no crossing sequence is generated more than twice. It was furthermore shown in this proof that, even if M generates the shortest possible crossing sequences on the segments t_i, the longest crossing sequence $R(n_i)$ must be such that

$$R(n_i) \geq \log_Q n_i - 2.$$

Thus, for all i, the computation time $T(n_i)$ on t_i must be such that

$$T(n_i) \geq 2 \sum_{j=0}^{r} j\, Q^j \ , \quad r = [\log_Q n_i]-3,$$

since there are Q^j different crossing sequences of length j and no crossing sequence can be used more than twice. Note that M does not have to generate all crossing sequences of length less than $\log_Q n_i - 2$, but then short sequences have to be replaced by longer ones and the inequality is strengthened. Returning to the inequality we see that $T(n_i) \geq 2 \sum_{j=0}^{r} j\, Q^j \geq 2\, r\, Q^r \geq 2(\log_Q n_i - 3)\, n_i\, Q^{-3}$, and therefore

$$\lim_{n\to\infty} \frac{T(n)}{n \log n} \neq 0.$$

Thus

$$\lim_{n\to\infty} \frac{T(n)}{n \log n} = 0$$

implies that A is regular.

Corollary 2. If M recognizes a non-regular set and defines $T(n)$, then

$$\sup_{n\to\infty} \frac{T(n)}{n \log n} > 0.$$

Proof. Follows from the theorem.

Since the set

$$A = \{ 1^{2^k} \mid k = 1, 2, \ldots \}$$

can be recognized in $[\log n]$ sweeps each of length n, we see that A is

$$T(n) = 2n[\log n]$$

recognizable. Thus there exist non-regular sets which are $T(n) = 2n[\log n]$-recognizable.

The previous results can be extended to obtain a relation between the amount of time and memory used in computations with unbounded crossing sequence length.

Theorem 3. Let M stop for all inputs and define $L(n)$, $R(n)$ and $T(n)$. Then

$$\lim_{n\to\infty} \frac{n}{L(n)} = 0$$

implies that there exists $C \geq 0$ and N such that

$$R(n) \geq C \log L(n)$$

$$T(n) \geq C\, L(n) \log L(n)\ , \text{ for } n \geq N.$$

Proof. First we show that for an input tape

$$t = w\, w_b$$

M cannot generate two identical crossing sequences on the initially blank tape w_b. If

$$C(w\, b^r\ ;\ b^k\, w_b) = C(w\, b^r\, b^k\ ;\ w_b)\ ,$$

then by Lemma 1 we can remove the segment b^k and the computation will not be changed, except that it will have k fewer crossing sequences. On the other hand, the removal of b^k from w_b leaves w_b unchanged and therefore

$$t = w\, w_b$$

is unchanged. Thus M will generate exactly the same number of crossing sequences as before and (since M stops) we conclude that $k = 0$.

Since

$$\lim_{n\to\infty} \frac{n}{L(n)} = 0$$

there exists an integer N_1 such that for $n \geq N_1$

$$L(n) \geq 2n.$$

Thus for every n, $n \geq N_1$, there exists an input of length n for which

$$L(n) - n \geq 1/2L(n),$$

and we see that for this input M uses $1/2L(n)$ tape squares of w_b. This implies that M must generate at least $1/2L(n)$ different crossing sequences since no crossing sequence can be repeated on w_b. There are

$$\sum_{i=0}^{r} Q^i \leq Q^{r+1}$$

different crossing sequences of length r or less $(Q \geq 2)$. Thus to generate $1/2L(n)$ different crossing sequences we must have

$$Q^{R(n)+1} \geq 1/2L(n) ,$$

and therefore

$$R(n) \geq \log_Q L(n) - 1 - \log_Q 2.$$

This implies that there exists positive constants C_1 and N_2 such that

$$R(n) \geq C_1 \log_Q L(n) , \text{ for } n \geq N_2.$$

The computation time $T(n)$, $n \geq N_1$, is not less than the sum of the length of the $1/2L(n)$ different crossing sequences generated by M . Thus

$$T(n) \geq \sum_{i=0}^{r} i\, Q^i , \quad r = [\log_Q L(n)]-3 ,$$

and therefore

$$T(n) \geq (\log_Q L(n) - 3)\, L(n) Q^{-3} .$$

This implies that there exists positive constants C_2 and N_3 such that

$$T(n) \geq C_2 L(n) \log_Q L(n) , \text{ for } n \geq N_3 .$$

To obtain the desired inequalities we let

$$C = \min (C_1 , C_2) \text{ and } M = \max (M_2 , M_3) .$$

Theorem 4. Let M stop for all inputs, define $R(n)$, $L(n)$, $T(n)$, and let $R(n)$ be unbounded. Then there exist two positive constants C_1 and C_2 such that for infinitely many values n_i

$$R(n_i) \geq C_1 \log_Q L(n_i)$$

$$T(n_i) \geq C_2 L(n_i) \log_Q L(n_i) .$$

Proof. Since $R(n)$ is unbounded, there are infinitely many values

$$L(n_1) < L(n_2) < L(n_3) < \cdots$$

such that

$$R(n) < R(n_i) \text{ if } L(n) < L(n_i) .$$

By arguments similar to the ones used in the proofs of Theorems 1 and 2, we can show that no crossing sequence can be generated more than twice during the computation when $R(n_i)$ is reached for the first time (i.e. on the shortest $L(n)$). Again by counting the number of crossing sequences we conclude that for some $C_1 > 0$ and $N_1 > 0$ we have

$$R(n_i) \geq C_1 \log_Q L(n_i) \text{ , for } n_i \geq N_1 .$$

Similarly, we can now compute the number of operations performed in these computations and show that for some $C_2 > 0$ and $N_2 > 0$ we have

$$T(n_i) \geq C_2 L(n_i) \log_Q L(n_i) \text{ , for } n_i \geq N_2 .$$

By picking the

$$n_i \geq \max (N_1 , N_2)$$

we have the desired infinite set of integers for which the inequalities hold.

The previous results showed that there exists a sharp break in the computation time when we go from regular to non-regular computations. Next we turn to the classification of the complexity of non-regular sets by their computation time on one-tape Turing machines. For related results for multi-tape Turing machines see [2] and [6].

HIERARCHIES OF TIME-LIMITED COMPUTATIONS

In this section we investigate the classification of non-regular sets by the time required for their recognition.

For a computable function $T(n)$ $(R(n))$ we refer to the set of $T(n)$-recognizable ($R(n)$-recognizable) sets of sequences as a <u>complexity class</u> and designate it by C_T (C_R). The next result shows that there are infinitely many complexity classes.

<u>Lemma</u> 3. If $T(n)$ is computable, then there exists a recursive set of sequences A not in C_T.

<u>Proof</u>. By a simple diagonal process [2].

Next we show that every computation can be sped up by a linear factor, if we permit a trivial condensing of the input string. That is, we permit to write several input symbols per tape square.

<u>Theorem</u> 5. If A is $T(n)$-recognizable, then A is

$$[1/2\ T(n)]\text{-recognizable.}$$

<u>Proof</u>. Let the input string be condensed to two input symbols per tape square and let A be recognized by M in time $T(n)$. Then, by techniques similar to those used in the proof of Theorem 2 in [2], we can show that there exists a machine M' which recognizes A and performs one operation for every two operations of M. Thus M' recognizes A in time

$$T'(n) = [1/2\ T(n)].$$

The next result shows that for slowly growing time functions a slight (non-linear) increase in the computation time is sufficient to recognize more complicated sets. To prove this result we define sweep functions which are very easy to compute and are used to count the number of sweeps over the input segment performed by a machine M and to terminate this computation if the number of sweeps grows too large. The sweep functions play a role similar to the real-time functions used in the study of the computation time of multi-tape machines [2] and the realizable functions used in the study of memory limited computations [3].

Definition. Let F be a monotone, increasing function from integers into integers,

$$F: \mathcal{J} \to \mathcal{J} ,$$

such that for some $Q > 1$ and for large n

$$3n \leq F(n) \leq Q^n .$$

Then F is a sweep function if and only if there exists a computable, monotone, increasing function g,

$$g: \mathcal{J} \to \mathcal{J} ,$$

such that the set

$$A = \{1^{g(k)} \; 0^{F[g(k)]-g(k)} \mid k = 1,2, \ldots\}$$

can be recognized by a Turing machine M which makes no more than

$$F^{-1}(n)$$

sweeps over the input segment of length n.

(Note that $F(n)$ cannot grow more rapidly than the exponential function, since otherwise

$$\sup_{n\to\infty} \frac{F^{-1}(n)}{\log n} = 0 ,$$

and by Corollary 1 only regular sets can be accepted in $F^{-1}(n)$ sweeps. The lower limit for $F(n)$ is used explicitly in the proof of Theorem 6; see also the discussion after Corollary 4.)

The sweep functions form an interesting and rich class of functions (which should be investigated further). For the present application it is sufficient to note that this class contains very many of the commonly used functions. For example, the following are sweep functions:

$$F(k) = k^{p/q},\ p > q$$

$$F(k) = 2^k$$

$$F(k) = 2^{(p/q)k},\ p/q < 1$$

$$F(k) = k\ [\log_2 k]^p\ ,\ p = 1,2,\ \ldots$$

$$F(k) = k\ [\log\log k]$$

etc.

To gain a better understanding of sweep functions consider

$$F(k) = 2^k.$$

Choose $g(k) = k$. Then using Lemma 2 we see that the set

$$\{ 1^k 0^{2^k - k} \mid k = 1,2, \ldots \}$$

is recognizable in k sweeps with

$$k \leq \log_2 n = F^{-1}(n).$$

Thus

$$F(k) = 2^k$$

is a sweep function.

To see the use of the auxiliary function $g(k)$ consider

$$F(k) = k^{p/q} .$$

(See also [1] for the use of related techniques.) Let $g(k) = 2^{qk}$. Then to recognize the set

$$\{ 1^{2^{qk}} 0^{2^{pk} - 2^{qk}} \mid k = 1,2, \ldots \}$$

we construct a machine M which checks (by the process described in the proof of Lemma 2) whether the length of the input sequence and the length of the segment of ones are powers of two, and checks whether the umber of sweeps to verify this for the two segments is in the ratio $^p/_q$. This can be done in

$$kp\text{-sweeps}$$

over the input sequence. Since

$$n = 2^{pk}$$

we see that

$$F^{-1}(n) = 2^{kq} \geq kp \qquad \text{(for large } n\text{)}$$

and therefore

$$F(k) = k^{p/q}$$

is a sweep function.

By similar techniques we can show that many other functions are sweep functions. It is the author's conviction that sweep functions should be investigated in more detail and their properties compared to those of real-time functions and constructable functions. So far this has not been done systematically.

We now utilize sweep functions to generalize Hennie's results (Theorem 4 and Corollary 4 in [1]) and show that sweep functions can be used to define sets of sequences with sharp requirements for their recognition time.

Theorem 6. If F is a sweep function, then there exists a set of sequences which is

$$R(n) = F^{-1}(n) \quad \text{and} \quad T(n) = n\, F^{-1}(n)$$

recognizable and is not $R_1(n)$ or $T_1(n)$ recognizable if

$$\lim_{n\to\infty} \frac{R_1(n)}{F^{-1}(n)} = 0 \qquad \text{or} \qquad \lim_{n\to\infty} \frac{T_1(n)}{nF^{-1}(n)} = 0 .$$

Proof. We show that the set

$$A = \{\, w_i\ w\ w_i \mid w_i \in (0+1)^*,\ l(w_i) = g(k),\ w = {}_aF[g(k)] - 2g(k) \,\}$$

satisfies the theorem. For the sake of brevity let

$$g(k) = k.$$

Since $F(k)$ is a sweep function we can in $k = l(w_i)$ sweeps check whether the length conditions are satisfied for the three segments and at the same time check whether the first and third segments are identical. In this computation

$$n = F(k)$$

and therefore

$$F^{-1}(n) = k.$$

Thus A is

$$R(n) = F^{-1}(n)\text{-recognizable.}$$

Furthermore, since every sweep is no longer than n we see that A is recognizable with no more than

$$T(n) = 2\,n\,F^{-1}(n) \text{ operations.}$$

Thus (by Theorem 5) A is

$$T(n) = n\,F^{-1}(n)\text{-recognizable.}$$

Next we show that if

$$\lim_{n\to\infty} \frac{T_1(n)}{n\,F^{-1}(n)} = 0 ,$$

then A is not $T_1(n)$-recognizable. To see this, note that for a fixed k there are 2^k different w_i such that

$$w_i\, a^{F(k)-2k}\, w_i \text{ is in } A \text{ and } l(w_i) = k .$$

Let

$$A_k = \{\, w_i\, a^{F(k)-2k}\, w_i \mid l(w_i) = k \,\} \subseteq A .$$

Then all the crossing sequences generated by M in the middle segments,

$$w = a^{F(k)-2k} ,$$

of sequences in A_k must be different. Since, if

$$C(w_i\ a^p;\ a^q\ w_i) = C(w_j\ a^r;\ a^s\ w_j)\ ,\ p \neq r \quad \text{or} \quad w_i \neq w_j\ ,$$

then M will accept (by Theorem 1 of [1]) the sequence

$$w_i\ a^p\ a^s\ w_j \quad \text{not in} \quad A\ .$$

Thus in the recognition of the strings in A_k
M must use $2^k[\ F(k)-2k\ ]$ different crossing sequences on the middle segments w. Since $F(k) > 3k$ there is a $C > 0$ such that

$$2^k\ [\ F(k) - 2k\ ] \geq C\ 2^k\ F(k)\ .$$

There are

$$\sum_0^r Q^i \leq Q^{r+1}$$

different crossing sequences of length r or less for a machine with Q states, $Q \geq 2$. In order to have

$$C\ 2^k F(k)$$

different crossing sequences we must have

$$r + 1 \geq \log\ [\ C \cdot 2^k\ F(k)\ ]$$

and therefore for some $C_1 > 0$,

$$r + 1 \geq C_1 k + \log F(k).$$

Since

$$n = F(k) \quad \text{and} \quad k = F^{-1}(n)$$

we conclude that (for large n)

$$r \geq C_1 \cdot F^{-1}(n)$$

and therefore

$$\sup_{n \to \infty} \frac{R_1(n)}{F^{-1}(n)} \neq 0.$$

Thus A is not $R_1(n)$-recognizable if

$$\lim_{n \to \infty} \frac{R_1(n)}{F^{-1}(n)} = 0.$$

Next we compute a lower bound for the computation time. If all crossing sequences of length r or less are used in the computation then the average computation time for strings in A_k is given by

$$\sum_{0}^{r} i\, Q^i / 2^k \geq \log [C\, 2^k\, F(k)] \cdot C\, 2^k\, F(k) / 2^k$$

$$\geq C_3\, F(k) \cdot k$$

for $C_3 > 0$. Again, since $n = F(k)$ and $k = F^{-1}(n)$, we have that for some sequence in A_k the computation time must exceed

$$C_3 \, n \, F^{-1}(n).$$

Thus

$$\lim_{n\to\infty} \frac{T_1(n)}{n \, F^{-1}(n)} = 0$$

implies that A is not $T_1(n)$-recognizable.

<u>Corollary</u> 4. Let F be a sweep-function. Then there exists a set A such that

$$A \text{ in } C_T$$

implies that

$$\sup_{n\to\infty} \frac{R(n)}{F^{-1}(n)} > 0.$$

<u>Proof</u>. An immediate consequence of the theorem.

The above results establish for a wide class of functions (the set of sweep functions) sets of sequences with well-defined computation times. Unfortunately, the reasoning holds only for slowly growing time functions, namely, for

$$T(n) \leq n^2.$$

It is easily seen that this limitation, as in many other similar arguments, exists because we constructed sets in which two segments of length n or less had to be checked for identity. This can be done within n^2 operations and thus these results cannot be extended past n^2 by these techniques.

To obtain related results for more complex computations, that is for larger time functions, we are forced to use diagonal arguments. Unfortunately, the diagonal arguments for one-tape machines are quite cumbersome and the results obtained in this paper for large time functions are much weaker than the previous result. On the other hand, the author conjectures that for arbitrarily large time functions $T(n)$ the condition

$$\lim_{n\to\infty} \frac{T_1(n)}{T(n)} = 0$$

implies that

$$C_{T_1} \neq C_T .$$

The next result, obtained in collaboration with John E. Hopcroft, gives the best result obtained until now.

Theorem 7. Let $T(n)$, $T(n) \geq n^2$, be defined by a Turing machine and be computed on $L(n) = [\log T(n)]$ - tape. Then there exists a set which is $T(n)\ [\log T(n)]$ acceptable and not $T_1(n)$ acceptable for $T_1(n)$ such that

$$\lim_{n\to\infty} \frac{T_1(n)}{T(n)} = 0.$$

Proof. We give a short outline of the proof by constructing a Turing machine M which recognizes in time $T(n)$ [$\log T(n)$] a set A which is not $T_1(n)$ - recognizable for any $T_1(n)$ with

$$\lim_{n\to\infty} \frac{T_1(n)}{T(n)} = 0.$$

Let M be a Turing machine which carries out two different computational processes (on different tracks of the tape).

a) First computation: M attempts to interpret some initial part of the input tape $w\ w_b$ as a description of a Turing machine, M_i , and then proceeds to simulate what this machine M_i would have done when presented with the input tape $w\ w_b$. If M completes the simulation and M_i accepts w then M rejects it and vice versa. If w does not describe a Turing machine and the simulation cannot be carried out then the computation is stopped and w is rejected. It can be shown that for every w (whose prefix describes a machine M_i) there is a constant k_i , such that every operation of M_i can be simulated in k_i operations by M.

b) Second computation: In this computation M counts the number of operations which M has performed for the first computation and stops and rejects the input if the first computation exceeds $T(n)$ operations for an input w of length n. Since $T(n)$ is defined by a Turing machine, M can count up to $T(n)$ in $T(n)$ operations. (For the sake of simplicity we assume that $T(n)$ is defined by a machine which stops for every input of length n in $T(n)$ operations.)

The two computations are independent and to carry them out simultaneously M alternates the operations: after performing one operation in the first computation (simulation) M marks the tape square the head is on and "remembers" the state of this computation and then returns to perform one operation of the second computation (counting), again after performing this operation M marks the tape square the head is on and "remembers" the state of this computation; in order to keep the two "current head positions" lined up M now proceeds to move the whole lower tape pattern (counting) so that the two head positions line up. After this the cycle is repeated and the machine alternates between the computations. Since $T(n)$ is defined by a Turing machine and is computed on $L(n) = [\log T(n)]$ - tape, we see that one cycle in this computation can be performed in $3[\log T(n)]$ or fewer operations. Thus $T(n)$ cycles (or $T(n)$ operations) in the simulation of M_i can be performed by M using no more than

$$3T(n)[\log T(n)]$$

operations. This implies (Theorem 5) that the set A accepted by M is $T(n)$ [$\log T(n)$] - recognizable.

We now show that if a Turing machine M_j operates in time $T_1(n)$ and

$$\lim_{n\to\infty} \frac{T_1(n)}{T(n)} = 0 ,$$

then M_j cannot accept the set A accepted by the previously described machine M. To see this recall that, for some k_j, in k_j operations M can simulate an operation of M_j when the description of M_j is the prefix of an input w_j presented to M. Because of the limit condition there is an N such that for $n > N$

$$k_j\, T_1(n) < T(n)$$

and therefore for some sufficiently long input w (whose prefix describes M_j) the machine M has enough time to simulate what M_j would have done with the input w and do the opposite. Thus the set accepted by M_j differs from the set accepted by M. This completes the proof since we have shown that in time

$$T(n)\ [\log T(n)]$$

we can accept a set not acceptable in time $T_1(n)$.

From the previous proof it is seen that the factor $[\log T(n)]$ entered the result of the previous theorem because of the necessity of performing two independent computations: the simulation of M_i and the counting. The simulation was used to get a set which differs from all sets of sequences in C_{T_1} and the counting operation was used to terminate those simulations which required more than $T(n)[\log T(n)]$ operations. It seems very likely that with deeper insight into the nature of one-tape computations, we should be able to eliminate the big sweeps between the two independent computations and decrease the time lost in shuttling back and forth between the two processes.

In this connection it is interesting to recall that a corresponding result for multi-tape machines was first derived in [2] and that this result contained a "square." Only after the simulation of multi-tape machines on two-tape machines was understood better [6] was the result improved and now it also contains the factor $[\log T(n)]$. In both cases it does not seem that the best possible result has been obtained.

RECOGNITION OF CONTEXT-FREE LANGUAGES

In this section we study the recognition speed of context-free languages on one-tape Turing machines.

Lemma 5. There are non-regular context-free languages which are recognizable in time

$$T(n) = n[\log n]$$

and with

$$R(n) = [\log n].$$

Proof. The context-free language

$$A = \{ 1^k 0^k \mid k = 0, 1, 2, \dots \}$$

is recognizable with

$$R(n) = [\log n] \text{ and } T(n) = n [\log n].$$

To see this we just recall that with no more than [log n] sweeps a Turing machine can compute the binary expansion of the length of the segment of ones and the segment of zeroes and see if they are identical (using Lemma 2). Thus the non-regular set A is

$$R(n) = [\log n] \text{ and } T(n) = n[\log n]$$

recognizable.

For $W = x_1 x_2 \dots x_k$ let $W^T = x_k \dots x_2 x_1$.

Lemma 6. The context-free language

$$A = \{ W \# W^T \mid W \text{ in } (0+1)^* \}$$

is

$$T(n) = n^2 \text{ - recognizable}$$

and not $T_1(n)$ - recognizabe for any $T_1(n)$ such that

$$\inf_{n\to\infty} \frac{T_1(n)}{n^2} = 0.$$

Proof. By a simple counting argument on crossing sequences (see [1]).

By using sweep functions and constructions similar to those used in the proof of Theorem 3 in [7] we can show that there exist infinitely many different computational complexity classes of context-free languages between the time functions

$$T(n) = n[\log n] \text{ and } T(n) = n^2.$$

The most interesting problem which is still open is to determine a least upper time bound in which every context-free language can be recognized on a one-tape Turing machine. We know that this time bound has to be at least

$$T(n) = n^2.$$

It is the author's conjecture that there are context-free languages which cannot be recognized on a one-tape machine in time

$$T(n) = n^2.$$

The next result establishes an upper bound for the recognition of context-free languages. It is not known whether it is a good bound and it is the author's conjecture that it can be improved considerably.

Corollary 3 [Younger]. Every context-free language is

$$T(n) = n^5 \text{ - recognizable.}$$

Proof. In [8] it is shown that every context-free language is $T(n) = n^3$ - recognizable on a multi-tape Turing machine. A straight forward implementation of this algorithm on a one-tape Turing machine shows that every context-free language is

$$T(n) = n^5 \text{ - recognizable.}$$

Next we show that it is recursively undecidable how much time is required for the recognition of non-regular context-free languages.

Theorem 8. There is no algorithm to decide whether a non-regular context-free language generated by grammar G can be recognized in time

$$T(n) = n\,[\log n].$$

Proof. Let A and B be k-tuples of non-null binary strings,

$$A = (w_1, w_2, \ldots w_k)$$

$$B = (v_1, v_2, \ldots, v_k)\ ,\ v_i, w_i \in (0+1)^* - \Lambda.$$

Let A' and B' be the same k-tuples over a primed alphabet and indexed from k+1 to 2k;

$$A' = (w'_{k+1}\ ,\ w'_{k+2}\ ,\ \ldots\ ,\ w'_{2k})$$

$$B' = (v'_{k+1}\ ,\ v'_{k+2}\ ,\ \ldots\ ,\ v'_{2k})\ ,\ v'_i\ ,\ w'_i\ \varepsilon\ (0'+1')^* - \Lambda\ ,$$

and for all i, if

$$w_i = a_1\ a_2 \cdots a_r\ ,\ a_j\ \varepsilon\ \{0+1\}\ ,$$

then

$$w'_{k+1} = a'_1\ a'_2\ \ldots\ a'_r\ .$$

Let $\underline{i}$ be the binary representation of the integer i. Consider the deterministic, context-free languages [10]

$$L(A) = \{\underline{i_1}\ \#\ \underline{i_2}\ \#\ \ldots\ \#\ \underline{i_p}\ \#\ \#\ \overset{w}{a}_{i_p}\ \ldots\ a_{i_2}\ \overset{w}{a}_{i_1}\ |\ p = 1,2,\ldots\}$$

$$L(B) = \{\underline{i_1}\ \#\ \underline{i_2}\ \#\ \ldots\ \#\ \underline{i_p}\ \#\ \#\ \overset{v}{b}_{i_p}\ \ldots\ \overset{v}{b}_{i_2}\ \overset{v}{b}_{i_1}\ |\ p = 1,2,\ldots\}.$$

Then

$$L(A) \cap L(B) \neq \emptyset$$

if, and only if, there exists a sequence of indices

$$i_1 , i_2 , \ldots , i_k$$

such that

$$a^T_{i_1} a^T_{i_2} \ldots a^T_{i_k} = b^T_{i_1} b^T_{i_2} \ldots b^T_{i_k} .$$

But the problem whether such a sequence of indices exists for a pair of binary k-tuples

$$A, B$$

is an unsolvable problem [9,10]. Thus it is recursively undecidable whether

$$L(A) \cap L(B) = \emptyset.$$

Recall that L(A) and L(B) are deterministic context-free languages and therefore

$$\overline{L(A)} \text{ and } \overline{L(B)}$$

are context-free languages [10]. Thus

$$\overline{L(A)} \cup \overline{L(B)} = \overline{L(A) \cap L(B)}$$

is a context-free language. Consider now the context-free language

$$L = \{a^n\, b^n \mid n = 1,2,\ldots\} \left(\overline{L(A\ A')} \cup \overline{L(B\ B')}\right),$$

where A A' is the 2 k-tuple

$$(w_1, w_2 \ldots, w_k, w'_{k+1}, w'_{k+2}, \ldots, w'_{2k}).$$

Then

$$L(A\ A') \cap L(B\ B') = \emptyset$$

implies that

$$\overline{L(A\ A')} \cup \overline{L(B\ B')} = \Sigma^*$$

is the set of all sequences over 0, 1, 0', 1', # and L is recognizable in (Lemma 5)

$$T(n) = n[\log n].$$

If

$$L(A\ A') \cap L(B\ B') \neq \emptyset ,$$

then there is a sequence of indices

$$i_1 , i_2 , \ldots , i_r$$

such that

$$a^T_{i_1} a^T_{i_2} \ldots a^T_{i_r} = b^T_{i_1} b^T_{i_2} \ldots b^T_{i_r}$$

and

$$a'^T_{i_1+k} a'^T_{i_2+k} \ldots a'^T_{i_r+k} = b'^T_{i_1+k} b'^T_{i_2+k} \ldots b'^T_{i_r+k} .$$

If we designate

$$\# \underline{i_1} \# \underline{i_2} \# \ldots \# \underline{i_r} \text{ by } E_1 ,$$

$$\# \underline{i_1+k} \# \underline{i_2+k} \# \ldots \# \underline{i_r+k} \text{ by } E_2 ,$$

$$a_{i_r} \ldots a_{i_2} a_{i_1} \text{ by } A_1$$

and

$$a_{i_r+k} \cdots a_{i_2+k}\, a_{i_1+k} \text{ by } A_2$$

then the set L contains sequences of the form

$$E_{i_1} E_{i_2} \cdots E_{i_m} \# \# A_{i_m} \cdots A_{i_2} A_{i_1}$$

with i_j in {1,2} and no sequence in

$$(E_1+E_2)^* \# \# (A_1+A_2)^*$$

which is not of this form. But then using Lemma 6 we conclude that L requires

$$T(n) = n^2$$

for its recognition. Thus

$$L \text{ is } T(n) = n[\log n]\text{-recognizable}$$

if, and only if

$$L(AA') \cap L(BB') = \emptyset$$

and therefore we cannot decide whether L can be recognized in

$$T(n) = n\,[\log n].$$

REFERENCES

1. F. C. Hennie, "One-tape, off-line Turing machine computations", Information and Control 8 (1965) 553-578.

2. J. Hartmanis and R. E. Stearns, "On the computational complexity of algorithms", Trans. Amer. Math. Soc. 117 (May 1965) 285-306.

3. R. E. Stearns, J. Hartmanis and P. M. Lewis, "Hierarchies of memory limited computations", Proceedings of the Sixth Annual Symposium on Switching Circuit Theory and Logical Design, 179-190, IEEE, New York, 1965.

4. B. A. Trachtenbrot, "Turing computations with logarithmic delay", (in Russian) Algebra i Logica 3 (1964) 33-48. English translation in University of California Computing Center Technical Report No. 5, Berkeley, Calif., 1966.

5. M. O. Rabin and D. Scott, "Finite automata and their decision problems", Sequential Machines: Selected Papers, Editor E. F. Moore, 63-91, Addison-Wesley, Reading, Massachusetts, 1964.

6. F. C. Hennie and R. E. Stearns, "Two-tape simulation of multi-tape Turing machines", J. of the A.C.M. 13 (October, 1966) 533-546.

7. P. M. Lewis, R. E. Stearns and J. Hartmanis, "Memory bounds for recognition of context-free and context-sensitive languages", Proceedings of the Sixth Annual Symposium on Switching Circuit Theory and Logical Design, 191-202, IEEE, New York, 1965.

8. D. H. Younger, "Context-free language processing in time n^3", Proceedings of the 1966 Seventh Annual Symposium on Switching and Automata Theory, 7-20, IEEE, New York, 1966.

9. E. Post, "A variant of a recursively unsolvable problem", Bull. Amer. Math. Soc. 52 (1946), 262-268.

10. S. Ginsburg, "The mathematical theory of context-free languages", McGraw-Hill, New York, 1966.