OPTIMAL CODE-REDUNDANCY METHODS FOR ERROR CORRECTION IN FINITE AUTOMATA

UDC 51:621.391

Yu. G. Karpov and M. G. Karpovskii

We propose to analyze the problem of correcting malfunctions in finite automata when errors are absent in the input signals. We state necessary and sufficient conditions for the existence of such automata having a specified memory volume. We show that for the proposed correction methods and for any malfunction multiplicity there are classes of automata for which correction does not increase the memory volume. We give asymptotic estimates for the complexity of automata with the correction of malfunctions synthesized by the proposed methods. We give estimates of the minimum number of absolutely reliable memory elements for the correction of malfunctions of any multiplicity. We describe the algorithms and characteristics for the appropriate programs used to implement the proposed malfunction-correction methods on a computer. We consider three variants of optimum-encoding algorithms using the methods for the solution of extremal combinatorial problems.

1. A method of error correction in finite automata has been proposed in [1], where the "correct" state of the automaton is reconstructed from its "incorrect" state and from the input signal, which is assumed to be uninfluenced by errors. A formal statement of the automaton error-correction problem and necessary and sufficient conditions for the existence of l-error-correcting automata that realize a given automaton transformation and have a specified memory volume are presented in [1]. Throughout the present article (as in [1]) we understand "errors" to mean transient malfunctions only. In the case of catastrophic structural failures the given methods can only be used to detect failures, but not to correct them. This qualification imposes a significant difference between the statements of the error-correction problem for automata in the present study and in [2].

In an automaton $\mathfrak{A}$ with input alphabet $\{x_1, x_2, \ldots, x_{n_x}\}$, set of internal states $\{a_1, a_2, \ldots, a_{n_a}\}$, and transfer function $f(x, a)$ we say that state a_p is x_s-attainable if there is a state a_q such that $f(x_s, a_q) = a_p$. If a realizable transformation $A(x)$ is given by a corresponding graph or table of transitions for the minimal automaton $\mathfrak{A}_{\min}$, the following theorem holds.

THEOREM 1. For the existence of an l-error-correcting automaton realizing the transformation $A(x)$ and having at most m binary memory elements it is necessary and sufficient that there exist an encoding of states of $\mathfrak{A}_{\min}$ by codes of length m such that for any x_s and any two nonequivalent x_s-attainable states a_p and a_q the following condition holds:

$$\|a_p \ominus a_q\| \geqslant 2l + 1, \tag{1}$$

in which $\|a_p \oplus a_q\|$ is the Hamming distance between the code sets corresponding to states a_p and a_q of $\mathfrak{A}_{\min}$.

Theorem 1 states that for the correction of an l-tuple error it is necessary and sufficient to ensure a distance of at least $2l + 1$, not between any two states of $\mathfrak{A}_{\min}$, but only between states that are attainable for one given letter of the input alphabet. This fact enables us in a number of cases to lower significantly the redundancy required for error correction.

We denote by $n_a(x_s)$ the number of x_s-attainable states of $\mathfrak{A}_{\min}$.

Translated from Kibernetika, No. 1, pp. 62-67, January-February, 1971. Original article submitted December 25, 1969.

COROLLARY 1. A necessary condition for the existence of an l-error-correcting automaton realiz-ng the transformation A(x) and having at most m binary memory elements is

$$\max_s n_a(x_s) \leqslant A(m, 2l+1). \tag{2}$$

ere A(m, 2, l + 1) is the maximum number of words of an m-place code with code length 2l + 1.

We denote

$$k_0 =]\log_2 \max_s n_a(x_s)[. \tag{3}$$

ere, as elsewhere,]N[and [N] denote the nearest integer not less than and not greater than N, espectively.

COROLLARY 2. A necessary condition for the existence of an l-error-correcting automaton realiz-ng the transformation A(x) and having at most m binary memory elements is

$$k_0 \leqslant \begin{cases} \min\left(m - \log_2\left(\sum_{i=0}^{l} C_m^i + \right.\right. \\ \left.\left. + \dfrac{C_m^{l+1} - C_{2l+1}^{l}\left[\dfrac{m}{2l+1}\left[\dfrac{m-1}{2l}\left[\cdots\left[\dfrac{m-1}{l+1}\right]\cdots\right]\right]\right]}{\left[\dfrac{m}{2l+1}\right]}\right)\right., \\ m - 4l + \log_2(2l+1)), \quad \text{for} \quad m > 4l+1, \\ \dfrac{4l+2}{4l+2-m}, \quad \text{for} \quad m \leqslant 4l+1. \end{cases} \tag{4}$$

Inequality (4) follows from (2) and the Johnson [3] and Plotkin [4] upper bounds for the existence of n m-place l-error-correcting code.

2. We now give sufficient conditions for the existence of error-correcting automata. The proof of ese conditions is of a constructive nature and generates a corresponding method for the synthesis of ror-correcting automata.

Let λ be a partition of the input alphabet X into nonintersecting subsets $\lambda_1, \lambda_2, \ldots, \lambda_{n_\lambda}$.

We say that an automaton state a_p is λ_s-attainable if there is an $x_r \in \lambda_s$ such that a_p is x_r-attain-ble. The number of λ_s-attainable states is denoted by $n_a(\lambda_s)$.

THEOREM 2. For the existence of an l-error-correcting automaton realizing the transformation (x) and having m memory elements it is sufficient that a partition λ exist for which

$$\begin{cases} \max_s n_a(\lambda_s) \leqslant B(m, 2l+1), & (5) \\ ]\log_2 n_\lambda[\leqslant m -]\log_2 \max_s n_a(\lambda_s)[. & (6) \end{cases}$$

ere B(m, 2l + 1) is the maximum number of words of an l-error-correcting m-place group code.

Proof. We first construct on the basis of the minimum atuomaton $\mathfrak{A}_{\min}$ realizing the transformation (x) a certain equivalent redundant automaton $\mathfrak{A}_\lambda$, as follows. We "split" every state a_p of $\mathfrak{A}_{\min}$ into quivalent states $a_{p_1}, a_{p_2}, \ldots, a_{p_q}$ of $\mathfrak{A}_\lambda$ in such a way that each of the states a_{p_r} (r = 1, 2, ..., q) is ttainable only for one block of the partition λ. We juxtapose in one-to-one fashion the blocks of λ and djacent classes of a certain l-error-correcting m-place group code [see (6)]. For each block λ_s we ncode all l_s-attainable states of $\mathfrak{A}_\lambda$ with elements of the corresponding adjacent class for the selected ode [see (5)]. In this case condition (1) of Theorem 1 is satisfied, and, hence, l-tuple errors can be orrected in $\mathfrak{A}_\lambda$.

Theorem 2 yields a method for the synthesis of l-error-correcting automata; special cases of this method are the replication (standby redundancy) methods and methods in which the set of states of the encoded automaton forms one correcting code [2]. These methods are obtained by setting $\lambda = I$, where I is the partition comprising a single block.

COROLLARY 3. For the existence of l-error-correcting automata realizing the transformation A(x) and having m memory elements it is sufficient that a partition λ exist for which

$$\begin{cases}]\log_2 \max_s n_a(\lambda_s)[\leqslant m - \log_2 \sum_{i=0}^{2l-1} C^i_{m-1}, & (7) \\ ]\log_2 n_\lambda[\leqslant m -]\log_2 \max_s n_a(\lambda_s)[. & (8) \end{cases}$$

We define memory redundancy as the difference between the number of memory elements of the minimum automaton and a redundant automaton with given correcting capacity that realize one particular transformation.

COROLLARY 4. For any number of states and inputs and any multiplicity of correctable errors there is a class of automata for which the correction of errors of given multiplicity does not require memory redundancy.

For example, let the number of states of the minimum automaton be 128, and let the number of input letters be 8. Then for all automata for which $\max_s n_a(\lambda_s) = 16$, memory redundancy is not required for $l = 1$. Here $\lambda_s = \{x_s\}$ $(s = 1, 2, \ldots, n_x)$ [see (7) and (8)].

It follows from Theorem 2 and Corollary 3 that the memory volume m of an error-correcting automaton depends on the choice of partition λ of the set of input letters. The problem of finding the optimum partition λ may be treated as the problem of minimizing the memory volume of the error-correcting automaton. Algorithms and experimental data for the testing of suitable programs for the solution of this problem are given in [5]. The main shortcoming of these algorithms is the acute dependence of the machine time on the number of input letters. In the present article we give some alternative algorithms for the solution of the same problem, along with experimental data on their testing.

Our algorithms are based on the methods of solution of extremal combinatorial problems [6] and make it possible to augment considerably the size of the transition table for the error-correcting automaton while minimizing the memory volume. In [5] algorithms and experimental testing data are also given for suitable programs to find the optimum correcting code for a given automaton and for the encoding of states of the elements of adjacent classes for the resulting code. These programs in combination with programs for the combinatorial synthesis of nonredundant automata form a system for the direct computer synthesis of redundant automata.

3. We next consider the problem of error correction in the case of correlation between errors.

Let a set of m memory elements of an error-correcting automaton be amenable to partition into two subsets comprising m' (m' $\leqslant$ m) and m-m' elements. Errors of multiplicity l' can occur in m' elements, and errors of arbitrary multiplicity can occur in m-m' elements. We denote the set of errors of this kind by $\Gamma_{m', l'}$.

THEOREM 3. For the existence of an automaton correcting the error set $\Gamma_{m', l'}$, realizing the transformation A(x), and having m memory elements it is sufficient that a partition λ exist for which

$$\begin{cases} \max_s n_a(\lambda_s) \leqslant B(m', 2l' + 1), & (9) \\ ]\log_2 \max_s n_a(\lambda_s)[\leqslant m - \log_2 n_\lambda \leqslant m'. & (10) \end{cases}$$

Theorem 2 follows from Theorem 3 for m = m' and $l = l'$.

The automaton correcting the error set $\Gamma_{m', l'}$ is constructed in the form of a series array of two automata, where the first is realized on m' memory elements and corrects l' errors, while the second is realized on m-m' elements and corrects errors of arbitrary multiplicity [7].

It is inferred from a comparison of the sufficient conditions of Theorems 2 and 3 that if the number of memory elements for the error-correcting automaton is greater than the minimum required number m' from conditions (5), (6) or (7), (8) for the correction of l errors, then for

$$]\log_2 \max_s n_a(\lambda_s)[\leqslant m - \log_2 n_\lambda < m'$$

it is always possible to construct an automaton on m memory elements that realizes the same transformation and corrects l + m-m' errors.

COROLLARY 5. For the correction of errors of arbitrary multiplicity in an automaton m' = $]\log_2 \max_s n_a(\lambda_s)[$ absolutely reliable memory elements are necessary and sufficient.

Corollary 5 follows from Theorem 3 for m' = $]\log_2 \max n_a(\lambda_s)[$ and $l' = 0$.

4. We now give estimates of the complexity of error-correcting automata. We introduce the following notation: n_a is the number of states of the minimum automaton $\mathfrak{A}_{\min}$ realizing A(x); $n_a(\lambda)$ is the number of states of the automaton $\mathfrak{A}_\lambda$ obtained by "splitting" of the states of $\mathfrak{A}_{\min}$ (see the proof of Theorem 2); n_x is the number of input letters of $\mathfrak{A}_{\min}$; n_y is the number of output letters of $\mathfrak{A}_{\min}$; $k_\lambda =]\log_2 \max_s n_\lambda(\lambda_s)[$; and $L(\mathfrak{A}^l)$ is the complexity [8] of an automaton correcting l errors and realizing A(x).

THEOREM 4. For $n_a \to \infty$, $n_x \to \infty$, and l = const

$$L(\mathfrak{A}^l) \lesssim \left(\frac{n_a(\lambda)\, n_x k_\lambda}{\log_2 (n_a(\lambda)\, n_x)} + \frac{n_a(\lambda)\,]\log_2 n_y[}{\log_2 n_a(\lambda)} \right) \cdot \varrho, \tag{11}$$

where ϱ is the maximum complexity for two-input logic elements.

The proof of this theorem rests on the asymptotic estimates obtained in [9] for the complexity of underdetermined functions. The only difference inherent in the proposed method for the synthesis of error-correcting automata is the dependence of the upper bound of the complexity on the choice of the partition λ of the set of input letters.

It follows from Theorem 4 that the complexity of an error-correcting automaton is asymptotically independent of the (fixed) multiplicity l of the errors to be corrected.

COROLLARY 6. For $n_a \to \infty$, $n_x \to \infty$, $\log_2 n_x = 0\,(\log_2 n_a)$ and for almost all automata $\mathfrak{A}_{\min}$ with const

$$\frac{L(\mathfrak{A}^l)}{L(\mathfrak{A}_{\min})} \lesssim \frac{n_a(\lambda)}{n_a}. \tag{12}$$

It follows from Corollary 6 that for

$$\lim_{\substack{n_a \to \infty \\ n_x \to \infty}} \frac{n_a(\lambda)}{n_a} = 1, \quad \log_2 n_x = 0\,(\log_2 n_a) \tag{13}$$

the correction of errors of any fixed multiplicity leads asymptotically to an increase in the complexity of the automaton. Condition (13) is satisfied, for example, for $\lambda = 1$. This result is consistent with the analogous result given in [10] for combinatorial networks.

Consequently, if (13) is fulfilled, the proposed error-correction method is asymptotically optimal, and the efficiency of the error-correction method increases with the number of states and the number of input letters for the automaton.

5. We now consider the memory-volume minimization problem, which reduces to the problem of finiding a partition λ that will minimize m of the conditions (5) and (6). To find the optimum λ we apply the following iterative solution procedure. In each step we choose from the set of all possible partitions of the input letters of $\mathfrak{A}_{min}$ a subset Λ of partitions that satisfies one of the inequalities (5) or (6). The problem now reduces to that of finding a partition $\lambda \in \Lambda$ which will minimize the left-hand side of the other inequality. Inasmuch as the number of code words and the number of adjacent classes of a linear group code is a power of two, it then follows from (5) and (6) that the number of steps of the iterative procedure does not exceed $]\log_2 n_a[$ if Λ is chosen from condition (5) and does not exceed $]\log_2 n_x[$ if Λ is chosen from condition (6).

Let us consider the set Λ of partitions of the input alphabet into n_λ blocks. The number of possible partitions $S(n_x, n_\lambda)$ of the set of input letters of power n_x into n_λ blocks is

$$S(n_x, n_\lambda) = \sum_{j=0}^{n_\lambda} \frac{(-1)^j (n_\lambda - j)^{n_x}}{j!\,(n_\lambda - j)!}. \qquad (14)$$

It follows from (14) that the determination of the optimum [in the sense of (5)] partitions λ by direct sequential search through all partitions is possible only for small values of n_x and n_λ, so that the construction of efficient approximative algorithms for solution requires an analysis of the structure of the set Λ (set of partitions containing exactly n_λ blocks).

The set of all possible partitions of a given set forms a lattice M of partitions under partial orderedness [11]. We say that a partition $\lambda^{(i)}$ covers $\lambda^{(j)}$ if $\lambda^{(i)} > \lambda^{(j)}$ and a λ does not exist such that $\lambda^{(i)} > \lambda > \lambda^{(j)}$. We refer to partitions that cover the null-element of the lattice M as points. We enumerate the input letters with the numbers 0, 1, ..., n_x-1 and denote the points of M by pairs (p, q), marking the single block of the partition containing the two elements (input letters) p and q.

We represent the partitions as an amalgamation of points of the lattice M. For a single-valued representation of the partitions we require lexicographic ordering of symbols; specifically, in the amalgamation of points (α_i, β_i) of M

$$\lambda = (\alpha_1, \beta_1) \cup (\alpha_2, \beta_2) \cup \ldots \cup (\alpha_k, \beta_k) \qquad (15)$$

the following relations must be observed: $\alpha_i < \beta_i$, $\alpha_i \neq \alpha_j$, $\beta_i \neq \beta_j$ for $i \neq j$ (i, j = 1, 2, ..., k). The following theorem is a consequence of the Schmidt-Ore theorem [11].

THEOREM 5. It is possible to set in one-to-one correspondence with any partition λ of a set of n_x elements into n_λ blocks a set of n_x-n_λ points that determine this partition by means of relation (15).

We now examine algorithms for the minimization of the functional $\mu = \max_s n_a(\lambda_s)$ determined by means of (5) and (6) on the set Λ of partitions. These algorithms are based on the above-described representation of partitions of the lattice M.

1. Adaptive Random-Search Algorithm

Every point (p, q) of M has associated with it the probability $\rho_{p,q}$ of its selection. We construct a generator of a random set of n_x-n_λ points determining a partition $\lambda \in \Lambda$. The probability that the generator will select the point (p, q) is equal to $\rho_{p,q}$. For every constructed partition $\lambda \in \Lambda$ we compute the value of the functional $\mu = \max_s n_a(\lambda_s)$.

Due to the complex dependence of the functional μ on the set of points entering into the partition we organize the training procedure, i.e., we vary the probabilities $\rho_{p,q}$ in accordance with the results of the preceding selections.

We organize the training procedure cyclically after every n steps (in our program we set n equal to 10 to 100), during which is accumulated the information needed to make a reward-and-punishment decision. The training sequence reduces to a variation of the probabilities $\rho_{p,q}$ of selection of points occurring during one cycle in the composition of "good" or "bad" partitions in more than γ percent of the cases (γ was varied from 60% to 80% in our program). In this case a partition is considered to be "bad" if $\mu \geqslant \bar{\mu}$ and "good" if $\mu < \bar{\mu}$ (where $\bar{\mu}$ is the arithmetic mean value obtained for the functional over all preceding steps).

The program written in commands of the M-200 digital computer for the realization of the algorithm described here occupies 1500_8 cells. The operation of the program requires four memory banks of length $C^1_{n_x}$ cells (including a bank for the point-selection probabilities, two banks for the number of occurrences of points in "good" and "bad" partitions, and an operational bank for the point-selection probabilities). If the working capacity is limited to that of the M-220 internal memory banks, the capacity of the latter does not suffice for the solution of the problem for more than 30 or 40 input letters. The time limitations for the formulated program is of the order of ten or twenty minutes, which is adequate for the memory-volume minimization of an automaton having 30 to 35 states. The algorithm described here therefore differs from those described in [5] in that the solution time is not as dependent on the number of automaton input letters and permits the solution of a problem of larger volume.

2. Local Optimization Method

In the solution of the problem by this method, for every partition $\lambda^{(i)}$ a set of adjacent partitions is determined, where a partition $\lambda^{(i)}$ is considered to be adjacent to $\lambda^{(i)}$ if the amalgamations of points corresponding to these partitions differ by exactly one point. Then in the usual way [6] an adjacency graph (G) is constructed, in which a successive descent is made to the region of a local minimum [6]. The partition globally minimizing the functional μ is sought by scanning of the local minima obtained by the realization of successive descents in the adjacency graph for randomly chosen $\lambda^{(i)}$.

It is essential to note the following characteristics of the realization of this algorithm. First, the set of all partitions adjacent to $\lambda^{(i)}$ contains a subset that can be discarded at once, namely the subset of partitions differing from the given one by permutations of elements between noncritical blocks.

A block λ_s of a partition λ is said to be critical when it maximizes the quantity $n_a(\lambda_s)$ for the given partition λ. Moreover, in the scanning of adjacent partitions we can readily determine the lower bound $\Delta\mu$ as $\max_s n_a(\tilde{\lambda}_s)$, where λ_s is the s-th block of the partition $\tilde{\lambda}$ obtained by decrementing once the number of points in λ. This operation also greatly diminishes the number of partitions that have to be scanned in the adjacency graph.

The program realizing this algorithm on the M-220 computer includes 1000_8 commands and is practically unrestricted in the memory of an intermediate-class digital computer. In a period of the order of 15 min the program is capable of determining the optimum encoding of states in automata for which the product of the number of states by the number of input letters is of the order 10^3. The algorithm is well recommended for implementation on a computer having a small operational memory.

3. Method of Branches and Bounds

The subset Λ of admissible partitions is chosen so as to satisfy inequality (5) for a fixed $B(m, 2l + 1)$. The set Λ consists of all partitions λ of the set of input letters into blocks λ_s such that

$$\max_s n_a(\lambda_s) \leqslant B(m,\ 2l + 1). \tag{16}$$

From this set we pick out the partition having the minimum number of blocks, i.e., we minimize the left-hand side of (6). The problem is solved in two stages.

From the construction of all possible blocks λ_s of λ satisfying (16) the routine is as follows. On the Boolean lattice Z [11] of all subsets of the set of states of the automaton we label n_x subsets A_j of x_j-attainable states of the automaton $\mathfrak{A}_{\min}$. Then on the lattice Z we pick out the set m^B of all subsets A_i^B containing exactly B elements. For each subset $A_i^B \in M^B$ we construct the absorption vector α_i, whose j-th component is equal to unity when $A_j \subseteq A_i^B$, i.e.,

$$a_{ij} = \begin{cases} 1. & \text{if } A_j \subseteq A_i^B; \text{ where } B = B(m, 2l+1)); \\ 0 & \text{in all other cases.} \end{cases}$$

The absorption vectors thus obtained for each A_i^B form the rows of an absorption matrix. Every row of the absorption matrix characterizes those amalgamations of automaton input letters in the block which satisfy (5).

In the second stage of solution of the problem, for the selection of the partition having the minimum number of blocks it suffices to choose the minimum number of rows of the absorption matrix so that at least one unity occurs in each column of the submatrix formed by the selected rows. The optimum partition problem can therefore be reduced to the familiar minimum covering problem. The solution of the latter is carried out by two methods, one approximative and one exact, where the method of branches and bounds is used for the exact solution.

The program used to realize the algorithm just described occupies about 1000_8 cells in the M-220 computer memory. The remaining volume of the operational memory banks is allocated for the rows of the absorption matrix.

Unlike the two preceding algorithms, this algorithm has the drawback that the solution time increases sharply with the number of automaton states. On the other hand, an increase in the number of automaton input letters has an insignificant effect on the solution time. About 12 min are required to solve the minimization problem on the M-220 computer for an automaton having 20 states and 35 to 40 input letters. Consequently, this algorithm is well suited to the memory-volume minimization of automata having a large number of input letters.

LITERATURE CITED

1. M. G. Karpovskii, "Error-detecting-and-correcting finite automata," Computer Engineering and Cybernetic Problems [in Russian], No. 5, Izd. LGU, Leningrad (1968).
2. M. A. Gavrilov, "Structural redundancy and operational reliability of relay devices," Proc. First Internat. IFAC Congr. on Automatic Control [in Russian], Vol. 3, AN SSSR/ Moscow (1960).
3. S. M. Johnson, "A new upper bound for error-correcting codes," IRE Trans. Information Theory, IT-8, 203-207 (1962).
4. M. Plotkin, "Binary codes having a specified minimum distance," Cybernetics Reviews [Russian translation], No. 4, IL, Moscow (1963).
5. M. G. Karpovskii, M. V. Levit, and V. I. Ruzhanskii, "Computer algorithms for the encoding and minimization of error-correcting automata," Computational Methods [in Russian], No. 5, LGU, Leningrad (1968).
6. V. N. Burkov and S. E. Lovetskii, "Methods for the solution of combinatorial-type extremal problems," Avtomat. i Telemekhan., No. 11 (1968).
7. M. G. Karpovskii, V. I. Ruzhanskii, and N. S. Shcherbakov, "Use of correcting codes for the detection and correction of errors in a finite automaton," Proc. Third Conf. Coding Theory and Information Transmission [in Russian] (1967).
8. O. B. Lupanov, "A method of network synthesis," Izv. VUZov, Radiofizika, No. 1 (1958).
9. G. I. Kirienko, "On self-correcting networks made up of functional elements," Problems of Cybernetics [in Russian], No. 12, Moscow (1964).
10. L. A. Sholomov, "Functionals for the complexity of systems of underdetermined functions," Problems of Cybernetics [in Russian], No. 19, Moscow (1967).
11. A. G. Kurosh, Lectures in General Algebra, Pergamon (1965).