REAL-TIME RECOGNITION OF THE INCLUSION RELATION*

Yu. V. Matiyasevich

Let $\mathfrak{A}$ be a certain alphabet of more than one letter. We denote by $P \succ Q$ the predicate "Word Q is included in word P" (we use upper-case Latin letters throughout the article to signify variables for words in alphabet $\mathfrak{A}$).

The problem of deciding the truth of the predicate $\succ$ has various interpretations. For example, we could say that we have a glossary P and are seeking a particular word Q that we require in it, or that we have a corpus of text P and are required to test for the occurrence in it of a given key word Q. Of major interest in this connection is the recognition of the predicate $\succ$ in real time. We can distinguish several problems differing in the degree of accessibility of information on words at each instant of time.

Thus, Freidzon [1] has considered the problem of deciding the truth of the predicate $P \succ Q$ for the case in which word $P * Q$ is received at the input, where $*$ is a separative letter not included in alphabet $\mathfrak{A}$. This problem, as shown in [1], is unsolvable in real time on multitape Turing machines with input.

Morrison has investigated the problem of deciding the truth of the one-argument predicate $P \succ Q$ for a fixed word P. This problem can be solved in real time on finite automata, which are easily synthesized from the given word P (see [2]).

We wish to consider the following two problems.

A. The real-time recognition of the truth of the one-argument predicate $P \succ Q$ for a fixed word Q.

B. The real-time recognition of the truth of the predicate $P \succ Q$ with the simultaneous reception of words P and Q at two inputs.

These problem statements are natural for the case in which the word P (glossary, corpus of text) is stored in memory as a one-dimensional bank and is extracted therefrom letter by letter.

The fundamental result of the present article is a theorem stating that Problem B is solvable on a Turing machine with a two-dimensional tape (the precise meaning of this concept is spelled out below). The machine we seek is synthesized in three steps.

*The results of this article were reported at the Leningrad Seminar on Constructive Mathematics on May 15, 1969.

Translated from Zapiski Nauchnykh Seminarov Leningradskogo Otdeleniya Matematicheskogo Instituta im. V. A. Steklova Akademii Nauk SSSR, Vol. 20, pp. 104-114, 1971.

© 1972 Consultants Bureau, a division of Plenum Publishing Corporation, 227 West 17th Street, New York, N. Y. 10011. All rights reserved. This article cannot be reproduced for any purpose whatsoever without permission of the publisher. A copy of this article is available from the publisher for $15.00.

We first specify how to synthesize a finite automaton that solves Problem A. We then show that the operation of this automaton can be simulated by a certain Turing machine, on whose two-dimensional tape is written the schema of this automaton. Finally, we show that the schema of the given automaton can be synthesized by another Turing machine in a sufficiently short time. The desired machine is obtained by the unification of these two machines.

We shall assume without loss of generality that alphabet $\mathfrak{A}$ consists of the two letters 0 and 1. We also introduce the following notation:

$$\bar{0} \leftrightharpoons 1, \quad \bar{1} \leftrightharpoons 0.$$

If a word X is representable in the form YZ, where Y and Z are words, we say that Y is an X-beginning and Z is an X-ending.

We denote by the letters m and n the lengths of the respective words denoted by the letters P and Q, and we use $p_i (1 \leqslant i \leqslant m)$ and $q_i (1 \leqslant i \leqslant n)$ to denote the i-th letter of the words P and Q, respectively.

1. We now describe the finite automaton $\mathfrak{A}_Q$ that solves Problem A. This automaton has $n+1$ states $a_1, a_2, \ldots, a_{n+1}$. State a_1 is the initial state. From state a_{n+1} the automaton transfers to the same state independently of the input signal. From state $a_i (i \leqslant n)$ the automaton transfers to state a_j when the letter $x (x \in \{0, 1\})$ is received at the input, where j is the length of the longest Q-beginning that is an ending of the word $p_1 p_2 \ldots p_{i-1}$. In state a_{n+1} the automaton generates at the output the letter И ("true") independently of the input signal. In state a_n the automaton generates the letter И at the output when the letter p_n is received at the input. In all other cases the automaton generates the letter Л ("false") at the output.

THEOREM 1. The automaton $\mathfrak{A}_Q$ generates the letter И at the output if and only if the word received at its input includes the word Q.

The proof of the theorem is based on the following lemma.

LEMMA 1. For any word P including the word Q, possibly only as its ending, when P is received at the input the automaton transfers to state a_j, where j is the length of the longest Q-beginning that is simultaneously a P-ending.

The lemma is easily proved by induction on the length of P.

2. Before describing the Turing machine that solves Problem A we need to refine the concept of a Turing machine with inputs and outputs, a two-dimensional tape, and several scanners.

The external memory of the machine is represented as a plane divided into square cells. We shall assume for convenience in describing the operation of the machine that all the cells are enumerated in a natural way by pairs of integers.

It is normally assumed that no more than one letter from a certain fixed alphabet can be written into each cell of the Turing machine tape. We also include in the description of the machine a certain alphabet Ю, called the internal alphabet of the given machine.

It will be more to our advantage, however, rather than to speak of a particular letter being written in a particular cell, to say that a particular cell is labeled with a particular letter. In this case each cell can be labeled with several different letters at once, or with none. It is clear that in order to return to the traditional variant it is sufficient to introduce a new alphabet having 2^S letters, where S is the number of letters of alphabet Ю.

One or more scanners are located on the tape. It is considered that each scanner is situated in one of the cells at any time. We also say that a scanner scans the cell in which it is situated. One cell can be simultaneously scanned by several scanners at once.

The machine operates in discrete time. After one operating cycle each scanner can label the cell scanned by it with one or more letters, and it can also "erase" earlier labels. The scanner can then remain in the same cell or move to one of the eight neighboring cells (each coordinate of which differs at most by unity from the corresponding coordinate of the scanned cell).

At each instant the machine is in one of a finite number of internal states.

Besides the internal alphabet we also have an input alphabet and an external alphabet. The operating cyles of the machine are divided into principal and auxiliary cycles. During a principal cycle one letter from the input alphabet enters each of one or more inputs of the machine, and the latter generates one letter from the output alphabet at each of the outputs.

The total state of the machine in an auxiliary cycle consists of:

a) an internal state;

b) a list (possibly empty) of all pairs of scanners that are situated in the same cell;

c) a list containing complete information on which letters are being used to label all cells scanned in the given cycle.

The total state of the machine also contains the following during a principal cycle:

d) a list of the letters received at each input.

The machine is specified by a table of transitions. This table has two columns; the left column lists all possible total states of the machine, and the right column contains the following:

a) complete information on how the labeling of the scanned cells is to be changed;

b) complete information on whether each scanner is to be shifted and, if so, where it is to be shifted;

c) the new internal state;

d) a list of the letters generated at each output (for the case of principal cycles).

In describing the machine, however, we shall not use the complete transition table, but an "abridged" transition table. It differs from the complete table in that the left column does not contain complete information on each of the possible states, but only partial information, which is nevertheless sufficient for filling in the right column unambiguously. In the right column, on the other hand, only changes are entered, i.e., the new state is not indicated if it coincides with the preceding one, the

movement of the scanner is not indicated if it stays in the same place, etc. Rows having an empty right side are eliminated from the table. The sumbols $\neg$ and $\vee$, as usual, are the negation and disjunction symbols.

We use the following notation. Notation of the form $\Lambda_1 : \Lambda_2 : \ldots : \Lambda_k \lambda_1 \lambda_2 \ldots \lambda_\ell$, where $\Lambda_1, \ldots, \Lambda_k$ denote scanners and $\lambda_1, \ldots, \lambda_\ell$ are letters of the internal alphabet, signifies that scanners $\Lambda_1, \ldots, \Lambda_k$ are situated in the same cell, which is labeled with the letters $\lambda_1, \ldots, \lambda_\ell$. Analogously, notation of the form $\Pi\pi$, where Π denotes an input and π denotes a letter of the input alphabet, signifies that the letter π enters the input Π. The movements of the scanners are denoted by arrows $\uparrow, \nearrow, \rightarrow, \searrow, \downarrow, \swarrow, \leftarrow, \nwarrow$, which have the logical suggested meaning. The arrows are placed to the right of the scanner symbols. Notation of the form $\lambda_1 \ldots \lambda_\ell \Lambda_1 \ldots \Lambda_k$ signifies that the cells scanned by scanners $\Lambda_1, \ldots, \Lambda_k$ are labeled in the given cycle with the letters $\lambda_1, \ldots, \lambda_\ell$. Moreover, the specific letters of the internal alphabet can be replaced by variables and an indication of their ranges of variation.

We shall assume that all cycles with order numbers of the form $i\nu+1$ are principal, where i is any nonnegative integer and ν is a fixed positive integer, called the cycle length of the given machine. We say about a machine having this sequencing of the principal cycles that it operates in real time. This terminology is justified by the following theorem.

THEOREM 2. For any Turing machine $\mathscr{L}$ operating in real time it is possible to synthesize a Turing machine $\mathscr{L}'$ having the same input and output alphabets and the same number of inputs and outputs, such that the cycle length is equal to unity (i.e., all cycles are principal) and such that for any words of the input alphabet, if they are received at the inputs of both machines, the same words are obtained at the outputs of both machines.

The theorem is easily proved by partitioning the external memory of machine $\mathscr{L}$ into large squares containing ν^2 cells each (ν is the cycle length of $\mathscr{L}$) and adopting as the external alphabet of $\mathscr{L}'$ an alphabet large enough that its letters can be used to enumerate the contents of the large squares. The operation of each scanner of machine $\mathscr{L}$ in this case is simulated by the operation of nine scanners of $\mathscr{L}'$ arranged in a 3x3 square.

We now describe a machine $\mathfrak{M}$ which can simulate the operation of the automaton $\mathfrak{A}_Q$. This machine has four states m_1, m_2, m_3, m_4; four scanners Δ, M, N, Ω; and one input Φ. The internal alphabet consists of five letters $0, 1, \alpha, \beta, \varepsilon$; the input alphabet of two letters $0, 1$; and the output alphabet of two letters H, Π. The cycle length is equal to 5. The initial state is m_1. The abridged transition table of machine $\mathfrak{M}$ is given below.

Operation of scanner M:

Φx	$x M \uparrow$

$x \in \{0, 1\}$

Operation of scanners Δ and N:

m_1 ; Δx ; $N x$	$\Delta\nearrow$; $N\uparrow$
m_1 ; $\Delta x \varepsilon$; $N \bar{x}$	m_2 ; $N\uparrow$
m_2 ; $\neg(\Delta\alpha \vee \Delta\beta)$	$\Delta\downarrow$
m_2 ; $\Delta\alpha$	m_3
m_2 ; $\Delta\beta$	m_4
$m_3 \vee m_4$; $\neg(\Delta 0 \vee \Delta 1)$	$\Delta\leftarrow$
m_3 ; Δx	m_1
m_4 ; Δx	m_2

$x \in \{0,1\}$

Output:

m_1; $\neg(\Delta 0 \vee \Delta 1)$	И
m_1 ; $\Delta : \Omega x$; Φx	И
All other principal cycles	Л

$x \in \{0,1\}$

Machine $\mathcal{M}$ simulates the operation of $\mathcal{A}_Q$ if its schema is imaged on the machine tape as follows. Each cell of the form $\langle i, i\rangle$ $(1 \leq i \leq n)$ is labeled with the letters q_i and ε. For each i $(1 \leq i \leq n$ there is a j $(1 \leq j \leq i)$ such that the cell $\langle i, j\rangle$ is labeled with the letter α or the letter β. If the cell $\langle i, j\rangle$ is labeled with α, then when the letter $\bar{q}_i$ is received by the input the automaton $\mathcal{A}_Q$ transfers from state a_i to a_j. If the cell $\langle i, j\rangle$ is labeled with β, then when the letter $\bar{q}_i$ is received by the input the automaton transfers from a_i to the same state into which it transferred from a_j with the letter $\bar{q}_i$ at the input. Cells of the form $\langle i, i\rangle$ cannot be labeled with the letter β. In the initial cycle scanner Δ scans the cell $\langle 1, 1\rangle$, scanners M and N scan $\langle 1,2\rangle$, and scanner Ω scans $\langle n, n\rangle$.

The machine $\mathcal{M}$ simulates the operation of $\mathcal{A}_Q$ in the following sense. We say that a certain machine configuration corresponds to state a_i of automaton $\mathcal{A}_Q$ if the machine is in state m_1 and scanner Δ scans the cell $\langle i, i\rangle$. When word P is received at the input of the automaton, the latter falls successively into certain states $a_{s_0}, \ldots, a_{s_m}$. When the same word is received at the input of machine $\mathcal{M}$, the configurations corresponding to states $a_{s_0}, \ldots, a_{s_m}$ begin to occur in succession. The following proposition holds in this case.

LEMMA 2. If after the word $p_1 \ldots p_k$ $(k \leq m)$ is received at the input of an automaton the latter transfers into state a_ℓ and $\ell \leq n$, then the configuration corresponding to state a_ℓ sets in no later than $4(n-\ell)+2$ cycles after input of the word $p_1 \ldots p_k$ to the machine.

The lemma is proved by induction on k.

Theorem 1 and Lemma 2 can be used to prove the following proposition.

THEOREM 3. For any word Q, if the schema of the automaton $\mathcal{A}_Q$ is imaged on the tape of machine $\mathcal{M}$ the machine will generate the letter И at the output if and only if the word received at its input includes the word Q.

3. The automaton $\mathfrak{A}_Q$ can have, in general, several different images on the tape. We now describe a one-input machine $\mathfrak{N}$ that constructs one of the possible images of $\mathfrak{A}_Q$ when the word Q is fed to the one input.

Machine $\mathfrak{N}$ has four states m_1, m_2, m_3, m_4; six scanners $A, B, \Gamma, \Sigma, \Xi, \Omega$; and one input Ψ. The internal and input alphabets coincide with the corresponding alphabets of machine $\mathfrak{M}$. The cycle length is equal to 5. The initial state is m_1. In the initial cycle the entire tape is empty, and the scanners are situated as follows: scanners A, Γ, Ξ in cell $\langle 1, 1\rangle$, scanners Σ and Ω in $\langle 0, 0\rangle$, and scanner B in $\langle 1, 0\rangle$. The abridged transition table for machine $\mathfrak{N}$ is given below.

Operation of scanners Ξ and Ω:

Ψx	$x\Xi\nearrow;\ \Omega\nearrow$

$x \in \{0, 1\}$

Operation of scanners A, B, Γ, and Σ:

$m_1;\ \Gamma x;\ \neg(\Sigma \bar{x})$	$\varepsilon\Gamma\nearrow;\ \Sigma\nearrow;\ A\nearrow;\ \beta B\nearrow$
$m_1;\ \Gamma x;\ \Sigma \bar{x}$	$m_2;\ \varepsilon\Gamma\nearrow;\ \alpha A\rightarrow;\ B\rightarrow$
$m_2;\ \neg(\Sigma\alpha \vee \Sigma\beta)$	$\Sigma\downarrow;\ A\downarrow;\ B\downarrow$
$m_2;\ \Sigma\alpha$	m_3
$m_2;\ \Sigma\beta$	m_4
$m_3 \vee m_4;\ \neg(\Sigma 0 \vee \Sigma 1)$	$\Sigma\leftarrow$
$m_3;\ \Sigma x$	m_1
$m_4;\ \Sigma x$	m_2

$x \in \{0, 1\}$

THEOREM 4. For any word Q machine $\mathfrak{N}$ will construct on its tape an image of automaton $\mathfrak{A}_Q$ in a finite number of cycles after receiving Q at its input; the transitions from states $a_1, \ldots, a_k$ $(k \leq n)$ will be labeled with letters α and β no later than $4 \times (k - l) + 1$ cycles after the word $q_1, \ldots, q_k$ has been received at its input if the automaton transfers from state a_k to state a_l with the input of the letter $\bar{q}_k$.

The theorem is proved by induction on k, with reliance on the fact that the distance between scanners A and Γ, if it changes, can only increase.

4. We now describe the machine $\mathfrak{R}$ that solves Problem B. The machine has sixteen states m_{ij} $(i, j = 1, 2, 3, 4)$; nine scanners $A, B, \Gamma, \Delta, \Sigma, M, N, \Xi, \Omega$; and two inputs Φ and Ψ. The internal, input, and output alphabets coincide with the corresponding alphabets of machine $\mathfrak{M}$. The cycle length is equal to 12. The initial state is m_{11}. In the initial cycle the tape is empty, scanners Δ, M, and N are situated as for machine $\mathfrak{M}$, and scanners $A, B, \Gamma, \Sigma, \Xi$, and Ω the same as for machine $\mathfrak{N}$.

In state $m_{i,j}$ scanners Δ, M, and N operate the same as the corresponding scanners of machine $\mathfrak{M}$ in state m_i, and scanners A, B, Γ, Σ, Ξ, and Ω the same as the corresponding scanners of machine $\mathfrak{N}$ in state m_j. The table of outputs of machine $\mathfrak{R}$ is as follows:

m_1; $\neg(\Delta 0 \vee \Delta 1)$; Φx ; Ψx	И
m_1; $\neg(\Delta 0 \vee \Delta 1)$; Φx ; $\neg(\Psi 0 \vee \Psi 1)$	И
m_1 ; Δ: Ωx; Φx ; $\neg(\Psi 0 \vee \Psi 1)$	И
All other principal cycles	Л

$x \in \{0, 1\}$

THEOREM 5. For any words P and Q machine $\mathfrak{R}$ will generate the letter И at the output after words P and Q have been fed to the respective inputs Φ and Ψ if and only if word Q is included in word P.

The proof of the theorem is analogous to that of Theorem 3 with the use of Theorem 4.

LITERATURE CITED

1. R. I. Freidzon, "A definition of recursive-predicate complexity which does not depend on the standardization of the concept of algorithm," Seminars in Mathematics, Vol. 8, Consultants Bureau, New York (1970), pp. 110-114.
2. D. R. Morrison, "A feasible library automaton," Internat. Congr. Mathematicians, Abstracts of Short Communications (Sec. I), Moscow (1966), p. 10.