SELF-ADAPTIVE AUTOMATA FOR DECODING MESSAGES

1961

SovietRxiv

Abstract

Full Text

CYBERNETICS AND CONTROL THEORY

V. I. LEVENSHTEIN

SELF-ADAPTIVE AUTOMATA FOR DECODING MESSAGES

(Presented by Academician A. I. Berg on 15 VII 1961)

In a preceding note (^12) (whose definitions and notation are used in what follows), simple algorithms were proposed for recognizing whether a coding system has the property of unique decodability ([8,]10), the property of bounded delay ([9,]11), and the synchronization property. Below we investigate the question of the possibility of carrying out the decoding of messages in an automaton with a finite number of states. It is easy to show that a decoding automaton for a given coding system exists if and only if the latter has the property of bounded delay. For coding systems with error correction (for methods of constructing such systems, see ([1-]5)) one can construct a decoding automaton that operates correctly even in the presence of a number of errors (random transitions of some letters of the code into others). However, if one considers disturbances of a more general kind, such as an automaton failure (a random transition to another state), synchronization failure (a shift of the code), etc., then any such failure may lead to incorrect operation of the automaton for all subsequent time. In this connection the concept is introduced of a self-adaptive automaton, which, after some bounded interval of time following the next failure, begins to operate correctly if no new failure has occurred during this time. It is shown in the note that a self-adaptive decoding automaton can be constructed if and only if the coding system has the synchronization property.

1. Let there be given an input alphabet $B = \{b_1, \dots, b_r\}$, an alphabet of states $S = \{S_0, S_1, \dots, S_{N-1}\}$ with a distinguished initial state S_0, and an output alphabet $A = \{a_1, \dots, a_q\}$. By an automaton we shall mean a collection of two discrete functions $f_1(S_i, b_j) = S_{i,j} \in S$ and $f_2(S_i, b_j) = \alpha_{i,j}$, where $\alpha_{i,j}$ is some (possibly empty) word in the alphabet A. An automaton may be regarded as a device which, receiving in state S_i an admissible* input letter b_j, processes it into the word $\alpha_{i,j}$ and passes into the state $S_{i,j}$. If an automaton, receiving in state S_{i_0} an admissible input sequence $b_{j_1} \dots b_{j_n}$, successively processes it into the word $\alpha = \alpha_{i_0,j_1} \dots \alpha_{i_{n-1},j_n}$ and passes into some state S_{i_n}, then we shall write this in the following form: $(S_{i_0}; b_{j_1} \dots b_{j_n} \mid S_{i_n}; \alpha)$.

A completely defined automaton will be called **stable with respect to** a set $\mathfrak{B}$ of infinite sequences in the alphabet B if there is a number t such that, for any sequence $\beta = b_{j_1} b_{j_2} \dots \in \mathfrak{B}$, there exists a state $S(\beta) \in S$ into which the

word $b_{j_1} \dots b_{j_t}$ brings the automaton from any state $S_i \in S$. The smallest of the numbers t possessing the indicated property will be denoted by $t_{\mathfrak{B}}$.

Let an automaton in the initial state S_0 realize a prescribed mapping of the set $\mathfrak{B}$ into the set of all infinite sequences in the alphabet A. If the automaton is stable with respect to the set $\mathfrak{B}'$, where $\mathfrak{B}'$ denotes the set of all sequences obtained from–

---

* Generally speaking, we do not assume that the automaton is completely defined, i.e., that f_1 and f_2 are defined on all pairs (S_i, b_j); in this case the concept of an input sequence admissible in a given state is introduced in the usual way (^6).

throwing away any number k ($k = 0, 1, \dots$) of the first letters of sequences from $\mathfrak{B}$, then we shall call it **self-adjusting**, and the number $t_n = t_{\mathfrak{B}'}$ will be called the **adjustment time**.

Let the self-adjusting automaton process, in the initial state S_0, the sequence

$$b_{j_1} b_{j_2} \dots \in \mathfrak{B}$$

and

$$(S_0;\ b_{j_1} \dots b_{j_t} \mid S'_t;\ a_t), \qquad t = 1, 2, \dots .$$

As a result of a random failure (either in the automaton itself or in the input sequence), the automaton at some instant t_0 may find itself not in the state S'_{t_0}, but in some other state, as a consequence of which it will operate incorrectly for a certain time. However, since the sequence

$$b_{j_{t_0+1}} b_{j_{t_0+2}} \dots \in \mathfrak{B}'$$

and the automaton is stable with respect to $\mathfrak{B}'$, during the time t_n (if no new failures occur) it will "adjust," and at time $t_0 + t_n$ it will be in the state $S'_{t_0+t_n}$, as in the case when there are no failures.

2. Consider a code system [12]

$$\{A = \{a_1, \dots, a_q\}, \quad U = \{u_1, \dots, u_m\}; \quad B = \{b_1, \dots, b_r\}, \quad V = \{v_1, \dots, v_m\}\}$$

($m \geq 2,\ \lambda_{\max} \geq 2$) of the simplest kind, where $A = U$ ($q = m,\ a_i = u_i,\ 1 \leq i \leq m$), and investigate the question of the possibility of constructing an automaton and, in particular, a self-adjusting automaton, for decoding messages. We shall call an automaton **decoding** (for the given code system) if it maps every infinite sequence representable in the form

$$v_{i_1} v_{i_2} \dots$$

into the sequence

$$u_{i_1} u_{i_2} \dots .$$

It can be shown that for every decoding automaton there exists a number t such that the automaton, being in the initial state, processes the first

$$\lambda(v_{i_1} \dots v_{i_k}) + t$$

letters ($k = 1, 2, \dots$) of an arbitrary sequence

$$v_{i_1} \dots v_{i_k} v_{i_{k+1}} \dots$$

into a word whose beginning is the word

$$u_{i_1} \dots u_{i_k}.$$

The least of these numbers t we shall call the **delay** of the decoding automaton and denote it by t_z.

$$\begin{aligned} u_1 = a_1 &\longleftrightarrow v_1 = 10, \\ u_2 = a_2 &\longleftrightarrow v_2 = 101, \\ u_3 = a_3 &\longleftrightarrow v_3 = 001, \\ u_4 = a_4 &\longleftrightarrow v_4 = 0111, \\ t_d = 8, &\quad t_s = 5. \end{aligned}$$

Fig. 1

Fig. 2. Decoding automaton for the code system of Fig. 1 and the self-adjusting automaton obtained by its completion.

$$N = 14 \ (N = 15), \qquad t_z = 6, \qquad t_n = 14$$

Fig. 3. Self-adjusting automaton for the code system of Fig. 1.

$$N = 16, \qquad t_z = 6, \qquad t_n = 9$$

Consider the dictionary (of elementary codes) $V = \{v_1, \dots, v_m\}$ over

$$B = \{b_1, \dots, b_r\}.$$

Denote (cf. [12]) by $H(V)$ (respectively $K(V)$) the set of all proper beginnings (ends) of words of the dictionary V, including the empty

word Λ. By the letters v, γ, δ with various indices we shall denote elements of the sets V, $H(V)$, and $K(V)$, respectively.

Theorem 1. *In order that, for a given coding system, there exist a decoding automaton, it is necessary and sufficient that it possess the bounded-delay property.*

We shall construct the decoding automaton for a coding system possessing the bounded-delay property in a manner analogous to that in which the operators in (7) were constructed. Consider an infinite r-ary oriented tree. Into each vertex of the tree, except the initial one, one edge enters, and r edges leave it in different directions. To each direction one may put into correspondence some letter of the alphabet $B = \{b_1, \dots, b_r\}$, as a result of which to each vertex of the n-th level there will be put into correspondence some word $b_{i_1} \dots b_{i_n}$. We shall carry out the process of construction successively by levels. At the zero stage we assign to the initial vertex of the tree the symbol S_Λ. At the n-th stage we consider all vertices of the n-th level remaining as a result of the preceding construction and apply the following rules:

1°. The word β corresponding to the vertex under consideration is the beginning of a code. Consider all representations $\beta = v_{j,1} \dots v_{j,l(j)}\gamma_j$, $j = 1, \dots, s$. Let $\beta' = v_{i_1} \dots v_{i_k}$ be a word of maximal length such that for any j $(j = 1, \dots, s)$ $l(j) \geq k$ and $v_{j,t} = v_{i_t}$, $1 \leq t \leq k$. Then we assign to the vertex under consideration the symbol $S_{v_{j,k+1} \dots v_{j,l(j)}\gamma_j}$ (the index does not depend on j), and to the edge entering it—the word $u_{i_1} \dots u_{i_k}$; moreover, if $\beta' \neq \Lambda$ $(k > 0)$, then we delete the subtree issuing from this vertex.

2°. Case 1° does not occur. Then we delete the vertex under consideration, the edge entering it, and the subtree issuing from it.

The process described has an end, since for any word β of length T_d that is the beginning of a code, $\beta' \neq \Lambda$. Suppose that in the process of construction the symbols $S_{\beta_0}, S_{\beta_1}, \dots, S_{\beta_{N-1}}$ $(\beta_0 = \Lambda)$ have been assigned to the vertices. Then the resulting graph may be regarded as the "unfolded" state diagram of an automaton with N states $S_{\beta_0}, S_{\beta_1}, \dots, S_{\beta_{N-1}}$, since for each state S_{β_i} and each input letter b_j admissible in the given state, the output word and the next state are indicated. It is evident that this automaton is a decoding automaton for the coding system under consideration, if the state S_Λ is taken as the initial state. The constructed automaton* has minimal delay t_3, which lies within the following bounds:

$$T_d - \lambda_{\max} \leq t_3 \leq T_d - \lambda_{\min}.$$

The necessity of the condition of the theorem is evident.

3. **Theorem 2.** *In order that, for a given coding system, there exist a self-adaptive decoding automaton, it is necessary and sufficient that it possess the synchronizing property.*

The construction of a self-adaptive decoding automaton for a coding system possessing the synchronizing property is carried out analogously to the preceding construction. However, since a completely defined automaton is being constructed, at the n-th stage of the construction, for any word $b_{i_1} \dots b_{i_{n-1}}$, one may assume known a state S_j and a word α such that

$$(S_\Lambda;\ b_{i_1} \dots b_{i_{n-1}} \mid S_j;\ \alpha).$$

Moreover, when an arbitrary word can be assigned to some edge without loss, we shall denote this word by the variable x. To the vertex considered at the n-th stage of construction, to which there corresponds some word $\beta = b_{i_1} \dots b_{i_n}$, the following rules are applied:

1°. The word β belongs to $H(V)$ or β is a proper beginning of some word belonging to $K(V)$. Then we assign to the vertex under consideration the symbol S_β, and to the edge entering it—the word Λ, if β is the beginning of a code, or x otherwise.

* All states of the automaton constructed are distinguishable in the sense of operation (6); nevertheless their number can sometimes be reduced by completing the definition of the automaton on those sequences that are inadmissible in the state S_Λ.

2°. Case 1° does not occur and the word β is the beginning of a codeword. Consider all possible representations $\beta = \delta_i v_{j,1} \dots v_{j,l(j)} \gamma_j$, $j = 1, \dots, s$. It is clear that for some p, $\delta_p = \Lambda$, and let $(S_\Lambda; v_{p,2} \dots v_{p,l(p)} \gamma_p \mid S_{\beta_1}; \alpha)$. If for every j ($j = 1, \dots, s$) there is a code $v_1^{(j)} \dots v_{t(j)}^{(j)}$ such that $\delta_j v_{j,1} \dots v_{j,l(j)} = v_{p,1} v_1^{(j)} \dots v_{t(j)}^{(j)}$, then assign to the vertex under consideration the symbol S_{β_i} entering the edge into it—the word $u_{p,1}\alpha$—and delete the subtree issuing from this vertex. Otherwise we do the same as in case 1°.

3°. Cases 1° and 2° do not occur and the word β is contained in a codeword. Consider all possible representations $\beta = \delta_j v_{j,1} \dots v_{j,l(j)} \gamma_j$, $j = 1, \dots, s$. Let the word δ_p have the smallest length among the words $\delta_1, \dots, \delta_s$, and $(S_\Lambda; v_{p,1} \dots v_{p,l(p)} \gamma_p \mid S_{\beta_1}; \alpha)$. If for every j ($j = 1, \dots, s$) there is a code $v_1^{(j)} \dots v_{t(j)}^{(j)}$ such that $\delta_j v_{j,1} \dots v_{j,l(j)} = \delta_p v_1^{(j)} \dots v_{t(j)}^{(j)}$, then assign to the vertex under consideration the symbol S_{β_1} entering the edge into it—the word x—and delete the subtree issuing from this vertex. Otherwise we do the same as in case 1°.

4°. Cases 1°, 2°, 3° do not occur. Let $(S_\Lambda; b_{i_2} \dots b_{i_n} \mid S_{\beta_1}; \alpha)$. Then assign to the vertex under consideration the symbol S_{β_1} entering the edge into it—the word α—and delete the subtree issuing from this vertex.

It can be shown that the construction process described above will terminate no later than the $(T_s + 2(\lambda_{\max} - 1))$-th stage, and the automaton constructed as a result will be a self-adaptive decoding automaton for the given code system; moreover, for this automaton $t_z \leqslant T_s + 2\lambda_{\max} - 3$, and $t_n \leqslant 2T_s + 5\lambda_{\max} - 7$. The proof of the necessity of the condition of the theorem is based on the following criterion ([12]): a code system does not possess the synchronization property if and only if there exists a relation of the form $\beta v_{i_1} \dots v_{i_k} = v_{j_1} \dots v_{j_l} \beta$, where $0 < \lambda(\beta) < \lambda(v_{j_1})$.

4. There exist other methods of constructing a self-adaptive automaton for a code system possessing the synchronization property. One of them consists in completing the decoding automaton constructed in Sec. 2 according to

the rules 1°–4° of Sec. 3. However, for this method we can assert only that $t_n \leqslant 3T_s + 5\lambda_{\max} - 7$. The construction method used in the proof of Theorem 2 is, generally speaking, associated with some increase in the number of states* and in the delay of the automaton, but guarantees a smaller adjustment time (cf. Figs. 2 and 3).

Remark. It is obvious that the conditions** of Theorems 1 and 2 are also sufficient for all code systems $\{A, U; B, V\}$ with a free dictionary U ([12]). For such code systems it can be shown that the condition of Theorem 1 (2) is necessary when the dictionary U is strongly free (respectively, completely free). Otherwise these conditions, generally speaking, are not necessary.

Received
30 VI 1961

CITED LITERATURE

[1] R. W. Hamming, Bell Syst. Techn. J., **29**, No. 2, 147 (1950).
[2] D. Slepian, Bell. Syst. Techn. J., **35**, No. 1, 201 (1956).
[3] R. C. Bose, D. K. Ray-Chaudhuri, Inf. and Control., **3**, No. 1, 68 (1960).
[4] M. Plotkin, IRE Trans., IT-6, No. 4, 445 (1960).
[5] V. I. Levenshtein, Problems of Cybernetics, issue 5, 123 (1961).
[6] D. D. Aufenkamp, F. E. Hohn, IRE Trans. EC-6, No. 4, 276 (1957).
[7] N. E. Kobrinskii, B. A. Trakhtenbrot, *Logical Investigations*, Acad. Sci. USSR Press, 1959, p. 352.
[8] A. A. Sardinas, G. W. Patterson, Conv. Rec. IRE, IT, part 8, 104 (1953).
[9] E. N. Gilbert, E. F. Moore, Bell Syst. Techn. J., **38**, No. 4, 933 (1959).
[10] Al. A. Markov, DAN, **132**, No. 3, 521 (1960).
[11] Al. A. Markov, DAN, **139**, No. 3 (1961).
[12] V. I. Levenshtein, DAN, **140**, No. 6 (1961).

* The number of states of an automaton constructed by any of the indicated methods can sometimes be reduced by choosing concrete values for x and carrying out equivalent transformations (see Figs. 2 and 3).

** In the case of code systems of the simplest type (see Sec. 2), Theorems 1 and 2 are also valid in the class of automata for which the output $a_{i,j}$ at each moment of time is either a letter of the alphabet A, or the empty word.

Note: Figure translations are in progress. See original paper for figures.