Combinatorial Problems over Logical Matrices in Logic Design and Artificial Intelligence

Arkadij Zakrevskij

Abstract - Logic design and artificial intelligence constitute a rich source of hard combinatorial problems which can be formulated in terms of logical matrices. Research into their classification and working out practically efficient solving algorithms were started in early sixties at the Tomsk University and the Siberian Physiko-Tekhnical Institute and developed later in the Institute of Engineering Cybernetics of the Academy of Sciences of Belarus. A brief review of obtained results is presented in this paper.

1. INTRODUCTION

Engineering discrete information systems on the base of modern microelectronics technology is quite impossible without their design automation. In its turn, formalization of the regarded design processes is a necessary step towards their automation. Meanwhile, it is well-known that the design of discrete systems is closely connected with solving various hard combinatorial problems, especially at the level of logical design, where the logical structure of the system is elaborated. The quality of designed systems greatly depends on these problems solutions which demand the lion’s share of time and memory resources when using modern computers. There arises a problem of combinatorial support for the design automation: classifying combinatorial problems and reducing them to a strict mathematical form, developing methods for their solution and synthesising programs for their implementation, creating an intellectual interface which should facilitate their using in engineering practice.

The nature of combinatorial problems of logical design is close to that of many problems of artificial intelligence. Most of them are NP-hard, i.e. their computational complexity (measured by used volumes of time and memory) depends exponentially on the quantity of input data [1]. They say that in principle there is no efficient algorithm for solving such problems. But only theoretical efficiency is supposed by that: it is accepted that an algorithm is efficient if only its computational complexity is restricted by some polynomial of the input data volume when the latter is unlimited. Meanwhile, it is possible to suggest practically efficient algorithms for solving many NP-hard problems. They can be successfully used in the band of real restrictions imposed on input data and/or in case of optimization problems when these algorithms enable to find solutions close to optimal ones.

Naturally, researches in this direction were conducted intensively for many years in different countries, by numerous teams. One of them is the laboratory of logical design of the Institute of Engineering Cybernetics of the Academy of Sciences of Belarus, which continued the research started at Tomsk University and the Siberian Physiko-Tekhnical Institute in early sixties [2-7]. This paper presents a short review of obtained results.

2. MATRIX REPRESENTATIONS

The majority of combinatorial problems of logical design and artificial intelligence can be formulated in terms of logical matrices, Boolean or ternary ones. Boolean matrices, convenient for visual perception, can be used for description of binary relations of various nature, and ternary matrices can be interpreted specifically as compressed forms of Boolean ones [6,8].

Let A and B be some finite sets, and # - some binary relation between them: # ⊆ A × B. And let [A#B] be the Boolean matrix of this relation, its rows corresponding to elements from A, and columns - to elements from B.

For instance, if B = {a,b,c,d,e,f}, and A is a set of three subsets from B, namely B₁ = {a,b,d}, B₂ = {b,e,f} and B₃ = {c,d,e}, then A and B are connected by the relation of belonging ∈ of elements from B to elements from A, which can be conveniently represented by a Boolean matrix [A ∈ B] (below on the left) or a ternary one, using the symbol of uncertainty “-” in case of incomplete information about this relation (below on the right - when it is not known if b belongs to B₁ and d - to B₃).

Logical matrices are suitable for application in dialogue design systems and are easily jointed with program modules. They are included in the number of basic operands of programming language LYaPAS, intended for logical problems and highly efficient in case of Boolean vectors and matrices processing [5,7,9,10].
served as a base for development of efficient means of matrix transformations, which solve numerous problems of logical analysis and synthesis of discrete devices and diagnostics of their failures, as well as many problems of logical recognition of technical and natural objects [11-16].

3. CLASSICAL COMBINATORIAL PROBLEMS

It was the problem of minimization of Boolean functions in the class of normal forms, disjunctive (DNF) or conjunctive (CNF), that served as an initial source of combinatorial problems over Boolean and ternary matrices.

Let \( X \) be a set of \( n \) Boolean variables, taking values from the set \( E=\{0,1\} \), \( M=E^n \) - the Boolean space of these variables, and \( M' \) - some its subset, regarded as characteristic for a Boolean function \( f: E^n \rightarrow E \). Both the set \( M' \) and the function \( f \) can be represented by a Boolean matrix \( [M' \#X] \) of the binary relation \#; an element from \( X \) takes the value 1 on an element from \( M' \). The matrix rows can be interpreted by that either as sets of values of the variables or as corresponding complete conjunctive terms constituting a canonical CNF.

The same information can be given in the more compact interval form. Ternary matrices are used by that, with elements choosing values from the set \( \{0,1,-\} \), and with rows interpreted as conjunctive terms or as intervals of \( M \), representing characteristic sets of these terms. It is considered that a Boolean matrix is equivalent to a ternary one if each of its rows (but only they) can be obtained from some row of the ternary matrix by changing values “-” for 0 or 1 (a ternary row covers a Boolean one in this case). The problem of minimization of a Boolean function in the DNF class is reduced to looking for a ternary matrix \( T \) which is equivalent to a given Boolean matrix \( B \) and has the minimum number of rows.

For example, the following matrix pair illustrates a transition from the canonical CNF \( f = ab'c'd' \vee abcd' \vee ab'c'd' \vee abc'd \vee abc'd \vee a'bc'd \) to the more compact CNF \( f = ad'\vee ab \vee bc'd \).

The classical method for minimization of DNF consists (with accuracy to terms) in constructing the set \( P \) of all maximal on \( M' \) intervals of the space \( M \) and selecting from \( P \) a minimum number of intervals containing together all elements of \( M' \) - or, what is the same, finding a shortest row cover for the Quine matrix \( [P \#M'] \), i.e. a minimum combination of rows containing together at least one 1 in each column [19].

The second of these problems (the covering problem) turned out to be very fruitful, having useful interpretations in different applications where not only shortest covers are regarded but irredundant ones as well (not covered by others). The following example illustrates an initial Boolean matrix \( A \) and a “reverse” matrix \( B \). Columns of the latter one point out all irredundant covers of matrix \( A \), the shortest covers in their number, one of which marked by 1s in the first column of \( B \) is represented also by matrix \( C \).

\[
A = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\
\end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
\end{bmatrix}
\]

Note that computing the “reverse” matrix is equivalent to conversion the set of “maximal 1s” of a monotone Boolean function (upper bound of the set \( M' \)) into the set of “minimum 0s” (lower bound of \( M\cap M' \)). This operation is used when checking conditions for implementation of Boolean functions by threshold elements [20,21].

A lot of publications were dedicated to the covering problem. There were proposed various algorithms for its solution, including trivial “greedy” algorithms and more sophisticated, exact and approximate [22-26].

This problem belongs to a rich class of problems of finding a Boolean matrix minor formed by some rows or columns and possessing some given quality by that. The problem of finding a minimum unconditional diagnostic test is another representative of this class. It is formulated as follows: a Boolean matrix \( B=[D \#S] \) is given which represents the binary relation \# between a finite set of failures \( D \) and a set of observable attributes \( S \) (\( d \#s \), if failure \( d \) has attribute \( f \)), and it is needed to choose from \( S \) a minimum number of attributes sufficient for recognition of any failure. Solving this problem is reduced to finding in \( B \) some minimum column minor with different rows (it is supposed that all rows of \( B \) are different). Any obtained result can be represented by a Boolean vector \( b \) with 1s marking the chosen columns.

For instance,
A number of logic circuits design problems are reduced to solving matrix logical equations $U = B \times C$. Here $U$, $B$, and $C$ are Boolean matrices, and conjunction ($\land$), on the one hand, and disjunction ($\lor$) or exclusive disjunction ($\oplus$), on the other hand, serve correspondingly as inside and outside operators when multiplying matrices. Some of these matrices are given by that, and other ones should be found. Regarding more complicated circuits entails solving systems of such equations. This problem is decomposed into a series well-formulated and more easy problems over separate matrices (50 such problems were described in [36]). For instance, the following ones are belonging to their number, where some Boolean matrix $B$ is regarded:

1) the problem about a minimum disjunctive basis [37]: to find for $B$ such matrix $C$ with minimum number of rows that any row of $B$ equals the componentwise disjunction of some rows of $C$;
2) the problem about a minimum linear basis: the problem is the same as the previous one, but exclusive disjunction is regarded instead of disjunction;
3) the problem about a minimum disjunctive code [38]: to find for $B$ all rows of which are different such matrix $C$ with minimum number of columns that all its rows are also different and each its column equals the componentwise disjunction of some columns of $B$;
4) the problem about a minimum packing: to decompose $B$ into a minimum number of row minors each of which contains in every column not more than one unit;
5) the problem about a compact packing: to find in $B$ such a row minor which contains in each column exactly one unit.

The following examples illustrate some of these problems:

\[
\begin{array}{cccccc}
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
a & b & c & d & e \\
0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
C = 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 \\
\end{array}
\]

Some exact and approximate algorithms for solving this problem are given in [11,14,15].

Thousands of publications were dedicated to the problem of Boolean functions minimization, [27-29] in their number, were functions of many arguments were considered as well as weakly specified Boolean functions. Such a function is defined by a pair of sets $M'$ and $M''$, composed of relative small number of inputs where the function takes accordingly values 1 and 0. Minimization of weakly specified Boolean functions is reduced to finding a minimum set of intervals covering together $M'$ but not intersecting with $M''$ by that.

4. MATRIX LOGICAL EQUATIONS

Development of modern VLSI technology stimulated researches in optimal silicon implementation of logic circuits. The matrix structure of circuits is adequately reflected by mathematical matrix tools suggested for logic analysis and synthesis. These tools solve the problems of mutual minimization of systems of Boolean functions using as criteria the whole number of conjunctive terms, and also the number of variables and literals - in case of partial functions [13,30-35].

In accordance with these methods, any system of Boolean functions is represented by a pair of matrices $B$ and $U$. The matrix $B$ serves for the enumeration of elements or intervals of the Boolean space $M$, constituting the area of definition for the functions, and the matrix $U$ shows the corresponding values of the functions represented by matrix columns.

Some problems which seem quite different at the interpretation level turn out to be very close under more abstract formulation. For instance, such are the problems of diagnostics, minimization of the number of arguments and decomposition of a system of Boolean functions. They are reduced combinatorially to selecting from a Boolean matrix $B$ a minimum column minor, which satisfies one of the following conditions:

a) rows corresponding to different rows of the matrix $U$ should be different (the condition of inside decomposition).

b) rows corresponding to equal rows of the matrix $U$ should be different (the condition of outside decomposition).

For example, the result of solving this problem for concrete matrices $B$ and $U$ under the condition $a$ is presented by the matrix $C$, and under the condition $b$ - by the matrix $D$.

\[
B = \begin{bmatrix}
1 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

\[
U = \begin{bmatrix}
1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 \\
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

\[
D = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
\end{bmatrix}
\]
Each of the given problems can be regarded at different aspects.

For instance, looking for a minimum disjunctive basis of a Boolean matrix \( B \) is practically equivalent to its decomposition into a product of two minimum matrices or to finding its shortest covering by unit minors (sets of unit elements situated on intersections of some rows with some columns). That can be seen from the following example:

\[
\begin{array}{ccccccc}
1 & 1 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 \\
& & & & & & \\
\end{array}
\times
\begin{array}{c}
1 \\
1 \\
0 \\
1 \\
1 \\
0 \\
\end{array}
= \begin{array}{c}
1 \\
1 \\
0 \\
1 \\
1 \\
0 \\
\end{array}
\]

Note that a minimum disjunctive basis for rows of the matrix \( B \) is represented in this example by the second multiplier, and the first one can be easily obtained from it being conjugated with it and representing at the same time a minimum disjunctive basis for the set of columns of the matrix \( B \).

An important class of problems of digital devices design is formulated in terms of linear logical equations, or logical difference equations [39].

Every Boolean function can be represented as a polynomial - the mod-2-sum of some elementary conjunctions. If all conjunctions are positive, it is a Zhegalkin polynomial; if each variable can be presented as a conjunction. If all conjunctions are positive, it is a polynomial - the mod-2-sum of some elementary logical difference equations [39].

Polynomials can be used as structure formulae of two-level AND/EXOR-circuits. Such circuits possess some advantages over conventional AND/OR-circuits: they are more testable and more compact in case of implementation of symmetrical Boolean functions typical for arithmetic. Their optimal synthesis can be reduced to looking for Zhegalkin or Reed-Muller polynomials realizing given Boolean functions and having by that the minimum number of terms. This problem becomes essentially more complicated in case of a partial Boolean function specified only on some \( k \) inputs presented by the rows of the matrix \( B \).

A method has been proposed by the author, reducing that problem to solving a system of \( k \) linear logical equations. Such system is represented by the expression \( Rx = u \), where \( R \) is a staircase Boolean matrix obtained by conjunctive closing of the set of columns of the matrix \( B \), and \( u \) - the vector of a regarded function values on the definition area. The vector \( x \) is to be found, which should satisfy this equation. The well-known Gaussian method of variables exclusion [41] enables to get rather easily some solution or even the whole set of solutions (if they exist), but it is insufficient for our aim - finding one shortest solution presented by vector \( x \) with minimum number of units. An exact solution of this problem can be found using a modified tree searching technique [42,44,45], an approximate one - directly from the matrix \( B \), without obtaining \( R \) and essentially faster [43].

The way of representation of partial Boolean function \( u(a,b,c,d,e) \) by a matrix \( B \), the conjunctive closure of the latter - by a matrix \( R \), and the obtained minimum implementing Zhegalkin polynomial \( u = c \oplus ab \) - by a vector \( x \) (given in the transposed form) is illustrated by the following example:

\[
\begin{array}{cccccc}
a & b & c & d & e & u \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 \\
\end{array}
\quad
\begin{array}{cccccc}
a & d & ad & b & c & bc & ab & ac & cd & ae \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
\end{array}
\]

There was solved also a more hard problem of the synthesis of AND/EXOR-circuits with minimum conjunctive terms and implementing systems of \( m \) Boolean functions of \( n \) variables specified on \( k \) inputs. The suggested method [46-49] is based on the theory of linear vector spaces and is especially efficient in case of weakly defined systems, with small \( k \) and without essential restriction on \( n \) and \( m \) which can reach several hundreds - for an approximating algorithm. This method was extended onto many-valued functions in [50].

Some impression about this problem can be received from the following example where the matrix \( B \) gives the whole area of definition, \( F \) presents the values of regarded functions on this area, rows of \( K \) show selected positive conjunctive terms, and \( Z \) demonstrates their distribution between mod-2-sums which implement the regarded functions: \( u = b \oplus c \oplus d, v = b \oplus d \oplus be, w = b \oplus c \oplus ac \).

\[
\begin{array}{cccc}
a & b & c & d & e \\
0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 \\
\end{array}
\quad
\begin{array}{cccc}
u & v & w \\
1 & 1 & 0 \\
1 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]
5. SOME PROBLEMS OF THE AUTOMATA THEORY

When a binary relation represented by a Boolean matrix \( B \) is defined on one set and is symmetrical, it can be interpreted as the relation of incompatibility on the set of partial states of a parallel automaton [51], and solving this problem - as looking for a minimum interval displacing code for these states, what is necessary for the hardware implementation of parallel algorithms for logical control [52-53]. This problem can be regarded also as the problem of covering the graph of incompatibility between partial states by a minimum number of complete bipartite subgraphs which can be represented by ternary vectors with 1s marking components of one part and 0s - marking the other part.

This can be illustrated by the following example of an incompatibility graph. A minimum solution of the regarded problem for this graph includes three complete bipartite subgraphs represented by the corresponding columns of a ternary matrix shown on the right.

\[
\begin{align*}
B &= \begin{bmatrix} 1 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & = F \\
1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 1
\end{bmatrix} \\
K &= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 = Z \\
1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1
\end{bmatrix}
\end{align*}
\]

Practically efficient algorithms for solving this problem were proposed in [54-56].

Rather rich in the sense of possible interpretations is also the problem of compact packing which specifically was applied for checking logical control algorithms for correctness [57]. For example, let a Boolean matrix \( B \) to present the relation of belonging of places of some net of free choice [58] to reachable markings, and the columns of a matrix \( C \) - to show all possible compact packages for \( B \). Then (as follows from the Hack theorem [58]) every row of \( C \) will have at least one unit if the net is live and safe.

6. LOGICAL INFERENCE IN RECOGNITION PROBLEMS

It is worth-while to note that many problems of pattern recognition are also reduced to solving combinatorial tasks. In [59], a logical approach has been suggested to their solving in the Boolean space of attributes, including inductive inference (obtaining knowledge from data) and deductive inference (using knowledge for computation of some goal attributes values). Data present information concerning individual objects of an experimental selection from a regarded subject area, and knowledge - information about this area as a whole, defining relations between attributes inherent in it. Knowledge is presented by a set of implicative regularities, which can be regarded as a generalization of the functional ones. Both data and knowledge are represented by binary and ternary vectors and matrices, and methods of Boolean functions theory are effectively used for their processing. For instance, many problems of equality transformations and logical inference are reduced to checking CNF for satisfiability.

The methods of the theory of Boolean functions were extended later onto finite predicates [60,61], sectional Boolean vectors and matrices were proposed for representation of data and knowledge, and the suggested approach was generalized for recognition in the space of many-valued attributes [62-64].

The following example illustrates the recognition via deductive inference. Suppose the objects of a regarded area are described in terms of the attributes \( a, b, c \) taking values accordingly from the sets \( \{1,2,3\}, \{1,2,3,4\} \) and \( \{1,2\} \), and knowledge is represented by a sectional Boolean matrix \( D \) with rows demonstrating separate regularities-disjuncts (given on the right in the algebraic form).

\[
a \begin{bmatrix} 1 & 2 & 3 & 4 \\
\end{bmatrix}
\]

\[
D = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix} \begin{bmatrix} (a=3) \lor (b=3) \\
(b=3) \lor (b=4) \lor (c=2) \\
(a=2) \lor (b=1) \lor (b=2) \lor (c=1) \\
(a=3) \lor (c=2) \\
\end{bmatrix}
\]

Any object of the regarded area has some definite value for every attribute and is represented by a vector with one 1 in each of three sections. It should satisfy all regularities and that means that three columns of \( D \) corresponding to the given values of attributes should cover all rows - have together at least one 1 in each of them. For example, the object presented by the vector 010.001.01 satisfies this condition, and the object presented by 001.001.01 - does not satisfy.

Suppose now that some object of the given area possesses the value 1 of the attribute \( c \). Using this information we can reduce the knowledge matrix \( D \), deleting the third row (the corresponding regularity is satisfied) and the last section (corresponding to the attribute \( c \) with fixed value). We obtain by that:

\[
a \begin{bmatrix} 1 & 2 & 3 & 4 \\
\end{bmatrix}
\]

\[
D' = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix} \begin{bmatrix} (a=3) \lor (b=3) \\
(b=3) \lor (b=4) \lor (c=2) \\
(a=2) \lor (b=1) \lor (b=2) \lor (c=1) \\
(a=3) \lor (c=2) \\
\end{bmatrix}
\]
0 0 1 . 0 0 0 0 0

It is easy to conclude that the analysed object has the value 3 of the attribute \( a \) and the value 3 or 4 of the attribute \( b \).

A knowledge matrix can be defined either by an expert or be obtained automatically from data - by the algorithm of inductive inference [65]. When data are strongly restricted, it is reasonable to use the syllogistic approach - in this case regularities tie attributes in pairs, and computations are fulfilled much faster than in the general case of arbitrary regularities [66-68]. It is necessary sometimes to take into account the effects of asymmetry - for instance, they could result in impossibility of transforming the regularity “if \( a=1 \), then \( b \neq 3 \)” into “if \( b=3 \), then \( a \neq 1 \)’’ [69].

Some expert logical recognition systems were made on the base of the suggested methods [70-72].

7. METHODS OF COMBINATORIAL SEARCH

Solving combinatorial problems over logical matrices is unavoidably connected with exhaustive search, but the latter can be sometimes greatly reduced when taking into account peculiarities of concrete problems.

The most general approach to solving combinatorial problems is based on scanning a search tree that is growing while we look for a solution [73-75]. This process has a recursive character: on each step a considered problem is changed for several alike ones but over reduced data. The descendant problems are regarded in some order and reduced if possible, and less worthwhile of them can be deleted. The processes of decomposing of arising situations are intertwined by that with reducing of them, and it is natural to strive for decomposition into minimum number of more simple situations. In the most effective way this striving can be satisfied only by a profound analysis of concrete problems and developing corresponding formal theories. Some results in this direction are reflected in [76-80].

8. CONCLUSION

Logical matrices can serve as universal means for formulating various problems of logic design and artificial intelligence (and not only for these problems). They are well co-ordinated with features of modern discrete mathematics and peculiarities of computers. They can provide a good base for combinatorial support of processes of logic design and artificial intelligence, enabling to raise essentially the efficiency of solving numerous problems from these areas.

In this article, only some of these problems were described, and next to nothing was said about methods for their solution - because of lack of space. One can find the appropriate information using the list of publications given below.

REFERENCES


[77] Zakrevskij A.D., Bibilo P.N., Dudkin A.A. A program package for the synthesis of combinational circuits in the basis of PLAs. - Upravlyayushchie systemy i mashiny, 1985, No 1, pp.27-29 (in Russian).

