KYBERNETIKA ČÍSLO 6, ROČNÍK 1/1965

Real-Time and Complexity Problems in Automata Theory

JIŘÍ BEČVÁŘ

This paper analyses the background from which problems of complexity in automata theory arise, and discusses a series of questions connected with action in real time.

1. INTRODUCTION

In recursive function theory, which provides an adequate basis for the description of constructive objects as well as of algorithmic processes occurring in mathematical logic and in the theory of deterministic digital computers, the attention was for a long time concentrated on solution of problems in a principal sense. A typical example is the question of decidability of some problem: if one succeeded in giving positive answer to this question, one was not, as a rule, much worried about the complexity of the decision procedure. Except for the important classes of elementary and primitive recursive functions, no detailed classification of recursive functions was exhibited. Grzegorczyk's classification of primitive recursive functions [1] apparently did not for many years attract due attention.

It was until in the 60's that the question of the complexity of constructive processes became really actual, in connection with an intensive study of special types of automata [2, 3] on the one side and of the generative power of various models of grammar [4] on the other. Among other results which exerted a stimulating influence on this development one can count Shannon's theory of measuring information, Post and Kleene's [5] classification of unsolvable problems, and perhaps also the continuing work concerning complexity of Boolean functions. It seems that the principal ideas of a new theory emerged independently and almost simultaneously in the heads of several persons. The first source to be found in the literature is Rabin's [6] summary of a lecture held at a meeting of the Israel Mathematical Union in 1959. But the first paper published in extenso is probably Yamada's [7a], where results of his two years older dissertation [7] are summarized. Approximately at the same time, Kolmo-

gorov [8] developed his conception of measuring the complexity of algorithms in a series of lectures at the Moscow State University. In the further development, an important role has been played by McNaughton's [9] survey, where especially questions concerning computations in real time are stated with perfect clearness and urgency. Among the most important papers hitherto published (or to be published) let us note [10, 11, 12, 13, 14]. A somewhat different direction is followed in the papers of Ritchie [15] and Cleave [16]. The author was not familiar with the doctoral dissertations of Blum [18] and Cole [18] when writing this report.

In this paper, which is intended partly as a survey, we shall pay special attention to the background from which problems of complexity arise, and discuss a series of questions connected with action in real time. Various types of automata will be compared, first as far as their construction is concerned, then according to their power when they are operating without or with certain restrictions on time, space etc. Several examples will illustrate problems of real time computations. We shall end with an account of some results of Hartmanis, Stearns, Trachtenbrot and others.

2. RESTRICTIONS

We shall exclude from our considerations two properties, which are actually present in almost all real devices, viz. a) probability, b) continuity; we shall thus restrict ourselves to a deterministic and discrete theory.

However attractive the use of the probabilistic concept of information in measuring complexity may appear, it seems that, on the one hand, it is still unclear in what direction the intervention of probability would be most appropriate. On the other hand, in the area of algorithms, we are primarily faced with deterministic (or indeterministic, but not tied with probability [2]) processes, exhibiting, as their most characteristic feature, purely combinatorial internal relations. Accordingly, a combinatorial variant of the concept of information would fit here, but a coherent theory meeting properly the needs of the theory of algorithms does scarcely exist nowadays.

Continuity is currently present e.g. in the field of automatic control. It seems that a really adequate theory of complexity of continuous processes may be somewhat hampered by the fact of our ignorance of the (existing!) frontier, beyond which an approximation of a continuous process by discrete, algorithmic ones is no more feasible. Now, no matter what the microstructure of the world may be, it seems that the macroview is for us of primary importance. This can be viewed as one of the justifications of the discreteness supposition.

Let us note that both entropy and continuity appear in the extremely interesting book [19] of Vituškin, the main result of which is that, in the domain of continuous functions, an appropriate and nontrivial measure of complexity is n/s (or n), for an s times differentiable (or analytic) function of n arguments.

There is some difference between how the concept of complexity is conceived in everyday life (including mathematical practice) and how it is treated in the present theory. Intuition and common sense tell us that the sequence 1011010001 is more complex than 0000000000, that from two patterns the one is more complex (more irregular) than the other, that a concrete proof is more complex than another, that the task of finding the way out from a labyrinth is harder than from another labyrinth, that the directions for use of one device are more complex than those concerning another device, that exercize no 37 from a book is harder to solve than exercize no 12, etc. These examples include a considerable variety of types of problems and it is questionable whether it would be possible to reduce them all to one universal model. A characteristic feature of a number of them is that they are (at least prima facie) concerned with the complexity of one sole finite object. Measuring the complexity of a single finite sequence of symbols is a typical example. (One possible way how to measure this complexity is by determining the number of internal states of a minimal finite autonomous automaton which, at its output, is able to produce this sequence.)

In contrast with this, the existing theory is concerned with measuring complexity of infinite problems. Their solution is conceived as a process, running in (real) time. Such a problem, consists as a rule of an infinity of partial problems (partial results), thus one is especially focused not on individual, but on asymptotic estimates of complexity.

It should be emphasized that the question of complexity of a problem includes as its integral part following questions:

1) In what form are the data of the problem given (e.g., encoding); in what form is the solution of the problem supplied.

2) Which means (e.g. which type of machine) are admissible for the solution.

3) Under which supplementary conditions (restrictions on time etc.) is the solution to be reached.

To set up a classification of problems then means first to determine which problems are solvable under which conditions, and then to compare them on the basis of some (partial) ordering of the conditions themselves, e.g. according to the mutual relations (magnitude) of time or space limits laid upon the solution, or according to the power of the type of automaton needed for the solution, etc. One has to expect, of course, that if the conditions, under which solutions to problems are to be supplied, are hardly comparable, then the resulting classification will be accordingly interwoven as well. Consequently, unless when led by special requirements of practice, we have to try to reach a classification which would be universal enough and relatively invariant under some intuitively unimportant changes of conditions 1)–3). Otherwise our results would have rather the character of industrial patents than of general mathematical theorems.

The problems occurring in the literature in connection with complexity questions are formulated rather nonuniformly, thus a normalization of them would be desirable. Let us try to describe the problems in a slightly more systematic way.

There are several reasons not to be mentioned here which condition the fact that, in machines, information is encoded mostly in the serial (sequential) manner. Accordingly, it is processed gradually, in correspondence with the flow of real time. Thus it is natural to formulate the basic problems in a way which is in harmony with the idea of a theoretical model of automaton, which is provided with a sequential input and output (in autonomous automata, the input may be absent, or it is irrelevant; more about various types of automata will be said in sect. 5).

I. Problems of the type *stimulus-response*. Usually solved by automata with input and output. There are four basic types according as the sequences of symbols representing the stimulus and the response are finite or infinite.

II. *Recognition* problems. Sequences are to be classified with respect to a given property. This problem can be viewed as a special case of the previous one; the machine accepts an input sequence (recognizes whether it has or has not the given property) on the basis of an associated property of the corresponding output sequence (in machines, the output sequence is often identical with the sequence of states of some central unit of the machine; among the states there may be some "final" states; their appearance in the sequence of states determines the above mentioned property of the output sequence).

III. *Generation* problems. The machine has to generate a sequence (usually infinite) of symbols. Solved mostly on autonomous automata.

To these three, schematically formulated basic problems, the majority of all known problems can be reduced. The interpretation used of course plays its role and has bearing on the evaluation of the complexity of the solution of a problem. Let us quote some of the possible interpretations.

I'. 1) Input and output sequence finite.

Computation of an arithmetical function of the type $N \to N$ or $N \times \ldots \times N \to N$ (N = the set of natural numbers).

2) Input and output sequences infinite.

A mapping of the type $\langle 0, 1\rangle \to \langle 0, 1\rangle$, i.e. from real numbers into real numbers. Alternatively: realization of a constructive operator (transformation).

3) Input sequence finite, output sequence infinite. Can be viewed as a (less traditional) variant of the generation problem, e.g. the input sequence represents the conjunction of axioms of a finitely axiomatizable theory, the output provides an enumeration of all theorems of this theory.

4) Input sequence infinite, output sequence finite. Perhaps never investigated.

According to II above, there is a possible interpretation as a recognition problem for real numbers from $\langle 0, 1\rangle$.

Mixed interpretations are also possible, e.g. in computations of partial recursive functions the output sequences will generally be both finite and infinite.

II′. 1) Recognition of sets (predicates) of natural numbers by computing their characteristic functions.
2) Recognition of properties of words in abstract languages.

III′. 1) Generation of a real number from $\langle 0, 1\rangle$, i.e. of its dyadic expansion.
2) Generation of a set of natural numbers (words). The codes of the numbers are in the output sequence separated by a special symbol.

Besides these interpretations of the basic problems, less traditional ones have been studied as well. E.g. many problems may be formulated both as stimulus-response problems and as generation problems. The problem III′ 1) may be viewed as the computation of a function f, where the value $f(n)$ is displayed by the n-th symbol of the output sequence. (Let us note that if as the generating device a Turing automaton with several working tapes and an ever-moving output tape is used, then the function f is primitive recursive [20]. An arbitrary general recursive function can be obtained in this way only under the supposition that the output is not all the time productive; this is the case of a nonuniform output, see sect. 5). Similar interpretation may be given to the problem III′ 2): here $f(n)$ is the number whose code lies between the n-th and $(n + 1)$-st occurence of a separating symbol in the output sequence. Another mode how to use the process of generation for the computation of a function is Yamada's [7a]: The output sequence contains only the symbols 0, 1. If we denote as $g(n)$ the n-th member of the output sequence, then Yamada's machine defines a function f such that $f(n)$ is the least m with the property $\sum_{i=1}^{m} g(i) = n$. On the other hand, the problem III′ 2) of generating a set of words may conversely be put as a stimulus-response problem. E.g. in Trachtenbrot [14], the Turing machine is interpreted alternatively as generating the set of those words which are images of all possible finite input words. Still another method how to evaluate constructively (but without respecting requirements of economy and of real time) a function f would be by generating the sequence of exactly the pairs of type $(n, f(n))$. To close this section, let us note that, curiously enough, in the theory of abstract languages, automata have been used almost exclusively for recognition purpose, though the most frequent approach to languages (i.e. via grammars) is the generative one.

5. TYPES OF AUTOMATA

Here we intend to survey some of the types of automata hitherto considered in the literature, among them those used in recent papers dealing with complexity questions. As real-time operations constitute one important part of the general complexity

problem, we shall discuss several questions concerning real time already in this and the next section, in close connection with some relevant features of the design and interpretations of various types of automata.

As it is known, in the abstract theory of discrete synchronous automata the length of tacts (units of operation time) is irrelevant, important being but their succession (order). Thus the concept of real time loses here in fact its primary importance: it remains here only as the reflection of an idea of how many operations can be realized in a time unit. Experience tells us that in real devices (at least in those which are akin to different variants of the Turing machine) the number of possible operations per unit of time is a priori limited. (This is true independently of the idea that the restriction upon velocity of operations implies a restriction upon the space – e.g. upon the number of squares on a tape – which can be processed in 1 sec. Even if we should imagine that the squares may indefinitely diminish with increasing index, so that the machine would not need to be of a potentially infinite size, the insurmountable bound on the velocity of operations still remains to exist, conditioned by the ever increasing difficulties as to the performance of such microoperations.)

Having thus substituted for real time the number of internal operations performable during a tact, we tacitly assumed that that part of the machine which becomes active in this time has an a priori limited complexity (which, according to what has already been said, may be reduced to the supposition of its limited size). In the iterative nets [21, 22] the situation is different and the active part may be steadily growing. The requirement of limited complexity of the active part is to be replaced in them by an obvious local variant. We shall however leave these devices aside, as well as the problems concerning the coordination (may be multi-level) of the activity of their parts which, as in the case of an ever augmenting population, may be combined with essential difficulties.

Let us now first consider three concepts, basic with automata: input, output, end of activity.

A) Input. Let us distinguish two cases:

a) *Free input.* At time t, immediately accessible to the automaton is only the t-th symbol of the input sequence. (If the automaton has to use, at t, the information concerning another input symbol – usually the t'-th where $t' < t$ –, then it must store it in the internal state, or on a special tape, or it must have an additional reading head.) Free input may be realized in an automaton either without tape of with the aid of an input tape, on which an input reading head is moving uniformly and without rewriting, one square per unit of time.

b) *Bound input.* We always suppose that the input sequence is given to the automaton independently so that it cannot choose it freely and when it exerts some changes upon it, this is a part of the internal action of the automaton. We speak of a bound input when the automaton is provided with an input tape (see a) above) on which, besides the input head, another, working head is moving and

processing the tape in an arbitrary manner. In usual formulations, the input head is passed over in silence and it is supposed that the input sequence has been written on the tape prior to the begin of the action of the machine. In cases where the input sequence is finite one can manage with one sole head (at the rate of the prolongation of action), modifying the head so as to serve both as input and working: first the head puts upon the tape symbols of the input sequence and then, as soon as the head begins to be fed by a special "neutral" input symbol signalizing that the input sequence is over, the head gradually returns to the first square of the tape and then starts operating as an actual working head.

B) Output

a) *Free output.* The symbols of the output sequence are given uniformly, one symbol per tact. This includes also the case of a *nonuniform* (free) output, when among the output symbols there may again appear a neutral one. The corresponding interpretation of the output sequence then consists in dropping occurrences of this symbol in the final result. Again the machine may or may not have an output tape. If it has, then the neutral symbols may be eliminated from the result by stipulating that their appearance means that the output (printing) head does neither print nor move.

b) *Bound output.* The output sequence is worked out on a tape which simultaneously serves as working tape. In the case of a finite output sequence this variant can again be reduced to the preceding one: During the actual operation of the machine a free output will produce neutral symbols. After the halt of the original machine, the new machine will return to the first square of the working tape and then, through its output, reproduce the result.

C) End

a) *Natural end.* According to what has already been said in II, sect. 4, the standard way of indicating the end is by means of final internal states of the (central unit of the) automaton. (To this case also other variants may be reduced, e.g. when the head goes off the working tape, etc.) Alternatively it is possible to signalize the end by the appearance of a special "final" symbol in the output sequence [23].

b) *Forced end.* We are faced with this kind of end when there are some external restrictions (of the kind of time or space limits, etc.) laid upon the operation of the machine. The most typical example is the classical finite sequential machine, where the machine is supposed to end its action at t, if the length of the input sequence is t. (Note that there the "final" internal states of the machine do not correspond to a real stop of the machine, since they may appear – when considered as output symbols – several times in an output sequence. Finite automata having a natural end instead of a forced one and consequently capable of producing output sequence whose length is different from the input sequence were studied e.g. in [23, 15].) In the case of general prescribed limits the forced end comes in when the machine exceeds these limits. If, in addition, the machine is provided with means for a natural end, then either this

end is reached within the limits or it is not. In the latter case we have to do with a forced end and the result (output sequence) is either viewed as identical with the partial result hitherto reached or alternatively this kind of end is interpreted as non-resultative. The interpretation used may be of importance especially in recognition problems, in connection with the difference between the concepts of strong and weak recognizability. (A set A of sequences is strongly recognizable by an automaton, when the automaton is capable of giving the answer "yes" to all sequences belonging to A and the answer "no" to sequences not belonging to A. Weak recognizability means that the automaton gives the answer "yes" just to sequences belonging to A, but it may be without result for the remaining sequences.) Note that, as a rule, the restrictions which are taken into consideration are external – they are not "built in" the automaton –, consequently it is not obvious that e.g. the complement of a weakly recognizable set should be recognizable under the same restrictions.

Now, let us schematically review some of the types of automata occurring in the literature, especially in connection with real-time and complexity questions, and let us specify them according to some of the characteristics just discussed.

(The variety of types of automata as well as of interpretations of the basic problems is responsible for the sometimes considerable difficulties concerning the possibility of comparision of the complexity of the (solutions of the) problems).

I. Automata without special working tapes

1) *Classical finite state sequential machines.*
Free input; free output.

2) *Two-way (one-tape) finite automata* [2].
Bound input, input tape with a two-way motion of the working, only writing head; free output.

3) *Linear-bounded automata* [3].
Bound input, input tape having the length of the input sequence and provided with a two-way moving and rewriting working head; free output.

4) *Combinatorial automata* [23].
Several variants, including also the preceding cases. Bound input, input tape generally with a two-way moving and rewriting head, subject to various types of restrictions; free nonuniform output.

5) *Classical Turing machines.*
Bound input; bound output. The input tape serves simultaneously as output tape and as working tape.

II. Automata with special working tapes

1) *Push-down automata* [24, 12].
Push-down working tape, free output. The input is either free or (in [12]) bound, with an input tape provided with two-way moving and only reading working head.

2) *Turing automata with special working tapes.*
They have all a free output, generally nonuniform. Working tapes are processed in an arbitrary manner.

a) [13, 12] Free input; free output; one or more working tapes.

b) [12] Bound input, the input tape is provided with a two-way reading only head; free output; one or more working tapes.

c) [11] As sub a), with the only difference that there may be several heads on a working tape.

d) [1] As sub a), but the working tape is only one and it is many-dimensional.

e) [7a] As sub a), but without input. The output is uniform.

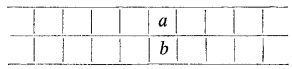
6. COMPARISON OF SEVERAL TYPES OF AUTOMATA

Let us now discuss the possibility of simulating the operation of one automaton by another in real time.

First, let us compare the classical Turing machine T_1 with the Turing machine T_2 having a free input and output and one working tape, since it seems that T_2 stands to T_1 in a very close relation. It turns out, however, that a faithful and simple mutual simulation of these two types of automata is hardly possible. To see it, consider first T_1. It has bound input (and consequently an imaginary, writing only input head), whereas the input of T_2 is free. Of course we can suppose that T_2 starts with an empty working tape, since all input information is fed into it through input. Despite of the further difference consisting in T_2 having free and T_1 bound output, it seems that to simulate at least the proper action of T_1 on T_2 in real time is possible only if we admit (what is very natural) that the working tape of T_2 will be provided with an *additional*, writing only head, which corresponds to the imaginary input head of T_1. This head brings directly from the input the initial tape information of T_1 and puts it on the working tape. Thus the new machine will not be a pure T_2 machine. Similar situation seems to arise whenever one wishes to simulate in real time a machine with bound input on a machine with free input.

Conversely, trying to simulate T_2 on T_1, we are meeting with the main difficulty that to T_2, the t-th member of the input sequence is accessible at t. T_1 has no possibility to mimic this fact, even when we decide to introduce in T_1 the mentioned imaginary input head, since this head has in T_1 no direct connection with the central unit.

Now, let us look at the relation between the types II 2a) and II 2c) (see sect. 5), i.e. between Turing machines with several working tapes and one single head on each tape, and Turing machines with several (finite number) heads on each working tape. First we find that *the operation of a machine T with n working tapes and one single head upon each can be simulated in real time by a machine T' with one working tape and n heads upon it.* It suffices to show this for $n = 2$. The working tape of T' is obtained by considering the tapes of T as one tape:



The working symbols in T' will be the pairs $\frac{a}{b}$. The two heads of T' will in fact be the heads of T, and each of them will be processing "its" half of the new tape. Note that obviously in T' the condition is satisfied that if the heads are both scanning the "same" square of the new tape, they both produce the same new symbol on it; on the other hand, the possibility is not excluded of both heads scanning on *different* squares the same symbol, but rewriting it differently.

An argument similar to the above shows that, more generally, *the operation of a machine with several working tapes and several heads upon each may be simulated in real time by a machine with one working tape and several heads upon it.* Thus the machine with one working tape and several heads is universal from the point of view of operation in real time (with respect to machines of the type II 2 c).

On the other hand, a similar reduction of a one-tape many-head automaton to the many-tape one-head type seems not to be known. For simplicity, let us consider a machine with one tape and two heads. Then it is possible to formulate two conditions concerning its action (compare above):

(C_1) *Strong compatibility*
If both heads are scanning the same symbol (may be on different squares), then both do change it into the same symbol.

(C_2) *Weak compatibility*
If both heads are on the same square, then the scanned symbol is changed by both heads into the same symbol.

A machine satisfying the condition (C_2) is only capable of recognizing the coincidence of the heads on a square. It would be interesting to know in what mutual relation stand machines satisfying these two conditions, and especially whether the formal difference of these conditions has any bearing on the possibility of the elimination of more heads on a tape at the rate of introducing more tapes. Similarly it would be of some interest to study many-tape one-head machines which are capable of recognizing whether or not two heads on different tapes are scanning squares with the same index. If we assume that the tapes do not move (moving are only the heads; for a motivation, see next sect.), and that they are parallel, then such a supposition as to the ability of a machine is quite natural and materially justified.

To conclude, let us note that, following [12], it is easy to show that if in the operation of a machine we are not interested in time, but in space used, i.e. in the number of squares successively scanned on all working tapes, then it is possible – at the rate of the prolongation of necessary time – to simulate the action of an arbitrary many-tape and many-head machine on a machine with one working tape and one head processing it.

7. INADEQUACY OF MANY-TAPE AND MANY-HEAD MACHINES FOR REAL-TIME COMPUTATIONS

The classical Turing machine (i.e. without the special input head which would begin to write the input sequence on the tape simultaneously as the working head starts to process the tape) may be viewed as a purely mathematical model, regardless of the possibility of its realization. On the other hand, this model was designed with the intention to represent an abstract version of the constructive treatment of information as is accomplished by men or real machines. Especially, the elementary acts are chosen so as to have a limited complexity and as to be realizable within the time reserved for a single tact (see sect. 5). This intention has some consequences as to the interpretation of the motion along the tape. Sometimes it is supposed that the central unit and the head (let us assume that they are fixedly attached to each other) remain without motion and that what is moving is the whole tape. Even if we suppose that the tape is growing, thus having always only finite length, nevertheless its length may exceed an arbitrary limit or, more precisely – e.g. when the tape is wound up – it will occupy an arbitrarily large space. (It has already been pointed out that to imagine the tape to be concentrated in a limited piece of space is combined with essential difficulties.) Then it seems not to be realistic to assume that to put such a tape with an astronomical size into motion is an operation of limited complexity (e.g. the energy needed for this is apparently arbitrarily large). Consequently, one is led to adopting the alternative interpretation, according to which the tape remains immobile and the pair head + central unit (which is a device of finite size) is moving along it.

Similar reasons lead us to adopt this point of view with a machine having working tapes, with one head upon each. If there are several tapes, then of course the central unit can no more be fixedly attached to all the working heads, since, in view of the immobility of the tapes, the heads may generally find themselves at an arbitrary distance. At the same time, whatever may be the position of the central unit, at least one head will have an arbitrary distance from it. But then it is no more possible to ignore the velocity of the propagation of signals between the heads and the central unit. Even if we assume that the operation of transfer of a signal is an operation of limited complexity (of course this is highly questionable) which lays claim only to time (the central unit waits until it receives the signal from the more distant head, in order to be able to process it in synchronization with the signal from the other head), it is however evident that we have no more justification to interpret this mode of action as an action in real time and to measure real time by the number of tacts.

A similar argument (which of course is not purely mathematical) may be set up against machines with several heads on the tapes, even if the working tape is only one. And, finally, difficulties of an analoguous kind arise even with the classical Turing machine when we assume it to be supplied with a special input head (see above). If we suppose (which seems reasonable) the central unit attached to the

ordinary, working head, then the input head will generally, after a sufficiently long time, find itself at an arbitrary distance from the central unit. Though there is no active communication necessary between the input head and the central unit, it is somewhat inconvenient to find such an important and surely active part of the machine as the input certainly is to be operating somewhere at an uncontrollable distance from the main body of the automaton. It would perhaps bring some profit and it would reflect more realistically the state of affairs, if we introduce into such models of automata the concept of a special channel (denoted e.g. as transporting tape), which would serve only the purpose of the transport (may be at a sufficiently high speed) of signals. Note that in the field of digital computers the comparability of the speed of operation with the velocity of signals also leads to new problems.

Let us finally put the following question: What can be said about the kind of operation of automata with many tapes and many heads upon each, if we know that the distances of all the heads (on the same as well as on different tapes) remain during the action always limited by a fixed number? From the point of view of simulation in real time, we may, according to the result from sect. 6, restrict ourselves to the sole case of an automaton with one tape and several heads. It is then possible to show that, *under the assumption that the distance of the heads is always limited, the kind of operation of this (and consequently of others as well) type of automaton may be simulated in real time by an automaton with one working tape and one head only*. Thus under the mentioned conditions, this most simple type of automaton is already the most universal. It is convenient to divide the proof into two steps. First by encoding the information on the tape by sufficiently large blocks we are able to construct a simulating machine in which the heads will always scan either the same or two neighbouring squares. Second, by an additional modification of the new machine, we can achieve that the information as to the action of the heads on neighbouring squares can always be stored in the internal state of the central unit of the machine. Consequently, one head will be enough.

8. EXAMPLES OF REAL-TIME COMPUTATIONS

We now give several examples of how many-tape and many-head machines can be used for solving problems in (strict) real time. All the problems are recognition problems, having connection with various kinds of grammars (in the sense of Chomsky [4]). The symbol w will always mean a nonempty word consisting of symbols from some finite set. (In the examples I–IV, this set will be $\{a_1, \ldots, a_n\}$.) The symbol w^{-1} means the converse word. All the machines which will be considered are Turing machines with free input and output, generally with several tapes and several heads upon each tape.

I. *Words of the type* $w\$w^{-1}$. They form a linear context-free language (linear in the sense that on the right side of each substitution rule, there appears at most one nonterminal symbol). Grammar:

Terminal symbols: $a_1, \ldots, a_n, \S$
Nonterminal symbols: S
Rules:

$$S \to a_i \S\, a_i \quad (1 \leqq i \leqq n)$$

$$\S \to a_i \S\, a_i$$

This set can trivially be recognized in real time by a one-tape one-head machine.

II. *Words of the type* ww^{-1}. They again form a linear context-free language. Grammar:

Terminal symbols: $a_1, \ldots, a_n$
Nonterminal symbols: $S, A_1, \ldots, A_n$
Rules:

$$S \to A_i a_i$$

$$A_i \to a_i A_j a_j \quad (1 \leqq i, j \leqq n)$$

$$A_i \to a_i$$

It seems not to be known whether this set is recognizable in real time by a many-tape one-head machine (cf. [9]). We do not know whether this is possible even by a many-head machine. It would be interesting to know whether a recognition in real time is at all possible (by a suitable machine).

III. *Words of the type* $w\S w$. They form a context-sensitive language. Grammar:

Terminal symbols: $a_1, \ldots, a_n, \S$
Nonterminal symbols: $S, A_1, \ldots, A_n, B_1, \ldots, B_n$
Rules:

$$S \to a_i \S\, B_i \qquad A_i a_j \to a_j A_i \quad (1 \leqq i, j \leqq n)$$

$$B_i \to a_i \qquad A_i B_j \to a_j B_i$$

$$\S \to a_i \S\, A_i$$

This set can trivially be recognized in real time by a one-tape two-head machine. On the other hand, P. Strnad [25] has recently shown that this can also be achieved by a three-tape one-head machine. Let us outline how this machine operates. Let $w = b_1 \ldots b_k$ and, for simplicity, let us suppose that k is even, $k = 2m$. The action of the machine, when its input is fed by a word of type $b_1 \ldots b_k \S\, c_1 \ldots c_k$, is divided into three time intervals A, B, C, corresponding to the following tacts:

$$A : 1, 2, \ldots, k, k+1 \ (= \text{appearance of } \S)\,,$$

$$B : (k+1)+1, (k+1)+2, \ldots, (k+1)+m\,,$$

$$C : ((k+1)+m)+1, \ldots, ((k+1)+m)+m = 2k+1\,.$$

The machine starts with empty tapes and with each head scanning the first square

of its tape. The action on the tapes T_1, T_2, T_3 during these intervals is as follows:

A) T_1: The head moves uniformly to the right and writes $b_1, \ldots, b_k$.
T_2: The head marks the first square and then, in each even tact, goes one square to the right, writing nothing.
T_3: The head is standing on the first square and does not write anything.

B) T_1: Uniform motion to the left. The head reads $b_k, \ldots, b_{m+1}$ and, through central unit, sends this information to T_3.
T_2: The head moves to the left and writes $c_1, \ldots, c_m$.
T_3: Uniform motion to the right. The head writes the symbols $b_k, \ldots, b_{m+1}$, supplied by T_1.

C) T_1: The head continues to move to the left and reads $b_m, \ldots, b_1$.
T_2: The head moves to the right and reads $c_m, \ldots, c_1$.
T_3: The head goes to the left and reads $b_{m+1}, \ldots, b_k$.

During the third time interval *C*, the central unit, in addition, accomplishes two kinds of action which make the final recognition possible: a) Comparison of the input symbols $c_{m+1}, \ldots, c_k$ with the symbols $b_{m+1}, \ldots, b_k$ read on T_3. b) Comparison of the symbols $b_m, \ldots, b_1$ read on T_1 with the symbols $c_m, \ldots, c_1$ read on T_2.

It would be interesting to know whether in this problem three tapes constitute the minimal number of necessary tapes. In the case of an affirmative answer, this would mean that it is generally not possible to simulate in real time a one-tape n-head automaton by an n-tape one-head machine.*

IV. *Words of the type ww.* It can be shown that this set is a context-sensitive language. On the other hand, like in the example II, we do not know whether it is at all recognizable in real time.

V. Let, for a symbol c, be $c^k = cc \ldots c$ (k times).
Let

$$R_1 = \{w : w = 0^m 1^k a 0^m \,;\ m, n = 1, 2, \ldots\}\,,$$
$$R_2 = \{w : w = 0^m 1^k b 1^n \,;\ m, n = 1, 2, \ldots\}\,,$$
$$R = R_1 \cup R_2\,.$$

Then R_1, R_2, R are context-free linear languages.
The grammar for R (including simultaneously grammars for R_1, R_2):
Terminal symbols: 0, 1, *a*, *b*
Nonterminal symbols: *S*, *A*, *M*, *B*, *N*
Rules:

$$S \to 0A0 \qquad S \to N1B1$$
$$A \to 0A0 \qquad B \to 1B1$$

* Added in proof: P. Strnad has recently succeeded in showing that this language can be recognized by a two-tape one-head machine.

$$\begin{array}{ll} A \to 1M & B \to b \\ M \to 1M & N \to N0 \\ M \to a & N \to 0 \end{array}$$

Rabin [13] has proved that R is recognizable in real time by a one-tape one-head machine.

VI. For a set D, let D^∞ be the set of all nonempty words consisting of symbols from D. Let $U = \{0, 1\}$, $V = \{2, 3\}$ and let

$$R_1 = \{w : w = uvau^{-1};\ u \in U^\infty,\ v \in V^\infty\},$$

$$R_2 = \{w : w = uvbv^{-1};\ u \in U^\infty,\ v \in V^\infty\},$$

$$R = R_1 \cup R_2 .$$

Then R is again a linear context-free language. Grammar:

Terminal symbols: 0, 1, 2, 3, a, b

Nonterminal symbols: S, A, B

Rules:

$$\left.\begin{array}{ll} S \to xAx & S \to NyBy \\ A \to xAx & B \to yBy \\ A \to yM & B \to b \\ M \to yM & N \to Nx \\ M \to a & N \to x \end{array}\right\} \quad x \in U,\ y \in V$$

Again Rabin [13] shows that R is recognizable in real time by a two-tape one-head automaton. (Consequently, as we know from sect. 6, there is a recognition by a one-tape two-head machine.) On the other hand, Rabin proves that R is not recognizable by a one-tape one-head machine.

VII. Let $U = \{0, 1\}$ and let

$$R = \{w : (w = u_1 \S u_2 \S \ldots u_n \S v) \,\&\, (n \geqq 1) \,\&\, (u_1, \ldots, u_n, v \in U^\infty) \,\&$$
$$\&\ (\text{there exists such an } i,\ 1 \leqq i \leqq n, \text{ that } u_i = v^{-1})\} .$$

Then R is a context-free language. Grammar:

Terminal symbols: 0, 1, §

Nonterminal symbols: S, A, B, C, $\bar{0}$, $\bar{1}$

Rules:

$$\left.\begin{array}{ll} S \to \bar{x}Ax & \bar{x} \to B\S x \\ A \to xAx & B \to x \\ A \to \S & B \to xB \\ A \to \S B\S & B \to xC \\ \bar{x} \to x & C \to \S B \end{array}\right\} \quad x \in U$$

Hartmanis and Stearns [11] prove that R is not recognizable in real time by any many-tape one-hand machine. It is an open question whether R is recognizable in real time by a one-tape many-head machine. It would be interesting to know whether the set of the converse words

$$R^{-1} = \{w : w^{-1} \in R\}$$

which, at first glance, seems to be recognizable more simply than R, is in fact recognizable in real time.

9. PROBLEMS OF REAL TIME. METHODS OF MEASURING COMPLEXITY

Though various requirements on operation in real time provide but one of the possible tools for measuring complexity of processes, there is an intuitive feeling, supported by everyday experience, that problems of real time are, when considered from the practical point of view, of a special urgency. Thus it is worth beginning with several heuristic considerations concerning the mutual relation between some practical aspects of real time and the position which real time takes on in the abstract, mathematical theory of automata and consequently in the theory of complexity as well.

One encounters in everyday life a series of situations where some task must be achieved within certain time limits: production of newspapers, forecast of weather, TV transmission etc. In these examples the "input" is relatively independent of the "output". This corresponds, in the language of automata, to devices with an unregulated input. All machines considered in sect. 5 were intended to solve problems with an unregulated independent input information. Now, properly speaking, this kind of practical activity in fact represents a rather rare exception, though time limits mostly remain to be present: automatic control, chess playing, driving of a car, tennis etc. Here there is always a feedback between input and output. This is especially true for the most common activity in real time, namely for the whole life of an animal or man: We are steadily influencing our input by motion, speech, work, etc. Thus the models of machines from sect. 5 as well as the problems that we have stated as basic for them do represent but a limited portion of actually arising situations. Second, and this seems to be more important, we realize that no machine or animal is exposed to severe requirements of action in real time (especially with unregulated input) for an unlimited period of time. There are long periods of time where the input is so to speak "in rest" or highly indifferent (sleep, current conversation). Moreover, for the solution of important problems in real time, machines and animals are usually being prepared in advance by training, gathering of information, learning, preparing new materials etc.; the action in real time then begins only when the preparations are found sufficient for a successful solution, meeting the needs of real time. Further, let us emphasize that while the action in real time as a rule consists of solving, in real

time and iteratively, a may be unlimited succession of single problems of limited complexity, we do not know of any single practical problem to be solved in real time which would consist of partial problems involving arbitrarily long input sequences. On the other hand, in basic problems described in sect. 4 arbitrarily long input sequences are current. Thus this feature is in contrast with actual real-time processes: they do not provide for a sufficient time for encoding quantities discretely with an arbitrary precision which, when serial coding is used, requires long sequences and unlimited delay. E.g. in TV transmission, one has to restrict the precision of encoding and consequently the length of input sequences. To conclude, we see that the practice of serial discrete encoding of information in automata has as its consequence that, for long sequence, the concept of real time is a mathematical extrapolation which does no more reflect the urgency which real time has in practical processes or in the case of treating "short" information.

Now, let us return to the mathematical side of the question and let us try to set up some general principles serving as a basis for measuring complexity of the basic problems. (In this connection, let us recall the importance of the conditions 1)–3) from sect. 3.) These problems are mostly capable of being divided into an infinity of finite partial problems. E.g. the stimulus-response problem naturally consists in finding concrete response to a concrete stimulus. If the stimulus and the response in a problem are always finite, then if, on the basis of some characteristics of the solution (concerning space, time etc.), we are able to associate with every stimulus (input sequence) P a certain number $s(P)$, then s is a *signalizing* function (term due to Trachtenbrot) which yields a certain characteristic of the solution of the whole problem. Sometimes one takes for the basic parameter not the sequence P but its length n, and accordingly one considers the function $s'(n) = \max_P s(P)$, where P runs over all sequences of length n. Further, in questions of asymptotic behaviour, it is convenient to consider the "smoothened" function $s''(n) = \max_{i \leqq n} s'(i)$. Let us note that the use of signalizing functions was anticipated by Trachtenbrot already in [26], where they were introduced in connection with the study of constructive operators, appearing in Post's problem of recursive reduction. (Following Trachtenbrot, a function s is a signalizing function of a function f defined by a system E of equations, if, for every n, there is a computation of the value $f(n)$ from E in which the maximal involved number does not exceed $s(n)$.)

In the case when the solution of a problem represents one or more infinite processes, then one has to find another way how to divide it into partial problems. Thus e.g. in the problem of generating an infinite sequence on the output of an autonomous automaton, the above parameter n may be associated with the n-th partial output result, be it the n-th output symbol (when generating an expansion of a real number), or, more generally, the n-th output word (when generating a set of words or natural numbers), or in this latter case n may be associated with the lengths of the output words. Again, we thus receive some signalizing function which characterizes the process.

Note that a signalizing function does not characterize a problem but only its concrete solution under given conditions. The most important and most frequently used signalizing functions are (for simplicity, we give only the unaccented notations and use capital letters; according to the type of problem, n has one of the meanings mentioned above):

a) *Time* function: $T(n)$ = the number of tacts (steps) necessary for reaching the result.

b) *Space* function: $L(n)$ = the number of squares used on various tapes for reaching the result.

Trachtenbrot [14] takes into consideration also the following signalizing functions:

c) *Regime*: $R(n)$ = the maximum of transgressions of the head of the common boundary of two neighbouring squares of the tape during the process of reaching the result, the maximum being taken over all squares.

d) *Oscillation*: $O(n)$ = the number of changes of the direction of the motion of the head in the process of reaching the result.

If especially $T(n) \leqq n$, then we say that the solution is reached in (strict) real time. With automata having input and output, this corresponds, in stimulus-response problems, to the idea of a free input, or of a bound input with a special input head. In connection with the problem of generation on an autonomous automaton, this terminology is adequate only if the n-th partial result is the n-th output symbol and the output is uniform. Thus e.g. Yamada's functions computable in real time actually do not fit into this pattern. The output of his machines is nonuniform (instead of Yamada's output symbol 0 we may imagine a neutral symbol; cf. sect. 5) and if we conceive, following a standard formulation, the computation of a function as a stimulus-response problem, then the value $f(n)$ is generally not provided in strict real time, determined by the length of the stimulus, i.e. of the code of n. Since Yamada uses unary encoding of the numbers, we may imagine an alternative arrangement of such computations, where the argument n is given to the input of some machine and the value $f(n)$ is simply the number of tacts needed by the machine to reach the final state. Then the time function $T(n)$ of the computation is directly equal to the computed function. This is, properly speaking, the true sense of Yamada's term "real time". Thus Yamada in fact does not classify functions according to $T(n)$, but he exhibits a certain class of functions, computable in certain manner. It would be interesting to know the mutual relation of his kind of computing his functions with the more standard one just mentioned.

Instead of measuring, with the aid of signalizing functions, the complexity of some concrete solution of a problem, we should perhaps like to achieve more than this, namely determining complexity of the problem itself. The problems would then be classified into classes of equally complex problems. Such a classification apparently has not yet been given. Instead, one is trying to define when one problem is decisively more complex than another. Such an ordering was given by Rabin in [6] for the

class of all recursive functions, under the condition that the functions are computed by means of Post's canonical systems. A criterion used is the number of necessary steps (applications of productions). Since Post's systems are not monogenic, whereas it is convenient to work with algorithmic (deterministic) devices, let us try to restate the main idea using instead the concept of the classical Turing machine and the time function $T(n)$. If M is a machine computing a function f, let $T_M(n)$ be the number of tacts needed by M to compute $f(n)$. Then we say that a function g is more complex than a function f if there exists a machine M computing f such that, for every machine L computing g, there exists n_0 such that $T_M(n) < T_L(n)$ for all $n > n_0$. Rabin states a result to the effect that the ordering of recursive functions which corresponds to this relation is nontrivial. (Unfortunately, there is no indication as to the place of most common functions in this classification and as to its structure as well.)

It seems, however, that a series of reasons provides motives for a somewhat different treatment of the whole problem than is a machine-independent, theoretically exhaustive and fine classification. Let us introduce the most important of them.

a) When trying to determine directly the complexity of a singular problem by means of signalizing functions we may find that it is not at all obvious that, among the various signalizing functions corresponding to various concrete solutions of the problem, there can be found some which, according to a certain ordering of the signalizing functions, would be minimal. Then all we can do is to try to find lower and upper bounds on the complexity of solution of the problem.

b) Signalizing functions of (concrete solutions of) concrete problems may be very complex and they are subject themselves to a classification (may be the same). In order to obtain something less complex than which we started with, it is advisable to exhibit a priori an auxiliary (may be incomplete) classification or ordering of some sufficiently reasonable signalizing functions. Simultaneously, it is profitable to consider these functions not as immediate signalizing functions, but only as their estimates. This leads to the constitution of large classes of problems each of which has a solution with a signalizing function satisfying the given estimate. (Another approach is Ritchie's [15], where functions and the signalizing functions of their computations are subject to the same inclusive classification: the $(i + 1)$-st class contains all functions having a computation with the signalizing function belonging to the i-th class.)

c) We are not equally interested in all problems or functions, but mostly in those occuring rather frequently or deriving their importance from other branches of research (e.g. various kinds of languages).

d) We have decisive reasons to be interested in the (possibility of) solution of problems with the use of some special kind of machines. Especially, we are interested in the converse problems of determining, whit the use of estimates of signalizing functions, the power of various types of machines themselves.

These motives are partly responsible for the emphasis laid upon the mutual relation, sometimes considerably complex, between the following things: a) type of pro-

blem, b) type of machine, c) type of grammar, d) estimate of signalizing function, e) type of signalizing function.

10. SOME RECENT RESULTS

In this last section, let us survey some of the interesting results obtained recently by Hartmanis, Stearns, Lewis, and Trachtenbrot. With the exception of the work of Trachtenbrot, let us restrict, for simplicity, to recognition problems, though the main body of results in the basic papers [10, 11] is concerned with problems of generating expansions of real numbers.

In the two mentioned papers which are concerned with a theory of complexity using time estimates of problems, these estimates ("time functions") are mostly chosen to be Yamada's real-time computable functions. The machines considered are Turing machines with free input and free nonuniform output, having several working tapes. There is a somewhat unorthodox definition of recognizability by such a machine M: M (strongly) recognizes a set R of words if, for any infinite input sequence and for any n, the n-th output symbol is 1 iff the initial segment of length n of the input sequence is a word belonging to R. Moreover, given a time function T, the set R is said to be T-recognized by M, if it is recognized by M in such a manner that the n-th output symbol is given within $T(n)$ or fewer tacts. Then there is first a variant of the "speed-up theorem": *If R is T-recognizable and $T(n) = n + E(n)$, $E(n) \geqq 0$, then, for every $\varepsilon > 0$, R is T_ε-recognizable, where $T_\varepsilon(n) = n + \varepsilon E(n)$.* Thus the rate of time $E(n)$ exceeding the necessary real-time minimum n may be reduced in an arbitrary linear manner. Second, a hierarchy of recognition problems results on the basis of the following result: *If, for two time functions T_1, T_2 real-time computable and monotone increasing, we have* $\sup T_1(n)/[T_2(n) \cdot \lg T_2(n)] = \infty$, *then there is a T_1-recognizable set which is not T_2-recognizable.* Finally, as far as the relation between one-head and many-head machines is concerned, there is the following result: *If a set R is T-recognizable by a many-tape, many-head machine, then it is T^2-recognizable by a one-tape one-head machine.*

Space estimates of complexity of recognition problems are the main object of the paper [12]. The machines used are again Turing machines with free nonuniform output, but the input may be free or bound (then two-way, without rewriting) and, in addition, the working tape may be of a push-down type. There is a whole series of results, from which let us mention only the following ones. (The space function L provides an estimate of the number of squares of the working tape which are used in order to obtain the result of recognition. Let us recall (cf. sect. 7) that in the space theory, one working tape is enough.) First there is an analogy of the speed-up theorem, now taking on the form of a linear compression statement: *If* $\sup L_1(n) : L_2(n) < \infty$, *then every L_1-recognizable set is L_2-recognizable.* Again there is a nontrivial hierarchy of recognition problems, resulting from the following result: *If — under minor restrictions as to L_1 — we have* $\sup L_1(n)/L_2(n) = \infty$, *then there*

exists a L_1-recognizable set which is not L_2-recognizable. There are further many problems and partial results concerning the mutual relation between languages, types of automata and types of space estimates. Let us mention only the results which are concerned with the lowest portion of the hierarchy, occupied by finite state languages (i.e. sets recognizable by finite automata). The authors state the following theorem: *For each machine model indicated below, there is a space function L_0 such that, for every space function L satisfying the equation* $\lim_{n\to\infty} L_0(n)/L(n) = \infty$, *every set which is L-recognizable (by the respective model) is already a finite state language.* The functions L_0 are as follows:

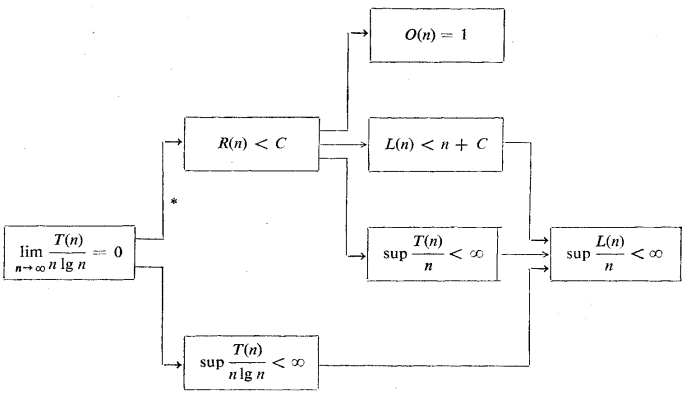
Model:	$L_0(n)$:
one-way Turing machine	$\lg n$
two-way Turing machine	$\lg \lg n$
one-way push down machine	n
two-way push down machine	$\lg \lg n$

(The specification "one-way" means free input, "two-way" means a bound input without rewriting.)

Now let us turn to Trachtenbrot's paper [14]. His model is essentially the classical Turing machine. He does not restrict himself only to recognition problems, but he considers chiefly general computational processes of the type stimulus – response. Recognition of a set is meant classically as the computation of its characteristic function. The main interest of Trachtenbrot is in the lowest portions of hierarchy and in mutual relations between restrictions involving the four kinds of signalizing functions (see sect. 8): a) time, b) space, c) regime, d) oscillation. His results are perhaps best displayed in a table where an arrow means that if there is a solution (computation) of a stimulus-response problem which satisfies the condition on the left, then there is a solution of the same problem (may be on another machine) satisfying the condition on the right. (See Fig. 1.)

As to the recognition problems, Trachtenbrot proves that if R is a set recognizable by a machine so that the corresponding time signalizing function satisfies the condition $\lim_{n\to\infty} (n \lg n)/T(n) = \infty$, then R is a finite state language. This includes as a special case Hennie's result [27], following which R is a finite state language whenever it is T-recognizable with $\inf n/T(n) > 0$.

To conclude, let us note that in a recent paper [28] Gladkij investigates a class of grammars, having a "restricted delay", i.e. grammars with the property that there exists a constant c such that, for every word w of length n, there is a derivation (in the grammar) of w using at most cn steps (substitutions). The class of corresponding languages then lies strictly between context-free and context sensitive languages. Gladkij's approach, following which the complexity of a problem (here: generation



* When the input words are running over all finite sequences

T = time function
L = space function
R = regime
O = oscillation

Fig. 1.

problem) is measured by the minimal number of steps which are necessary for the solution of it within some nonmonogenic combinatorial system is akin to the original conception of Rabin [6].

(Received May 11th, 1965.)

REFERENCES

[1] A. Grzegorczyk: "Some classes of recursive functions". Rozprawy matematyczne, IV (1953), Warsaw.
[2] M. O. Rabin, D. Scott: "Finite automata and their decision problems". IBM J. Research and Development, 3, (1959) 114–125.
[3] J. Myhill: "Linear bounded automata". WADD Tech. Note No. 60–165, Univ. of Pennsylvania Rept. No. 60–62 (1960).
[4] N. Chomsky: "Three models for the description of language". IRE Trans. on Information Theory, IT-2, (1956) 113–124.
[5] S. C. Kleene, E. L. Post: "The upper semi-lattice of degrees of recursive unsolvability". Ann. Math., 59, (1954) 379–407.
[6] M. O. Rabin: "Speed of computation of functions and classification of recursive sets". Bull. Res. Counc. of Israel, vol. 8F, (1959) 69–70.
[7] H. Yamada: "Counting by a class of growing automata". PhD Thesis, Moore School of Elect. Eng., University of Pennsylvania (1960).
[7a] H. Yamada: "Real time computation and recursive functions not real-time computable". IRE Trans. on Electronic Computers, EC-11 (1962) 753–760.
[8] Oral communication by S. V. Jablonskij (1962).

[9] R. McNaughton: "The theory of automata, a survey". Advances in Computers, vol. 2, 379–421, ed. F.L. Alt, Academic Press, New York 1961.
[10] J. Hartmanis, R. E. Stearns: "On the computational complexity of algorithms". Trans. Amer. Math. Soc. (in press).
[11] J. Hartmanis, R. E. Stearns: "Computational complexity of recursive sequences". Proc. of the Fifth Annual Symposium on Switching Circuit Theory and Logical Design, held at Princeton University, (1964) 82–90.
[12] J. Hartmanis, P. M. Lewis II, R. E. Stearns: "Classification of computations by time and memory requirements". Text of an invited paper read at the IFIP 65 Congress in New York, May 1965.
[13] M. O. Rabin: "Real time computation". Israel J. of Math., 1 (1963), 203–211.
[14] B. A. Trachtenbrot: "Turing computations with logarithmic delay" (in Russian). Algebra i logika 3 (1964), no 4, 33–48.
[15] R. W. Ritchie: "Classes of predictably computable functions". Trans. Am. Math. Soc., 106 (1963) 139–173.
[16] J. P. Cleave: "A hierarchy of primitive recursive functions". Zeitschrift f. math. Logik und Grundlagen d. Math. 9 (1963) 331–345.
[17] M. Blum: "A machine-independent theory of complexity of recursive functions". PhD Thesis, M.I.T., Departm. of Math. (1964).
[18] S. N. Cole: "Real time computation by iterative arrays of finite state machines". PhD Thesis, Computation Laboratory, Harvard University.
[19] A. G. Vituškin: "Ocenka složnosti zadači tabulirovanija". Gosudarstvennoe izdatel'stvo fiziko-matematičeskoj literatury, Moscow 1959.
[20] V. A. Uspenskij: „Lekcii o vyčislimych funkcijach", p. 470. Gosudarstvennoe izdatel'stvo fiziko-matematičeskoj literatury, Moscow 1960.
[21] F. C. Hennie: „Iterative arrays of logical circuits". M. I. T. Press 1961.
[22] J. Holland: "Iterative circuit computers". Proc. Western Joint Computer Conf., pp. 259–265, San Francisco 1960.
[23] J. Bečvář: "Finite and combinatorial automata. Turing automata with a programming tape". Proc. of the IFIP congress 62, 391–394, North-Holland Publ. Co., Amsterdam 1963.
[24] N. Chomsky: "Context-free grammars and push-down storage". Quarterly Progress Report No. 65, M.I.T., 187–194 (1962).
[25] Oral communication.
[26] B. A. Trachtenbrot: "Tabular representation of recursive operators" (in Russian). Doklady Akad. nauk SSSR, 101 (1955), 417–420.
[27] F. C. Hennie: "One-tape, off-line Turing machine computations". Information and Control (to be published). Reference from [12].
[28] A. V. Gladkij: "On the complexity of derivations in immediate constituents grammars". (In Russian). Algebra i logika 3 (1964), no 5–6, 29–44.

Problémy reálného času a složitosti v teorii automatů

JIŘÍ BEČVÁŘ

Autor se zabývá možnostmi precizace intuitivního pojmu složitosti problémů; jedná se o klasifikaci třídy řešitelných úloh, přičemž za základ klasifikace jsou brány časové a prostorové nároky, které klade řešení vyšetřovaného problému na automatu. Automat přitom není uvažován prostě jako matematický předmět, na který aplikujeme abstrakci potenciální uskutečnitelnosti, nýbrž jsou uplatňovány argumenty vycházející z představy automatu jakožto fyzikálního zařízení operujícího v reálném čase a prostoru.

Doc. Jiří Bečvář, Vysoká škola strojní a textilní, Hálkova 6, Liberec.