OTS: 60-11,428

JPRS: 2424

5 April 1960

ALGORITHMIZATION OF MATHEMATICAL PROBLEMS

- USSR -

L. A. Kaluzhnin

RETURN TO MAIN FILE

19990507 050

Distributed by:

OFFICE OF TECHNICAL SERVICES
U. S. DEPARTMENT OF COMMERCE
WASHINGTON 25, D. C.
Price: $0.75

U. S. JOINT PUBLICATIONS RESEARCH SERVICE
205 EAST 42nd STREET, SUITE 300
NEW YORK 17, N. Y.

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

JPRS: 2424

CSO: 3292-N

ALGORITHMIZATION OF MATHEMATICAL PROBLEMS

[This is a translation of an article written by L. A. Kaluzhnin in Problemy Kibernetiki (Problems of Cybernetic), No 2, Moscow, 1959, pages 51-68.]

Rapid-acting computer machines (RCM's) appeared comparatively recently: the first universal machines were built only 10 years ago while their wide adaptation at the present time is taking place literally in front of our eyes. Despite this, the problems which occur in connection with the operation of the already existing RCM's, with the searches for new fields of their application, and with the design of new machines have already succeeded in exerting a noticeable influence on the development of mathematical thought. Investigations of the problematics which are projected here are now being conducted in many scientific centers in the USSR as well as in foreign countries. In Kiev during the last year, two parallel seminars took place; in these, fundamental problems connected with the work of RCM's were discussed: in the Institute of Mathematics of the Acad Sci USSR, in the field of programming, under the leadership of V. S. Korolyuk and Ye. L. Rvacheva-Yushchenko in the Kiev State University -- a seminar on the theory of algorithms and mathematical logic under the leadership of the author. Both seminars supplemented each other fruitfully. Participating in the work of the seminars were the instructors of Kiev State University V. S. Mikhalevich and A. V. Skorokhod, the coworkers of the Institute of Mathematics, the graduate students G. G. Lyubchenko and A. A. Stogniy, and also various undergraduates, of which one should in particular mention I. N. Kovalenko, L. P. Nizhnik, N. M. Martinenko, V. Yu. Metus, and E. I. Chernobel'skaya. In this paper we wish to communicate about certain arrangements of problems which arose during our joint work and also about the first results which we obtained. It stands to reason that during the short period of our work we were able essentially to map those initial positions on the basis of which we intend to continue our investigations. But the individual, most simple problems were nevertheless advanced so far that they can even now be checked on the existing RCM's. It is interesting to note that the work of the seminars again confirmed the great role which the most

abstract fields of mathematics--mathematical logic, theory of machine mathematics.

Our aim is to find general methods of preparing mathematical and logical problems for their solution on RCM'S. Moreover, we are interested somehow whether a certain problem can "fundamentally" be solved by a machine and also whether it is possible to "actually" accomplish this solution at the existing level of development of the RCM'S or a fundamentally possible solution is at the present time practically not possible or economically not advantageous. This latter will take place, for example, when the number of elementary operations necessary for the solution of the problem is excessively large ($> 10^{12}$), when in the process of operation, the machine should retain exceptionally large initial or intermediate information, or, finally, when the preparation of the problem for its solution by machines is exceptionally cumbersome. Nevertheless, according to our deep conviction, the investigations of the fundamental possibility of solving mathematical problems on RCM's can be of great significance: on one side we are convinced that the RCM's will rapidly develop and that which is practically unattainable (or not advantageous) today becomes possible tomorrow; on the other side, such investigations are capable of leading to ideas of new machines and, in any case, to facilitate the development of RCM's.

It is necessary to state that up to now RCM's have essentially been used for the solution of calculating (arithmetic) machines, but recently the machines have successfully come into use also for the automation of other fields of intellectual labor: it is known that RCM's are used for the translation from one language into another, for accounting, and others. A large number of important mathematical and logical problems which occur in connection with the proactice of automatic control and, in particular, in connection with the operation of the RCM's themselves (programming and others) are being solved to a greater degree on RCM's. Naturally, more extensive problems arise. A large number of mathematical problems is known, not necessarity arithmetic, the solution of which requires only the mechanical application of rules once and for all established and capable of being excecuted by man, without knowing or investigating the sapid sense of the problem. We shall name several of these problems: (1) identical rearrangements of letter algebraic expressions (opening of the brackets, reduction of homogeneous members, etc); (2) differentiation of complex expressions of elementary functions (or of any other class of functions which satisfy some or other given functional equations); (3) calculation of indeterminate integrals from

the expressions of a certain class; (4) the solution of certain classes of differential equations containing some or other letter parameters; (5) finally, complete mathematical theories (comparatively simple, it is true) are known in which for each sufficiently formulated proposition with respect to the rules through a finite number of steps it is possible to verify whether it is the theorem of the given theory or not and to find automatically its proof.

As regards these and similar problems -- and the number of examples can be increased as much as is desired -- it is possible to make the same remark as with respect to the calculations with the use of four-action arithmetic: it is known what elementary operations and in what sequence it is necessary to employ, but if the original expressions are sufficiently complex, then the number of necessary operations can turn out to be so great that the practical solution of the problem becomes impracticable for man. (This pertains, for example, even for such a simple school problem as identical rearrangements of rational letter expressions, if the number of "letters" and "brackets" which make up the given expression is sufficiently great; the same can be said also with respect the other examples cited by us.) In all such cases the solution can fundamentally be accomplished by a machine and the origin and development of RCM's makes it possible partly even now and to a still greater extent in the near future to solve such problems in practice.

But one should not think that the accomplishment of the solution of such problems on RCM's is possible directly; on the contrary, new original problems arise therein which pertain to mathematics as well as to engineering. While putting all engineering problems aside, we shall take account of the mathematical problematics.

Those rules which are given to man for the solution of a problem of any class turn out to be insufficient for the solution of this problem on a machine: the formulation of the rules given usually verbally in common language is almost always not exhaustive -- it frequently appeals to the experience of the operator, which he acquired beforehand -- and is not well-defined -- at different stages of the solution process it gives the operator freedom of selection of one or another path (moreover, sometimes the selection should be made as a function of the signs of the original or intermediate results clearly not formulated in the rules). Besides, the formulation of the problem itself is not exhaustive and well-defined in mathematical practice. All that was said pertains in particular to the problem of proving the theorems of any solvable theory: the very formulation of the theorem

is usually verbal (inaccurate and not well-defined while in the course of the proof itself, the majority of intermediate stages -- in particular, the majority of logical conclusions -- are only implied. Briefly, it can be said that in the attempt to accomplish machine automation of a portion of mathematical work, we encounter the fact that in established mathematical practice problems, calculations, predictions, and conclusions are given in an insufficiently formalized form. By the problem of algorithmization we understand (as a first approximation) the further formalization of the mathematical apparatus, sufficient for its realization on RCM's. This problem turns out to be non-trivial. On the other hand, for problems of an entirely different type, an apparatus has been developed during the last decades which makes it possible to approach the investigation of our problem from a blank. Precisely, the so-called formalization of mathematical and logical conclusions was undertaken in connection with the central problems of the bases of mathematics for the investigation of the structure of axiomatic mathematical theories -- in the first place, arithmetic, theory of groups, and mathematical logic. Without going into the details of these complex problematics -- this would take us too far away -- we will state only several words how such a formalization will appear in principle. In order to describe any field of mathematics, a peculiar "language" is introduced: a certain number of elementary symbols called words; mechanical rules are given which make it possible to discern "intelligent words" of the given field, i. e., those which correspond to objects of the given theory and its predictions; finally, formulation is made of mechanical rules of actions on the words (for example, to delete a certain letter, to write in a certain letter, to reposition certain letters, to replace a certain letter with another, etc), the use of which in a definite order makes it possible to change some words into others. Tha language and rules are built, finally, in a manner for the mechanical actions to correspond and reflect a certain sapid sense which we have in mind in the given case; but all the rules should be formulated so that they could be applied to symbols without consideration of any sapid sense (such rules are called syntactic rules). It has been known for a long time that presentation is made in precisely such a form of arithmetic actions on numbers recorded in any system of numeration or identical rearrangements of rational letter expressions; but the same can be said of other formalized mathematical theories. Those mathematical problems which originate in the examination of the actions (according to given rules) on words are combinatorial

problems with respect to finite systems of the given symbols. The mathematical discipline called metamathematics deals with the study of the problems pertaining thereto.

Just as in metamathematics, so in the practice of automatic calculating machines -- which is of interest to us in the first place -- we have in general outlines the following situation in the solution of the problems: all the data which pertain to the problem under examination -- the entire information -- should be recorded in accordance with clearly formulated rules as the word in a certain alphabet; each problem is included in a certain class of problems (sometimes we call the class of problems as common problem) and the syntactic characteristics should be known (that is, characteristics pertaining to the recording and not to the sapid sense) of words which make it possible to characterize all those and only those words which pertain to the given class of problems; such a syntactic characterization should also be known for the class of words which correspond to the solutions of problems of the given class; further, it is necessary to know the syntactic rules of the action on words, which make it possible to process any word representing a specific problem of the given class into a word corresponding to its solution. The selection of such syntactic rules will be called the algorithm for the solution of problems of the class under examination, Since in the practice of RCM's the use of these rules takes place automatically without the interference of man, then the algorithm itself should be formulated in a well-defined manner in a certain standard form which corresponds to the engineering data of the machine (briefly -- "in the language of the machine"), as the word in a certain alphabet. The information of the rules of the solution of the common problem, expressed in the language of the machine, is called a program.

We wish to note that the projected system includes not only the automatic solution of mathematical problems; it includes also, for example, the problem of translation from one language into another. The translation from the French into Russian will appear thus: (1) Class of original words -- class of phrases of the French, recorded in the Latin alphabet (including punctuation marks and intervals between the words of a phrase). The class is characterized by syntactic rules which point what combinations of "letters" are grammatically correct French phrases. (2) Final class of words -- class of Russian phrases recorded in the Russian alphabet and syntactically characterized by grammatical rules. (3) Algorithm-- clear assignment of all the rules

(lexicographic and grammatical) of conformance between the Russian language and the French.

In a similar manner it is possible to formulate the problem of the programming and others.

And so, schematically we have the following situation:

(1) The common problem is formulated. Each specific problem is characterized by a definite word, the corresponding common problem -- by a class of words which have a certain syntactic characteristic. A group of words having this characteristic will be designated as the original information of the given common problem.

(2) It is necessary to find the solution of the common problem (its result). It is also characterized by a class of words having a certain syntactic characteristic; moreover, for each specific problem there corresponds one and only one word of this class -- the solution of this specific problem. The class of these words will be called the final (resultative) information of the common problem.

(3) (a) Formulation is made of the mechanical rules for the processing of the words, applied to the original words and which translate each of these into the final word -- a recording of the corresponding solution. The rules are given in a verbal formulation. The combination of these rules is called the algorithm of the solution of the common problem.

(b) The algorithm is formalized; the rules (a) are formulated as words in a certain alphabet; the algorithms for the solution of all possible common problems are given as a class of syntactically definite words. The latter are called systems of algorithms (or programs if we are dealing with RCM's).

The concrete apparatus of the formalization of the common problems (alphabet, words, syntactic characteristics, syntactic rules of action their "sapid decoding" etc) will in accordance with (1), (2), and (3) be called the language.

If the language is given, then the problem of algorithmization consists of the formulation of the sapid mathematical problems according to (1) - (3). It is easy to explain to oneself that each type of a universal RCM has its own language which depends on the technical data of the machine; ordinary programming consists of the algorithmization of the problems in the language of the definite machine. It is to note that the types of universal RCM's are rapidly being improved and modified: along with this, their language will also continue to change. For this reason, it already does not seem expedient to make investigations of the problem of

algorithmization as applicable to the language of any existing machine. To this it is necessary to add various other considerations. The language of any concrete, existing machine is stronglycomplicated by its technical limitation. This limitation can be transient in the sense that this or another technical difficulty is obviated, whereas it is precisely this that complicates the algorithmization. In the second place, we are convinced that the investigations in the field of algorithmization, in their turn, are capable of revealing new requirements for future RCM's and prompt new ideas for their realization. All these considerations lead to a situation in which in the course of our work on the algorithmization of problems, three basic, closely related directions appear gradually.

(1(The problem of the development of a standard language. This standard language should, on one side, reflect the inherent, non-transient characteristics of the existing RCM's and, on the other side, it should be adapted to the specificity of the algorithmization, i. e., to correspond as closely as possible established practice of mathematical calculations and conclusions.

(2) For an already selected language, to algoritmize with it different mathematical (particularly non-arithmetic) and logical problems.

(3) To develop an algorithm of the automatic translation from the standard language to the specific language of the concretely given universal RCM.

All three problems should be solved parallel because each new success or, conversely, the origin of fundamental or practical difficulties in the solution of one of the problems affects substantially the solution of two others. As a standard language, we shall for a time stop on the language of normal algorithms by A. A. Markov. As a common problem for algorithmization with this language, a problem was arranged for the recognition of logical identities in the theory of predictions. The solution of this problem in principle was attained (G. G. Lyubchenko, Yu. V. Meytus). At the same time, I. N. Kovalenko explained the principles of translation from the language of normal algorithms into the language of existing command-address RCM's. These first results indicated in which direction it is necessary to modify the language of normal algorithms for the gradual development of the standard language, necessary in the algorithmization of mathematical and logical problems. Moreover, substantial aid comes from an examination of other formalized languages known from mathematical logic -- recursive functions,

Turing Machine, etc. and formalisms which originated from programming practice, in the first place, from the system proposed by A. A. Lyapunov [3] and further developed by Yu. I. Yanov [5].

2. Normal Algorithms by A. A. Markov

We shall dwell briefly on the language of the Markov algorithms and examine it from the point of view of its suitability as a standard language for the algorithmization of problems. It turns out that from this point of view it has numerous shortcomings. (We would like to note that the criticism pertains to the Markov apparatus of algorithms as a possible language for the actual algorithmization of mathematical and logical problems and in this it is necessary to take into account that it was not developed for this purpose. As proof of the algorithmic insolvability of mass problems, the apparatus of the normal algorithms has numerous merits; for fundamental problems of such a type, it is, apparently, the most convenient of the apparatus (recoursive functions; Turing machines, etc. known up to now.)). We do not intend to present in detail the entire theory by A. A. Markov, but shall limit ourselves only to its elementary portion; moreover, equally to the extent in which it is, in the given case, necessary to us; those who are interested in more detailed information are referred to the monograph by A. A. Markov [4].

The language in which the algorithms are constructed are determined by the following original concepts. A certain selection of symbols is given -- "letters" called an alphabet. Any finite, linearly situated sequence of letters is called a word; in particular, it is permitted to examine blank sequence of letters -- a blank word (we will designate it by Λ). It is agreed in what manner in the given alphabet a recording (or coding) is made of those maghematical objects to which it is intended to apply the algorithms (moreover, it is not mandatory for each word to be a recording of a certain object).

We shall give several examples of how it is possible to code elementary mathematical objects.

It is customary to record natural numbers in the decimal system. This means that it is possible to select numbers as letters, i. e., an alphabet consisting of the symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. The recording of any natural number will then be any word the first letter of which differs from 0.

If it is agreed instead of this to use for the recording a binary system, then the two-letter alphabet 0, 1 is selected as the alphabet. For theoretical considerations it is simpler to record the natural numbers (including 0) in a single-letter alphabet consisting of an indifferent sign -- for example |,— to record the natural number n by a word consisting of n letters equal to |, i. e., 1—|, 2—||, 3—|||, etc. We shall select precisely the latter recording for further use.

For coding whole numbers, A. A. Markov proposes the use of an alphabet of the letters {—, | }, for recording rational numbers -- the alphabet {—, /, |}. Then, for example the number -- 2/3 will be coded easily by the understandable recording —||/|||.

For coding numerical vectors (of rational numbers) i. e., pairs of numbers, triplets of numbers, etc, it is necessary also to add to the alphabet for the recording of the numbers a certain new sign -- "separatory sign" -- for example the sign*. Then the pair (correspondingly a triplet) of rational numbers can be recorded in the form A*B (correspondingly A*B*C), where A, B, C are the recordings of the corresponding numbers. By means of an alphabet which includes one more separatory sign ⊛, it is possible to record the matrices. And precisely, the matrix can be regarded as a sequence of lines and each line as a numerical vector; in order to separate the lines, it is possible to use ⊛ as a separatory sign. Then, for example, the matrix

$$\begin{pmatrix} 0, & 2 \\ \frac{1}{2}, & -1 \end{pmatrix}$$

will be recorded thus: * || ⊛ | / || * — |.

For the future it is useful to note that ordinary recordings of rational letter expressions can be regarded as words in an alphabet consisting of letters (let us say, small Latin letters), of an alphabet for numbers, of the sign +, —, ×, : and the brackets, (,).

The reader can easily clarify to himself how it is possible to code also other mathematical objects. (Real and complex numbers, as they are understood in the classical analysis, continuous functions, and other objects of analysis cannot be coded in a similar manner. In order to make such coding possible, it is necessary to narrow these concepts. The problems pertinent to this are dealt with in a new mathematical field called constructive analysis. We shall not touch upon these problems here despite the fact that they are very real also for the theory of algorithms and will no doubt play a large role in its further development.)

By algorithm is understood the generally accepted and fully well-defined instruction how in what sequence to process the recording of the original information in order to obtain recordings of the final information -- the sought for solution. Such algorithms are, for example, the rules of addition or multiplication of natural numbers recorded in the decimal system, rules of extracting the square root, multiplication of matrices, rearrangements of entire rational expressions by removal of the barackets and reduction of homogeneous members (just as we are taught in school) many others. All these rules are such that in their formulation one encounters exclusively the characteristics of words and not the objects defined by them, i. e., exclusively the syntactic characteristics. Nevertheless, the sense of the expression "generally accepted instruction" is insufficiently clear and without further refinement cannot serve as the basis of purely automatic action. It is natural to require that any such instruction should represent a sequence of descriptions of certain, sufficiently elementary actions each of which could actually be accomplished automatically and, in particular, could be performed by a certain machine. Refinements of such a type were proposed by different authors; one of the best known of these pertinent algorithms -- "Turing machine" -- represents a system of action proposed by A. A. Markov and designated by him as the "normal algorithm" (abbreviated n. a.). This is what it involves.

The operation of connection has been defined for the words: the connection of the given words P and Q -- this is a word obtained by adding on the right the word Q to the word P; this is designated by PQ. It is said that the word A enters into the word X, if X is represented as the union of the type

$$X = YAZ. \qquad (1)$$

For the word X there are possible, generally speaking, such different representations: ih accordance with this, one speaks about the different entries of A into X. The entry is called the first if the word Y is the shortest for all possible representations of the type (1).

As an elementary operation for n. a., A. A. Markov selects the substitution of a certain given word B instead of the first entry of the given word A (this operation is recorded $A \to B$). More accurately, if $X = YAZ$ (with the shortest Y), then the application of the operation $A \to B$ to the word X, changes the latter into the word YBZ.

N. a. in its turn is determined by the system which consists of the sequence of formulas of the type

$$(\mathfrak{A})\begin{cases} A_1 \to B_1, \\ A_2 \to B_2, \\ \vdots \\ A_n \to B_n. \end{cases}$$

This system describes the rearrangement of the given word X according to the following rule: (a) there is such a first (counting up from below) formula that its left portion enters into the word under examination $X = X_1$; let this be the formula $A_i \to B_i$; then (b) instead of the first entry A_i in X_1 B_i is substituted; a certain word X_2 is obtained; (c) one proceeds with the word X_2 just as with the word X_1, etc; (d) if at a certain k-th stage not one of the left portions of the formulas of the system enters into X_k, then the process ceases and X_k is considered the result of the application of the n. a. $\mathfrak{A}$ to the word X.

A certain complication of the system of the n. a. is foreseen: some formulas of the system are noted as conclusive; they are recorded as $A_i \to . B_i$. In the description of the action of n. a. with a system containing conclusive formulas, it is agreed that the algorithm ruptures as soon as a conclusive formulas is applied and the resulting word is considered the result of the action of the algorithm.

Example 1. The n. a. for the calculation of the absolute value of the difference of two natural numbers. The purpose of the algorithm is to process the recording of a pair of natural numbers $n*m$, where n and m are the recording of the numbers in the alphabet $\{|\}$, into the recording of the absolute value of their difference. The corresponding n. a. is given by the system

$$\begin{cases} |*| \to *, \\ * \to \Lambda. \end{cases}$$

Example 2. Discernment of correctly constructed rational letter expressions.

The alphabet consists of a certain number of Latin letters, for example, a, b, c and of the following signs: $+, -, \times, (,), \circ$. A correctly constructed formula is determined inductively, as a word in this alphabet:

(1) a blank word and single-letter words a, b, c, are in essence the formulas;

(2) If A and B are in essence the formulas and $A \neq \Lambda$, $B \neq \Lambda$, then the words $(A+B)$, $(A-B)$, $(A \times B)$ are the formulas;

(3) a word is a formula only when it is in accordance with (1) and (2).

It is necessary to find the n. a. which discerns

words that are formulas, i. e., such a n. a. which processes word-formulas (and only these) into a blank words.

An example of the sought-for n. a. is the following:

$$\begin{cases} b \to a, \\ c \to a, \\ + \to \circ, \\ - \to \circ, \\ \times \to \circ, \\ (a \circ a) \to a, \\ a \to \Lambda. \end{cases}$$

We recommend to the reader to clarify to himself with the examples the action of this n. a. and to prove that it precisely solves the given problem.

According to the Markov principle of normalizations, any algorithm is equivalent to a certain n. a.; in other words, if for the solution of a certain problem, it is possible to point out also the n. a. which leads to the same solution. On the other hand, it is not difficult to point out an algorithm which changes the system of the n. a. into a program of ordinary command-address RCM's. (Such an algorithm was pointed out by I. N. Kovalenko in an obvious form.) Thus, in principle, the language of the Markov n. a. can be selected as the standard language for the algorithmization of problems. But only in principle -- in practice this language, in the form in which it was developed by A. A. Markov, is entirely not suitable. One can easily become convinced of this with examples of the construction of even very simple arithmetic and non-arithmetic problems. We shall indicate the basic shortcomings of the n. a. which occurred in the attempt to utilize them for the solution of the problems of interest to us; in the next paragraph we shall point out those changes which are projected for the elimination of these shortcomings. It seems to us that there are three substantial shortcomings; these do not depend on one another.

(1) The Markov n. a. does not have the concept of "memory" and thereby the n. a. for the solution of any kind of problem is by its structure very far from the corresponding program for command-address RCM's. The change of the n. a. to such a program is fundamentally possible as this was already pointed out, but in practice it turns out to be rather complex while the resulting program is practically unrealizable due to the limitation of the operative memory of the RCM's.

(2) Then n. a. is not economical; even for entirely non-complex problems the corresponding n. a. become excessively cumbersome; their systems contain hundreds and thousands of formulas of substitutions and it is easy to become convinced that intermediate words of very great length will occur in the course of the work of the algorithm.

(3) Complexity of construction of compositons for the n. a. In the construction of an algorithm for the solution of a certain, sufficiently complex problem in any kind of given language, it is natural to dismember the problem into various, more simple, more elementary ones and to construct algorithms for these latter and to shape the sought-for algorithm for the entire problem as a certain system of successive applications of "partial algorithms." From the compilation of such a system it is natural to start algorithmization in any language; in particular, this is the procedure in compiling the program for RCM's. One of the most important requirements, which it is necessary to impose on a standard language, is that the final algorithm for the solution of a complex problem should be constructed simply and economically from partial algorithms. The language of the n. a. satisfies this requirement in an entirely insufficient measure. Fundamentally, this problem is solvable and the greater portion of the monograph by A. A. Markov deals with the solution of the corresponding questions. In particular, the following problems are solved:

(a) Let the two n. a. $\mathfrak{A}$ and $\mathfrak{B}$ be given. It is possible to construct the n. a. $\mathfrak{C}$, equivalent to $\mathfrak{AB}$, i. e., it leads always to the same result as the algorithm and consists of the application of the algorithm to the given word and of the algorithm $\mathfrak{A}$, to the result of the action of the latter $\mathfrak{B}$.

(b) Let $\mathfrak{A}$ and $\mathfrak{B}$ be two n. a. It is possible to construct an n. a. which processes any word X into a pair of words $\mathfrak{A}(X)*\mathfrak{B}(X)$, where $\mathfrak{A}(X)$ and $\mathfrak{B}(X)$ is the result of the application, respectively, of $\mathfrak{A}$ and $\mathfrak{B}$ to the words X and $*$ is a certain separatory sign.

(c) Let $\mathfrak{A}, \mathfrak{B}, \mathfrak{C}$ – be three n. a. and $\mathfrak{L}$ – is a certain "discerning" n. a., i. e., and n. a. which processes the given words X into a blank or non-blank word depending on whether the word X does or does not have a certain syntactic property W.

Then, it is possible to construct the n. a. $\mathfrak{D}$, which acts as an algorithm of the following type: at first the algorithm $\mathfrak{A}$ acts on the given words X ; if the result $\mathfrak{A}(X)$ has the property W then it is processed by means of $\mathfrak{B}$, if not, then in accordance with $\mathfrak{C}$.

(d) Finally, the n. a. $\mathfrak{C}$, is constructed which

fulfills the repetition of the given algorithm $\mathfrak{A}$ until a certain property W appears and which is discernible by a certain n. a. $\mathfrak{B}$.

By means of the idicated procedures, it is possible in principle to solve the above indicated problem of combining the simple n. a. into complex ones in accordance with the given system. But in practice the sought-for complex n. a. becomes rapidly very cumbersome and in the course of its work new words of great length appear; besides, the compilation of a complex n. a. in accordance with the given procedures becomes practically unrealizable very rapidly.

3. Algorithms which are Determined by Graph-Systems

1. Formal Description of a Graph-System

By graph-system Γ we understand a mathematical object described in the following manner:

A certain finite selection of objects is given

$$\mathfrak{A} = \{\mathfrak{A}_1, \mathfrak{A}_2, \ldots, \mathfrak{A}_n\},$$

these are called operators, and a certain second finite selection of objects called discriminators

$$\Phi = \{\Phi_1, \Phi_2, \ldots, \Phi_m\}.$$

The graph-system Γ or, more accurately, $\mathfrak{A}$-Φ- graph-system (for the given selections of operators $\mathfrak{A}$ and discriminators Φ) -- is a finite, connected and directed linear complex, i. e., a finite number of points (joints) some of which are joined by directed segments (arrows) and such that, starting from any point, it is possible to reach any other, following along these segments (not mandatorily always in the direction of the arrow) and which satisfied the following conditions:

(1) One joint of the complex is marked and called the input (B); the input is the only joint in which not a single arrow ends; exactly one arrow emanates from the input.

(2) The complex has one noted joint called the output (a); not one arrow emanates from the output.

(3) For each joint which differs from the input and output is compared in a well-defined manner either with a certain operator $\mathfrak{A}_i$ --such a joint will be called a $\mathfrak{A}$-joint -- or a discriminator Φ_j; ; then we will speak about the Φ- joint (it is not necessary that each operator $\mathfrak{A}_i$ and each discriminator Φ_j be compared with a certain joint;

on the other hand, certain operators or discriminators can be compared with several different joints). Moreover:

(a) if α is a $\mathfrak{A}$--joint, then only one arrow will emanate from it;

(b) if α is a Φ--joint, then exactly two arrows will emanate from it; these are noted, respectively, by plus and minus.

This ends the formal description of the graph-system. We shall now pass on to the sapid description of the $\mathfrak{A}$-Φ-algorithm given by the graph-system.

2. Theoretical plural interpretation

For the given selections $\mathfrak{A}=\{\mathfrak{A}_1, \mathfrak{A}_2, \ldots, \mathfrak{A}_n\}$ and $\Phi=\{\Phi_1, \Phi_2, \ldots, \Phi_n\}$ of each $\mathfrak{A}$-Φ- -graph-system can be compared in a well-defined manner with a "algorithm" for the calculation of a certain function if one interprets the operators $\mathfrak{A}_i$ as reflections of a certain mass in itself and the discriminators -- as operations which discern certain characteristics of elements and accomplish the application of some or other reflections to the element depending on whether this element has the given characteristic or not.

We shall give a strict determination of the $\mathfrak{A}$-Φ- -algorithm for the given interpretation.

Let $\mathfrak{A}=\{\mathfrak{A}_1, \mathfrak{A}_2, \ldots, \mathfrak{A}_n\}$ and $\Phi=\{\Phi_1, \Phi_2, \ldots, \Phi_m\}$ be the given selection of the operators and discriminators. By interpretation of these selections is meant the following: a certain mass M is given; each operator $\mathfrak{A}_i \in \mathfrak{A}$ is compared with the reflection A_i of the mass M in itself (the comparison is not mandatorily mutually well-defined; it is assumed that the same reflection corresponds to different operators $\mathfrak{A}_i$); for each discriminator $\Phi_j \in \Phi$ there corresponds a certain characteristic F_j of the elements of the mass M. Moreover, as is usual, by the characteristic of F_j of the elements is understood the break-up of the mass M into two non-intersecting submasses: Fj (its elements are called elements having the characteristic F_j) and $\overline{Fj}$ (the mass of elements not having the characteristic F_j); one of the masses Fj or $\overline{Fj}$ can be blank. The comparison of the discriminator Φ_j with the characteristic of F_j is not mandatorily mutually in a well-defined manner.

The interpretation of the selections $\mathfrak{A}$ and Φ is determined by the assignment of the mass M and the conformities $\mathfrak{A}_i \to A_i$; $\Phi_j \to F_j$. We shall designate such an interpretation by $\{M; \mathfrak{A} \to A; \Phi \to F\}$.

Let a certain interpretation $\{M; \mathfrak{A} \to A; \Phi \to F\}$ be given. Then, any $\mathfrak{A}$-Φ- graph-system can be regarded as a well-defined instruction for the fulfillment of a certain

action, as a result of which certain elements from M will change into new elements from M.. We accomplish this action thus:

Let the element $m \in M$ proceed to the input of the system; after this, it runs through the system, following the arrows, changing every time it passes through a $\mathfrak{A}$--joint. This run takes place in accord with the following rule.

(a) It is assumed that the starting element m has already passed into a certain element m' and, following the arrow, has entered into the joint a, compared with the operator $\mathfrak{A}_i$; then, the element $m' = A_i(m')$ continues its path along the only arrow which emanates from a

(b) Let, as above, the rearranged element m' proceed into a certain joint a, noted by the discriminator Φ_j; then the same element m' abandons this joint along the arrow noted by a plus or minus, depending on whether m' has or does not have the characteistic of F_j.

(c) If at a certain stage the changed element $\overline{m}$ proceeds to the output, then we will consider that the $\overline{m}$ is the result of the application of the $\mathfrak{A}$-Φ- algorithm determined by the $\mathfrak{A}$-Φ- graph-system Γ for the interpretation $\{M; \mathfrak{A} \to A; \Phi \to F\}$, and we will make use of the recording $\overline{m} = \Gamma(m)^*$). (Since we used the letter Γ to designate the graph-system, while the determination of the corresponding algorithm includes, besides this, the assignment of interpretation, then the designation $\Gamma_{(m)}$ generally speaking has no sense; but for the sake of brevity, we will sometimes drop the reference to the interpretation when such carelessness in the recording cannot lead to a misunderstanding.).

It is easy to explain to oneself that for the given interpretation the graph-system describes in a well-defined manner the function determined in M, with the values in M. We wish to underscore that, generally speaking, an algorithm thus determined does not lead for each $m \in M$ to a final result: in the general case, the function is determined in M, but not on M.

The very trivial example of such a phenomenon is given by the following graph-system: the mass of operators $\mathfrak{A}$ is blank, the mass of discriminators consists of one element Φ (Fig 1). It is quite obvious that for any interpretation of $\Phi \to F$ the corresponding algorithm will change each element of $m \in M$, which has the characteristic F, into itself, whereas for each element which does not have the characteristic of F the action of the algorithm will never be interrupted and will not lead to any result. In the second case we will state that the corresponding algorith Γ is not applicable to the element m and we shall consider the value of $\Gamma(m)$ indeterminate.

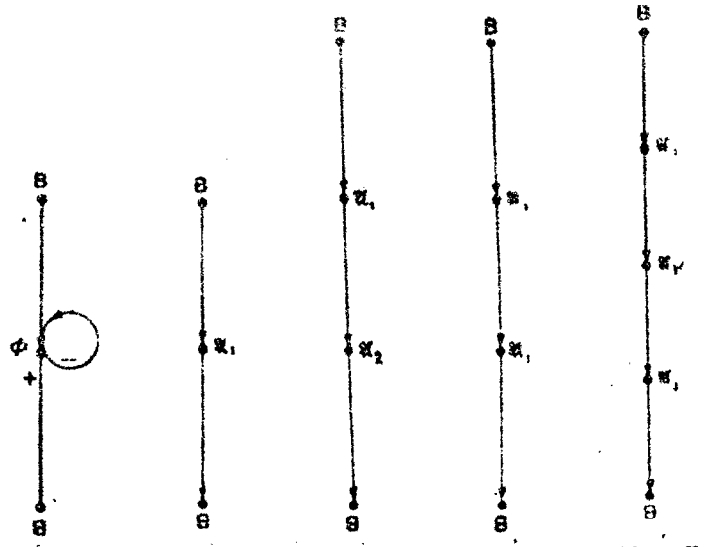

Fig 1 Fig 2 Fig 3 Fig 4 Fig 5

We shall give several of the most simple examples of algorithms described by graph-systems:

(1) "To apply operator $\mathfrak{A}_1$!" (Fig 2).

(2) "To apply operator $\mathfrak{A}_1$, while to the result -- the operator $\mathfrak{A}_2$!" (Fig 3), in particular "the iteration of operator $\mathfrak{A}_1$" binary (Fig 4), ternary (Fig 5) etc.

(3) Conditional iteration -- "To apply operator $\mathfrak{A}_1$ until the element acquires the characteristic of F_1!" (Fig 6).

(4) "To apply to the element $m \in M$ the operator $\mathfrak{A}_1$ or $\mathfrak{A}_2$ depending on $m \in F_1$ or $m \in F_1$!" (Fig 7).

The reader can compile easily such and more complex examples.

We wish to note that by algorithm we understand to action described by the graph-system and not the function in M which is accomplished by the application of these rules. It is easy to explain to oneself that for the given interpretation it is possible that the algorithms described by different graph-systems will for any $m \in M$ lead to the same results, i. e., different graph-systems can accomplish the same function; then we shall call the given algorithms equivalent. More accurately: let $\{M; \mathfrak{A} \to A; \Phi \to F\}$ — be a certain interpretation of the selections $\mathfrak{A}$ and Φ. Two algorithms described by the $\mathfrak{A}$-Φ-graph-systems Γ_1 and Γ_2 are considered equivalent for the given interpretation of $(\Gamma_1 \approx \Gamma_2$ for $\{M; \mathfrak{A} \to A; \Phi \to F\})$, if for any $m \in M$ we have $\Gamma_1(m) = \Gamma_2(m)$: the latter equality should be understood in the sense that both elements $\Gamma_1(m)$ and $\Gamma_2(m)$ are either identical or both elements $\Gamma_1(m)$ and $\Gamma_2(m)$ are not determinate. This determination of equivalence depends,

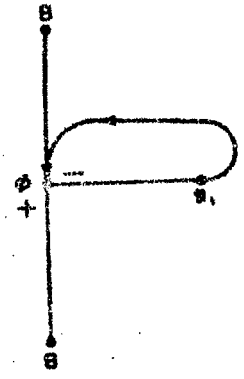

Fig 6

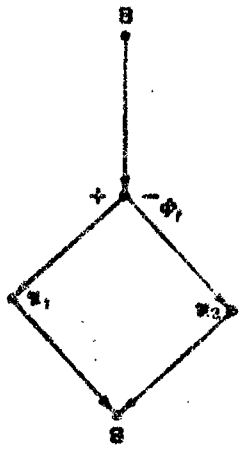

Fig 7

obviously, on the very graph-systems of Γ_1 and Γ_2, as well as on the given interpretation of $\{M; \mathfrak{A} \to A; \Phi \to F\}$.

One can easily point out the selections $\mathfrak{A}, \Phi$ and such of their interpretations that the establishment of equivalence of the algorithms corresponding to different graph-systems turns out to be an algorithmically unsolvable mass problem (see [4]). Actually, as we shall see, in definite interpretations certain $\mathfrak{A}$-Φ- -algorithms can be reduced to Markov n. a. (for other interpretations -- to Turing machines or, finally, to partial-recursive functions); the problem of equivalence for the $\mathfrak{A}$-Φ- algorithms reduces itself then to the corresponding problem for n. a. the latter, however, as is known, generally speaking is algorithmically unsolvable.

Along with such a determined equivalence it is natural to regard a stronger equivalence. Namely, we will state that the graph-systems Γ_1 and Γ_2 are strongly equivalent ($\Gamma_1 \approx \Gamma_2$), if for any interpretation their corresponding algorithms are equivalent in the above indicated sense. Strong equivalence is the equivalence of the graph-systems themselves; it does not depend on the interpretation. The establishment of strong equivalence of two $\mathfrak{A}$-Φ- graph-systems is a more simple problem. It is analogous to the corresponding problem regarded by Yu. I. Yanov [5]; we are planning to return to this question shortly. In the meantime, it is clear that the establishment of ordinary equivalence is practically more important; this problem, in particular, naturally arises in searching for the "most advantageous" in any reasonable sense) algorithm for the solution of one or another problem.

For a graph-system it is possible to introduce the

operation of "substitution of a graph-system into a graph-system." This operation makes it possible in an entirely natural manner to accomplish the composition of algorithms and opens the possibility of developing a unique "algebra of algorithms." The problems pertinent to this have so far only been noted. But from the very first steps, it is already seen that it will possible to attract to the solution of the occurring problems various sapid and already sufficiently developed mathematical disciplines: in the first place, the algebra of logic, the theory of proportions, the narrow calculation of predicates, and, perhaps, the theory of models, the theory of recoursive functions, etc.

The operation of substitution is determined thus: let the selections $\mathfrak{A}=\{\mathfrak{A}_1, \mathfrak{A}_2, \ldots, \mathfrak{A}_n\}$ and $\Phi=\{\Phi_1, \Phi_2, \ldots, \Phi_m\}$ be given and let there be two $\mathfrak{A}\cdot\Phi$-graph-systems Γ_1 and Γ_2. We shall separate in Γ_1 the entries of one of the operators $\mathfrak{A}_i$ and substitute instead of each such entry the entire graph-system Γ_2 (having eliminated, of course, in the latter its input and output). The substitution of the system Γ_2 instead of the operator $\mathfrak{A}_i$ will lead anew to the $\mathfrak{A}\cdot\Phi$-graph-system.

It would be logical and useful to introduce the operation of the substitution of the graph-systems in place of the Φ_i-joints, but on the basis of the concepts introduced up to now, it is not possible to give such a determination. For this it would be necessary to expand somewhat the apparatus of the graph-system. We wish to point out only two natural paths in this direction.

In any case, it is necessary to expand the concept of graph-systems. We shall a graph-system of the second type an object determined just as this was done in point 1 in the formal description of the graph-system, with only the difference that in point (2) not one but two outputs β' and β'' are specified; determinations of the interpretation and action of the algorithm according with the graph-system also remain unchanged.

Algorithms described by graph-systems of the second type could accomplish the operation of "discrimination" of the characteristics of the elements $m \in M$ on the basis of the action of the given operators $\mathfrak{A}$ and Φ_i. For the graph-system Γ of the second type it would have been possible to determine in a natural manner the operation of substitution of such a system instead of the entry of discriminators. There are no difficulties at the level of the formal description of the graph-systems.

It is another matter at the level of the sapid description of the algorithm; here, a substantial non-conformance is obtained. Actually, if the graph-system Γ of the

second type is substituted instead of the entry of the discriminator Φ_i, then in discriminating the characteristics of the element m fed to it, it will feed to the outputs a^+ or a^-, generally speaking, not the element m, but the element $\Gamma(m)$, rearranged by it. Thus, the action of the algorithm described by the graph-system Γ of the second type does not fully correspond to the sapid sense of the discriminator Φ_i.

In order to eliminate the non-conformance which takes place, two paths are projected:

(1) One can expand the sapid sense of the discriminators Φ_i, assuming that in the interpretation of Φ_i joint accomplishes simultaneously also the discrimination of a certain characteristic and the rearrangement of the element entering therein in accordance with a certain reflection. Such a modification of the sense of the discriminator would make it possible to eliminate the above indicated non-conformance without changing thereby the formal determination of the graph-system. But this complicates substantially the sapid description of the action of the algorithm in accordance with the given graph-system and thereby disturbs the logical order of the theory.

(2) The second path which is suggested and which seems to us more promising could consist of the introduction into the graph-system, along with the already determined elements (input, output, joints, and arrows) of certain new elements which accomplish the storage of information (i. e., at the given level of the sapid description -- which retain certain elements from M developed in the course of action of the algorithm) and which transmit this information in accordance with definite laws for definite stages of the action in certain joints. Then the algorithm which corresponds to the graph-system of the second type could be supplemented by a similar element of memory so that, as a result of the work of the algorithm, the original element m would be restored at the output a^+ or a^- depending to which of these two outputs the result of the action of the algorithm in accordance with the graph-system proceeded.

The most rational method of introducing such elements of memory into the graph-system, moreover, in such a manner as to retain the possibility and sense of the operations of the substitutions, is an important but not yet solved question. We propose this question to the attention of the reader, while, because of the above indicated difficulties, we refrain from the introduction of graph-systems of the second type.

3. Constructive Markov Interpretations of Graph-Systems

$\mathfrak{A}$-Φ- algorithms, in the form in which they were described in the preceding heading, can be regarded as "conditional;" they are realizable to the extent to which the reflections of A_i are realizable for the given interpretation, while the characteristics of F can actually be discerned. For the actual accomplishment of an algorithm, the interpretation should be so selected that the operations $\mathfrak{A}_i - \Phi_j$ (we shall call these elementary) will be so simple that no doubt will arise as to the possibility of their accomplishment. More than that, if the algorithm is intended for the fulfillment in a certain automatic device, then it is natural to require for the elementary operations to be capable of being accomplished by certain, standard simple cells of the machine.

These requirements, because of their very essence, do not permit a strict mathematical interpretation: they pertain to those original data which, from the point of view of mathematics, are not subject to consideration (this does not mean that they are not material to mathematics). We do not intend to present here in detail the considerations pertinent to this and refer the reader who is not familiar with this to the extensive literature on this problem (Cliny [2], § 70, Markov [4], Turing [7]).

Several such interpretations can be selected and, depending on which we select, we come either to the language of n. a. by Markov or to the Turing machines or even to partial-recursive functions. In this paper we will touch upon only the first two interpretations. This is all the more natural in that it is precisely the language of n. a. by Markov and its critique that were precisely for us the point of departure for the development of the $\mathfrak{A}$-Φ- algorithms.

We shall determine the concepts of the Markov interpretations and the Markov algorithms (not mandatorily normal).

Let $\mathfrak{A} = \{\mathfrak{A}_1, \mathfrak{A}_2, \ldots, \mathfrak{A}_n\}$ and $\Phi = \{\Phi_1, \Phi_2, \ldots, \Phi_m\}$ — be the selections which determine the graph-system. We shall designate by F_X the characteristic of words (in a certain alphabet) to contain at least one entry of the word X, and by $A_{X \to Y}$ — the operation of substitution of the word Y instead of the first entry of the word X (if X does not come out in the word under consideration, then it is not rearranged by the operation $A_{X \to Y}$). Sometimes, we will write $[X]$ instead of F_X and $[X \to Y]$ instead of $A_{X \to Y}$.

The interpretation $\{M; \mathfrak{A} \to A; \Phi \to F\}$ is called a Markov interpretation if:

(1) M is a mass of all possible words in a certain alphabet;

(2) all the F_i are in essence the characteristics of the type F_X;

(3) all the A_i are in essence operations of the type $A_{X \to Y}$.

Any algorithm described by a certain graph-system with a certain Markov interpretation will be called a Morkov algorithm. The n. a. are a specific case of Markov algorithms in the above determined sense. And precisely, let for example

$$\begin{cases} X_1 \to Y_1, \\ X_2 \to .Y_2, \\ X_3 \to Y_3, \\ X_4 \to Y_4 \end{cases}$$

be a system of n. a. ("." in the second formula indicates that this formula is conclusive, Then the instruction by A. A. Markov regarding the action of the algorithm according to this system is equivalent to a $\mathfrak{A}$-Φ- algorithm which is accomplished according to the graph-system (Fig 8), where $\mathfrak{A}_i \to [X_i \to Y_i], \Phi \to [X_i]$.

It is easy to see that the n. a. will always correspond to the Markov algorithms with graph-systems of a similar specific type (a detailed formal description of such graph-systems is presented to the reader). But on the other side, the above indicated theorems by A. A. Markov show how easy it is to explain to oneself that any Markov algorithm in our sense is equivalent to a certain algorithm described by a normal graph-system (i. e., such which corresponds to the n. a.). Thus, the normal graph-systems can be regarded as certain canonical forms in Markov interpretations, while the indicated theorems -- as a partial solution of the problem of equivalence for Markov algorithms (precisely partial because different n. a. can be equivalent while the solution of the problem of equivalence for these is, generally speaking, a problem that is algorithmically unsolvable).

The methods indicated in these theorems for the construction of complex n. a., starting from more simple ones, actually turn out to be sufficiently cumbersome. In the meantime, the construction of Markov algorithms described by graph-systems can be accomplished sufficiently simply and the constructed algorithms are substantially more economical than the corresponding n. a. (we were convinced in this in the practice of algorithmization of different mathematical and logical problems). Besides this, the construction of such algorithms follows the ordinary methods of the

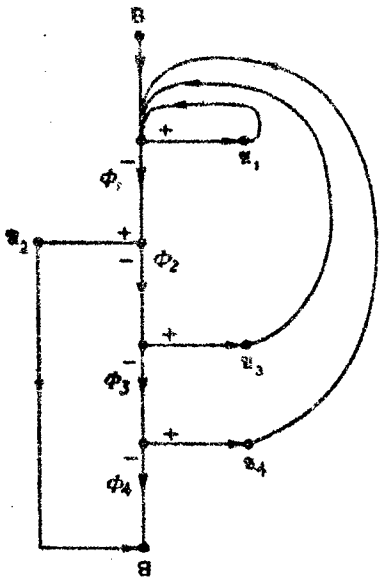

Fig 8

solution of problems. The problem is divided into more simple problems for which the construction of n. a. is sufficiently simple, then a graph-system is constructed the joints of which correspond to the fulfillment of the algorithms for the partial problems.

And so, the Markov algorithms are based on the interpretation in which the elementary operations consist of the discrimination of the entry (or non entry) of a certain given word into the word which is subject to processing and in the substitution of certain words in the place of other words. From the point of view of the accomplishment of such an angorithm on the solving device, these operations are obviously insufficiently simple; for this purpose, these should, in turn, be divided into more simple ones. A detailed analysis of this problem shows that it is possible (and natural) to select the following as elementary operations: discrimination, whether a certain letter under consideration is graphically equivalent to a certain given letter or not; notation of the letter under consideration; substitution of a given letter instead of that under consideration; change from the letter under consideration to the neighboring one to the right or to the left. Algorithms described by graph-systems in interpretations with elementary operators of the above indicated type will be called Turingian; they are somewhat changed descriptions of the Turing machines [7]. The stablishment of the convergence of Markov algorithms and, in particular, of the n. a. to

the Turingian ones and conversely is in essence new proof of the equivalence of the n. a. and the Turing machines [7]. On the other hand, this information makes possible practical automatic conversion of Markov algorithms into programs of command-address RCM's.

True, it is necessary to point out that the accomplishment of the Markov algorithms on command-address RCM's is excessively complex and in most cases is practically unrealizable. In essence, it is matter of the solution on RCM's of combinatorial problems and here we adhere to opinion by C. Shannon [6] who writes: " for combinatorial problems, it seems typical that the colossal gain in speed is possible by the utilization of specialized machines instead of universal calculating machines of dicrete action. This means that universal machines actually are not universal in use but are specialized in the direction which favors the problem of analysis. Logically it is no doubt true that the so-called universal machines are capable also of solving combinatorial problems but their output is very low in such use. Problems connected with the planning of universal machines adapted for a wide field of combinatorial problems are in all evidence very complex but they represent no doubt great theoretical interest.

We hope that a deeper study of the questions connected with the algorithmization of problems will facilitate the development of universal machines of dicrete action, which will be capable of solving such problems in a more economical manner.

Bibliography

1. Detlovs V. K. Normal algorithms and recursive functions. DAN [Reports of the Acad. Sci.] 90, No 3 (1953), 249-252.
2. Cliny St. K. Introduction to mathematics. Moscow, Publishing House of Foreign Literature, 1957.
3. Lyapunov A. A. Logical systems of programs. Problems of cybernetics (collection of papers), 1, Moscow, Fizmatgiz, 1958.
4. Markov A. A. Theory of algorithms. Works of the Mathematical Institute of the Acad. Sci. USSR, XLII (1954).
5. Yanov Yu. I. Logical systems of algorithms, Problems of cybernetics (collection of papers), 1, Moscow, Fizmatgiz, 1958.

6. Shannon C. E., Moore E. F. Machine Aid for switching Circuit Design. Proc. of the IRE (Computer issue) 41, No 10 (1953), 1348-1351.
7. Turing A. M. On computable numbers. Proc. London Math. Soc., 1937 (2) 42, pp 230-265; V 43, p 544.

Received by Editorial Office

17 October 1957

END