INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2012, VOL. 58, NO. 1, PP. 71–76
Manuscript received December 31, 2011; revised March 2012. DOI: 10.2478/v10177-012-0010-x

Features Reduction Using Logic Minimization Techniques

Grzegorz Borowik, Tadeusz Łuba, and Dawid Zydek

***Abstract*—This paper is dedicated to two seemingly different problems. The first one concerns information theory and the second one is connected to logic synthesis methods. The reason why these issues are considered together is the important task of the efficient representation of data in information systems and as well as in logic systems. An efficient algorithm to solve the task of attributes/arguments reduction is presented.**

***Keywords*—Machine learning, knowledge representation, discernibility function, logic minimization, attribute reduction, complement.**

I. Introduction

IN the areas of machine learning, artificial intelligence, as well as logic synthesis we often deal with some functional dependencies (for example, in the form of decision tables) in which not all attributes are necessary, i.e. some of them could be removed without loss of any information.

This problem has been investigated from a number of points of view. One of them is whether the whole set of attributes is always necessary to define a given partition of the universe and the other concerns the simplification of decision tables, namely the reduction of condition attributes in a decision table. The reduced set of attributes is then used to compute the set of decision rules. Removing unnecessary attributes can save memory space or computational time and have other benefits over working on non-reduced information systems. However, especially with big tables, the problem of finding which attributes can be removed is nontrivial [1]–[15].

A similar problem arises in logic synthesis where circuits performance can be presented as truth tables. These tables are in fact decision tables with two valued attributes, where condition attributes for decision table are input variables for truth table and similarly decision attributes for decision table are output variables for truth table of the circuit. In the practical application of Boolean algebra the key problem is to represent Boolean functions by formulas which are as simple as possible. One approach to this simplification is to minimize the number of variables appearing in truth table. Then the reduced set of input variables is used in other optimization algorithms, e.g. logic minimization and logic decomposition. Argument reduction combined with other design techniques allows the designer to reduce the size of implemented circuits [16]–[19].

Grzegorz Borowik and Tadeusz Łuba are with the Institute of Telecommunications, Warsaw University of Technology, Warsaw, Poland (e-mail: gborowik@tele.pw.edu.pl; luba@tele.pw.edu.pl).

Dawid Zydek is with the Department of Electrical Engineering, Idaho State University, USA (e-mail: zydedawi@isu.edu).

Interestingly, these two independently researched issues, have resulted in a series of computational methods, algorithms and their computer implementations. These implementations possess so many similarities that it is worthwhile investigating and applying their common realizations [20].

The paper begins with an overview of basic notions of information systems, functional dependencies, decision tables and reducts. In section II we discuss relations between multiple-valued logic and decision table systems with respect to classification of data with missing values. It is particularly shown that elimination of attributes can be easily obtained using standard procedures used in logic synthesis. New contribution is presented in section III, where we describe how to apply complementation algorithm and provide the new variant of the attribute reduction process. Finally, experimental results are presented.

II. Preliminary Notions

The information system contains data about objects characterized by certain attributes. More formally, an information system is a pair $\mathbb{S} = (U, A)$, where U is a nonempty set of objects (in logic synthesis they are usually called minterms) called the universe, and A is a nonempty set of attributes (variables). If we distinguish in an information system two disjoint sets of attributes $A = C \cup D$, called respectively condition and decision attributes (input and output variables), then the system will be called a decision table (in logic synthesis it is called a truth table). The decision table describes conditions that must be satisfied in order to carry out the decisions specified for them. Any decision table defines a function f that maps the direct product of U and A into the set of all values.

The values of the attribute c, i.e. $e_1 = f(u_1, a_i)$ and $e_2 = f(u_2, a_i)$ are said to be *compatible* ($e_1 \sim e_2$), if and only if $e_1 = e_2$ or $e_1 = -$ or $e_2 = -$, where '$-$' represents the case when attribute value is unknown.

On the other hand, if e_1 and e_2 are defined and are 'different' it is said that e_1 is *not compatible* with e_2 and is denoted as $e_1 \nsim e_2$. The consequence of this definition is a COM relation defined as follows:

Let $B \subseteq C$ and $p, q \in U$. The objects $p, q \in \text{COM}(B)$ if and only if $f(p, c) \sim f(q, c)$ for every $c \subseteq B$.

The objects p and q, which belong to the relation COM(B), are said to be *compatible in the set* B. Compatible objects in the set $B = C$ are simply called *compatible*.

The compatibility relation of objects is a tolerance relation (reflexive and symmetric) and hence it generates compatible classes on the set of objects U.

Compatibility relation allows us to classify objects but the classification classes do not form partitions on the set U, as it is in the case of indiscernibility relation (IND) [13]. COM(B) classifies objects grouping them in compatibility classes, i.e. $U/\text{COM}(C')$, where $C' \subseteq C$.

For the sake of simplicity, collection of subsets $U/\text{COM}(C')$ will be called *r-partition* on U and will be denoted as COM(C').

R-partition can be used as a tool to classify objects of a data table description. It can be shown that the r-partition concept is a generalization of the ideas of partition algebra [21]. Thus, all the symbols and operations of partition algebra are applyied to r-partitions.

The justification of generalization of the r-partition concept is demonstrated in the following example:

Example 1. For the function expressed in the form of Table I, the symbol '$-$' in an object may assume the value 0 or 1. It results in an object representing multiple rows. Hence, the classification by indiscernibility relation is no more valid. This is clearly explained by the fact that the object u_1 for the attributes c_1, c_3, c_4, i.e. $(0--)$ is not the same as the objects u_2 (001) and u_5 (01$-$). The object u_1 represents in fact a set of objects 000, 001, 010, 011.

TABLE I
SAMPLE DECISION TABLE

	c_1	c_2	c_3	c_4	d
u_1	0	0	–	–	0
u_2	0	1	0	1	0
u_3	1	0	0	1	1
u_4	1	1	0	1	–
u_5	0	1	1	–	1
u_6	1	0	1	0	–
u_7	1	1	1	0	0
u_8	1	0	1	1	1
u_9	1	1	1	1	1

It is also not precise to say that object u_1 is compatible with either object u_2 or object u_5. According to definition of COM relation, objects of the set $U = \{u_1, \ldots, u_9\}$ for the attributes c_1, c_3, c_4 can be classified as: $1,2;\ 1,5;\ 3,4;\ 6,7;\ 8,9$. This is evidently not correct because the object $0--$ contains the object 000 as well, and this is different from both u_2 and u_5. Therefore, the object u_1 also has to belong to an another class. These classes ought to be as follows:

$$1;\ 1,2;\ 1,5;\ 3,4;\ 6,7;\ 8,9.$$

Any set of objects specifying a data function can be divided into various classes for different attributes. Such a family of classes (i.e. r-partition) is denoted by $\Pi(B)$, where B is a selected subset of the set $C = \{c_1, \ldots, c_m\}$. The r-partition of a single element set $B = \{c_i\}$ is denoted as $\Pi(c_i)$ or simply Π_i.

The r-partition generated by a set B is nothing but the product of r-partitions generated by the attributes $c_i \in B$:

$$\Pi(B) = \bigcap_{i:c_i \in B} \Pi(c_i).$$

If $B = \{c_{i_1}, \ldots, c_{i_k}\}$, the product can be expressed as: $\Pi(B) = \Pi_{i_1} \cdot \ldots \cdot \Pi_{i_k}$.

Example 2. For the Table 1

$$\begin{aligned}
\Pi_1 &= \{1,2,5;\ 3,4,6,7,8,9\},\\
\Pi_2 &= \{1,3,6,8;\ 2,4,5,7,9\},\\
\Pi_3 &= \{1,2,3,4;\ 1,5,6,7,8,9\},\\
\Pi_4 &= \{1,5,6,7;\ 1,2,3,4,5,8,9\}.
\end{aligned}$$

Therefore, for the set $B = \{c_1, c_3, c_4\}$,

$$\Pi(B) = \Pi_1 \cdot \Pi_3 \cdot \Pi_4 = \{1;\ 1,2;\ 1,5;\ 3,4;\ 6,7;\ 8,9\}.$$

The above result of $\Pi(B)$ can also be calculated directly from the definition of COM relation.

For the first row of the Table I, i.e. $u = (00--)$ and for the set $B = \{c_1, c_3, c_4\}$, the objects belonging to $\Pi(B)$ are calculated as a set difference

$$U' = (0--) \backslash \{l_1, l_2\},$$

where l_1 is the product of the objects u_1 and u_2, l_2 is the product of u_1 and u_5, i.e.

$$l_1 = (0--) \cdot (001) = 001,$$

$$l_2 = (0--) \cdot (01-) = 01- .$$

Therefore $U' = \{(000)\}$, and a single element set $\{1\}$ has to be added to the COM(B), i.e.

$$\Pi(B) = \text{COM}(B) \cup \{1\}.$$

III. Elimination of Input Variables

In this section the process of detection and elimination of redundant attributes using concepts of logic systems is described. However, appropriate simplifications caused by functional dependency features as well as the generalization to the case of rough partition will be efficiently applied.

An argument $x \in X$ is called *dispensable* in a logic specification of function F iff $P(X - \{x\}) \leq P_F$, otherwise, i.e. $P(X - \{x\}) \nleq P_F$, an argument is called *indispensable* (i.e. an essential variable).

The meaning of an indispensable variable is similar to that of a core attribute, i.e. these are the most important variables. In other words, no indispensable variable can be removed without destroying the consistency of the function specification. Thus, the set of all indispensable arguments will be called a *core* of X and will be denoted as CORE(X).

We have to eliminate an input variable in order to find the core and then to verify whether the corresponding partition inequality holds. To make this procedure more efficient a key theorem is formulated (we reformulate this problem to apply more useful tools which are efficiently used in switching theory [16], [22]):

A set $B = \{b_1, \ldots, b_k\} \subseteq X$ is called a *minimal dependences set*, i.e. *reduct*, of a Boolean function F iff $P(B) \leq P_F$, and there is no proper subset B' of B, such that $P(B') \leq P_F$.

It is evident that an indispensable input variable of function F is an argument of every minimal dependence set of F.

Now we introduce two basic notions, namely discernibility set and discernibility function, which will help us to construct an efficient algorithm for attribute reduction process.

Let $\mathbb{S} = (U, C \cup D)$ be an information system. Let p, q ($p \neq q$) are objects of U, such that $d \in D$ is a decision attribute and $f(p,d) \nsim f(q,d)$. By C_{pq}, we denote a set of attributes called a *discernibility set* which is defined as follows:

$$C_{pq} = \{c_i \in C : f(p, c_i) \nsim f(q, c_i)\}. \tag{1}$$

A *discernibility function* $f_{\mathbb{S}}$ for an information system $\mathbb{S}$ is a boolean function of m attributes $c_1, \ldots, c_m$, defined by the conjunction of all expressions $\vee(C_{pq})$, where $\vee(C_{pq})$ is the disjunction of all attributes $c \in C_{pq}$, $1 \leq p < q \leq n$.

A strong connection between the notions of a reduct $\mathrm{RED}(\mathbb{S})$ in an information system $\mathbb{S}$ and prime implicant of the monotonic boolean function $f_{\mathbb{S}}$ was investigated among others by Skowron, Kryszkiewicz and Słowiński [9], [13]:

$$\begin{aligned}&\{c_{i_1}, \ldots, c_{i_k}\} \in \mathrm{RED}(\mathbb{S}) \\ &\text{iff } c_{i_1} \wedge \ldots \wedge c_{i_k} \text{ is a prime implicant of } f_{\mathbb{S}}.\end{aligned} \tag{2}$$

Note that minimization of the discernibility function is equivalent to transforming it from the conjunctive normal form (CNF) (in which it is originally constructed) to the disjunctive normal form (DNF) and finding a minimum implicant. Such a transformation is usually time-consuming and therefore it is important to find more efficient algorithms.

Thus, we will describe the collection $\{C_{pq}\}$, of all C_{pq} sets in the form of the binary matrix M for which an element b_{ij} ($i = 1, \ldots, t = \mathrm{CARD}(\{C_{pq}\})$, $j = 1, \ldots, m = \mathrm{CARD}(C)$) is defined as follows:

$$b_{ij} = \begin{cases} 1, & \text{if } c_j \in C_{pq_i}, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Thus, the M matrix is a $0-1$ matrix determined by C_{pq} sets. Our goal is to select an optimal set L of arguments corresponding to columns of M. Here a 'column covering' L means that every row of M contains a '1' in some column which appears in L. More precisely, a column cover of binary matrix is defined as a set L of columns such that for every i

$$\sum_{j \in M} b_{ij} \geq 1. \tag{4}$$

Covers L of M are in one-to-one correspondence with the reduced subsets of arguments, i.e. reducts.

An interesting approach is based on the fact that the unate complementation is intimately related to the concept of a column cover of the binary matrix [23].

Theorem 1 [23]. Each row i of $\overline{M}$, the binary matrix complement of M, corresponds to a column cover L of M, where $j \in M$ if and only if $\overline{M}_{ij} = 1$.

The rows of $\overline{M}$ include the set of all minimal column covers of M. If $\overline{M}$ was minimal with respect to containment, then $\overline{M}$ would precisely represent the set of all minimal column covers of M.

Let each column of M corresponds to conjunction factor of F_M, which is defined by the disjunction of all M_i where M_i is the conjunction of negative literals $\overline{x}_j$ corresponding to $b_{ij} = 1$.

To obtain discernibility function in the minimal DNF we apply the fast complementation algorithm of unate Boolean functions adopted from ESPRESSO [23].

The fast complementation algorithm for monotonously decreasing function F_M is based on the Shannon expansion of F_M, for simplicity denoted by F:

$$F = x_j F_{x_j} + \overline{x}_j F_{\overline{x}_j} \tag{5}$$

where $F_{x_j}, F_{\overline{x}_j}$ are cofactors of F with respect to splitting variable x_j, i.e. the results of substituting '1' and '0' for x_j in F.

Thus by complementing (5)

$$\begin{aligned}&\overline{x_j F_{x_j} + \overline{x}_j F_{\overline{x}_j}} = (\overline{x}_j + \overline{F}_{x_j})(x_j + \overline{F}_{\overline{x}_j}) = \\ &= x_j \overline{F}_{x_j} + \overline{x}_j \overline{F}_{\overline{x}_j} + \overline{F}_{x_j} \overline{F}_{\overline{x}_j}(x_j + \overline{x}_j) = x_j \overline{F}_{x_j} + \overline{x}_j \overline{F}_{\overline{x}_j}\end{aligned}$$

we obtain a formula:

$$\overline{F} = x_j \overline{F}_{x_j} + \overline{x}_j \overline{F}_{\overline{x}_j} \tag{6}$$

which is the key to a fast recursive complementation process.

Hence applying the property of unateness the equation (6) can be expressed as simplified formulas, i.e.:
when $F_{x_j} \leq F_{\overline{x}_j}$

$$F = F_{x_j} + \overline{x}_j F_{\overline{x}_j} = F_{\overline{x}_j}(F_{x_j} + \overline{x}_j)$$

and by complementing we obtain a simplified formula:

$$\overline{F} = x_j \overline{F}_{x_j} + \overline{F}_{\overline{x}_j}, \tag{7}$$

and when $F_{x_j} \geq F_{\overline{x}_j}$

$$\overline{F} = \overline{x}_j \overline{F}_{\overline{x}_j} + \overline{F}_{x_j}. \tag{8}$$

In order to find an efficient algorithm to complement a unate function we will again represent F as a binary matrix $M(F)$. Let there be a one-to-one correspondence between columns of M and variables of F. Let each row of $M(F)$ correspond to a product term of M. Then

$$M_{ij}(F) = \begin{cases} 1, & \text{in } i\text{-term there is variable of column } j, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The matrix $M(F)$ will be directly used in complementation algorithm.

We illustrate the unate complementation algorithm with a following example.

Example. For the discernibility matrix $M(F)$ the appropriate function f_M is as follows:

$$f_M = (x_2 + x_3 + x_4)(x_1 + x_2)(x_3 + x_4)(x_2 + x_3 + x_5).$$

Performing the multiplication and applying absorption law we obtain:

$$f_M = x_2 x_3 + x_2 x_4 + x_1 x_3 + x_1 x_4 x_5.$$

The same result can be obtained performing double complementation of the function f_M (Fig. 1).

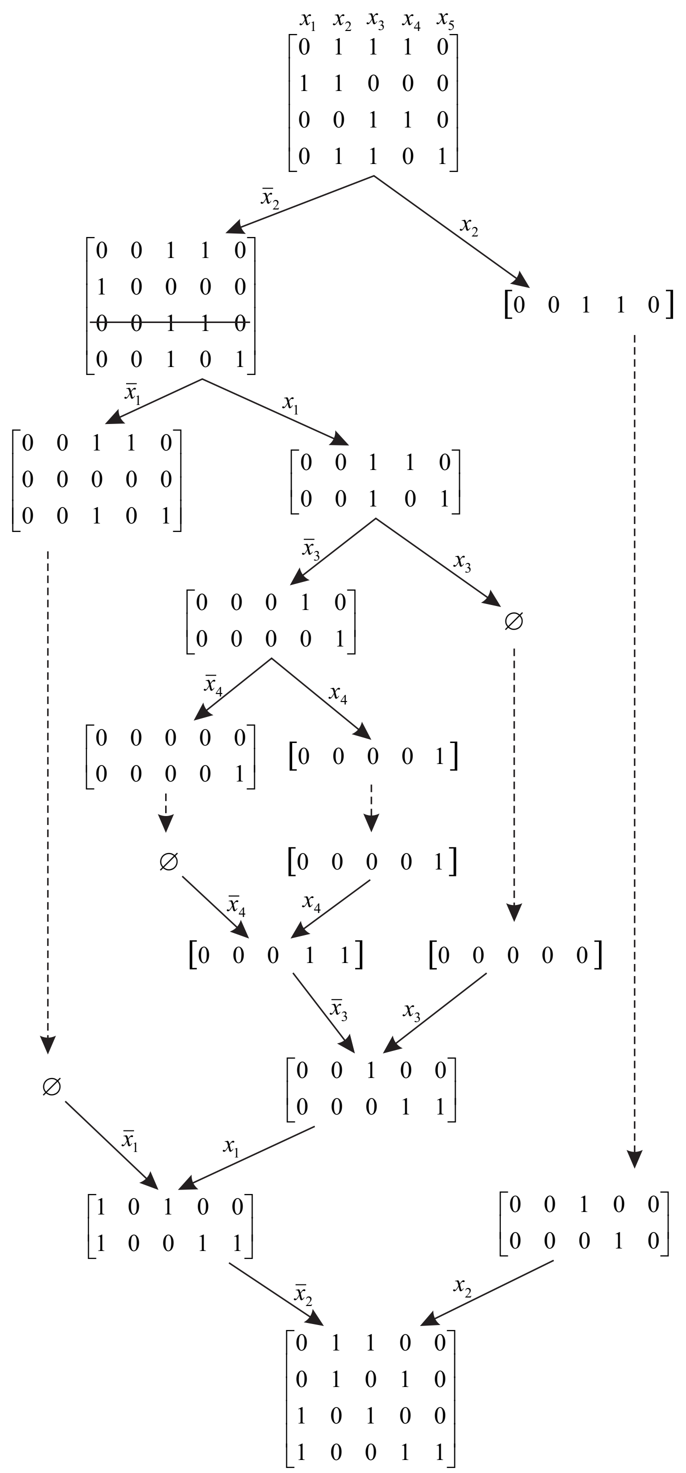

Fig. 1. Complement shcme of $F = \overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_1\overline{x}_2 + \overline{x}_3\overline{x}_4 + \overline{x}_2\overline{x}_3\overline{x}_5$.

Let $F = \overline{f}_M = \overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_1\overline{x}_2 + \overline{x}_3\overline{x}_4 + \overline{x}_2\overline{x}_3\overline{x}_5$. Hence:

$$M(F) = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

In order to identify the splitting variables we choose them among the shortest terms in F. Here we select the second term, yielding variables x_1 and x_2. Since the variable that appears most often in the other terms of F is x_2, we decide to choose this one.

Now we compute the cofactors of F with respect to the variable x_2:

$$F_{\overline{x}_2} = \overline{x}_3\overline{x}_4 + \overline{x}_1 + \overline{x}_3\overline{x}_5, \qquad M(F_{\overline{x}_2}) = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$F_{x_2} = \overline{x}_3\overline{x}_4, \qquad M(F_{x_2}) = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

The cofactor with respect to $\overline{x}_j$ is obtained by setting up the j-th column to '0', and the cofactor with respect to x_j is obtained by excluding all the rows for which the j-th element is equal to '1'.

In each branch of the recursion we examine the possibilities for complementation using easily computable special cases, i.e.:

a) There is a row of all 0's in $M(F)$ (empty conjunction is equal **1**) – the complementation is equal to **0** (empty set),
b) $M(F)$ is empty (empty disjunction is equal **0**) – the complementation is equal to **1** (the row of all 0's),
c) $M(F)$ has only one row – the complementation is calculated by applying De Morgan Law to the unique term.

In our example the complement of F_{x_2} is:

$$\overline{F}_{x_2} = x_3 + x_4, \qquad M(\overline{F}_{x_2}) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$F_{\overline{x}_2}$ must be processed further and yields:

$$F_{\overline{x}_2\overline{x}_1} = \mathbf{1}, \qquad M(F_{\overline{x}_2\overline{x}_1}) = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$F_{\overline{x}_2 x_1} = \overline{x}_3\overline{x}_4 + \overline{x}_3\overline{x}_5, \qquad M(F_{\overline{x}_2 x_1}) = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

and hence $\overline{F}_{\overline{x}_2\overline{x}_1} = \mathbf{0}$.

For $F_{\overline{x}_2 x_1}$ we have:

$$F_{\overline{x}_2 x_1 \overline{x}_3} = \overline{x}_4 + \overline{x}_5, \qquad M(F_{\overline{x}_2 x_1 \overline{x}_3}) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$F_{\overline{x}_2 x_1 x_3} = \mathbf{0}, \qquad M(F_{\overline{x}_2 x_1 x_3}) = \emptyset$$

The complement of $F_{\overline{x}_2 x_1 \overline{x}_3}$ is:

$$\overline{F}_{\overline{x}_2 x_1 \overline{x}_3} = x_4 x_5, \qquad M(\overline{F}_{\overline{x}_2 x_1 \overline{x}_3}) = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

and the complement of $F_{\overline{x}_2 x_1 x_3}$ is:

$$\overline{F}_{\overline{x}_2 x_1 x_3} = \mathbf{1}, \qquad M(\overline{F}_{\overline{x}_2 x_1 x_3}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

By merging these results we obtain

$$\overline{F}_{\overline{x}_2 x_1} = x_3 \overline{F}_{\overline{x}_2 x_1 x_3} + \overline{F}_{\overline{x}_2 x_1 \overline{x}_3} = x_3 \cdot \mathbf{1} + x_4 x_5 = x_3 + x_4 x_5$$

$$M(\overline{F}_{\overline{x}_2 x_1}) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

then

$$\overline{F}_{\overline{x}_2} = x_1 \overline{F}_{\overline{x}_2 x_1} + \overline{F}_{\overline{x}_2 \overline{x}_1} = x_1(x_3 + x_4 x_5) + \mathbf{0} = x_1 x_3 + x_1 x_4 x_5$$

TABLE II
RESULTS OF ANALYSIS THE PROPOSED METHOD IN COMPARISON TO RSES AND ROSE2 DATA MINING SYSTEMS

database	attributes	instances	ROSE2	RSES/ROSETTA	compl. method	reducts
house	17	232	1s	1s	187ms	4
breast-cancer-wisconsin	10	699	discerns missing	2s	823ms	27
kaz	22	31	out of memory (33min)	70min	234ms	5574
trains	33	10	discerns missing	out of memory (5h 38min)	6ms	689
agaricus-lepiota-mushroom	23	8124	discerns missing	29min	4min 47s	507
urology	36	500	out of memory (3h 27min)	out of memory (12h)	42s 741ms	23437
audiology	71	200	discerns missing	out of memory (1h 17min)	14s 508ms	37367
dermatology	35	366	discerns missing	out of memory (3h 27min)	3min 32s	143093
lung-cancer	57	32	discerns missing	out of memory (5h 20min)	111h 57min	3604887

$$M(\overline{F}_{\overline{x}_2}) = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

and finally

$$\begin{aligned} \overline{F} = x_2\overline{F}_{x_2} + \overline{F}_{\overline{x}_2} &= x_2(x_3 + x_4) + x_1x_3 + x_1x_4x_5 = \\ &= x_2x_3 + x_2x_4 + x_1x_3 + x_1x_4x_5 \end{aligned}$$

$$M(\overline{F}) = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

We interpret the final expression as the set of reducts: $\{x_2, x_3\}$, $\{x_2, x_4\}$, $\{x_1, x_3\}$, $\{x_1, x_4, x_5\}$. Note that performing the multiplication and applying the absorption law we obtained the same set of reducts.

IV. EXPERIMENTAL RESULTS

Many computer data mining systems were developed. In particular, the well-known Rough Set Exploration System elaborated at the University of Warsaw. The system implements many advanced procedures. Some algorithms of RSES have also been embedded in the even more famous ROSETTA system located in the Biomedical Center in Sweden [24], [25].

ROSE2 (Rough Sets Data Explorer) is a software implementing basic elements of the rough set theory and rule discovery techniques. It has been developed at the Laboratory of Intelligent Decision Support Systems of the Institute of Computing Science in Poznań, basing on fourteen-year experience in rough set based knowledge discovery and decision analysis. All computations are based on rough set fundamentals introduced by Z. Pawlak with some modifications proposed by Słowinski and Ziarko [26].

These tools were used to compare them with the presented synthesis method (Table II). Experiments performed show that despite many efforts directed to the designing of an effective tools for attribute reduction, existing tools are not efficient.

The confirmation of this supposition are experiments with the system RSES. Our research has shown that this system can not process data tables with large number of indeterminacy. An important example from the literature is *trains* database. Applying the new method we generate 689 reducts, however RSES 333 only (not selecting the option 'Do not discern with missing values'). While running the system with the option 'Do not discern with missing values' selected it ends up after several hours of computation yielding 'Not enough memory' message. Note, that not selecting the option it takes into account missing values when calculating the discernibility matrix. Finally, we obtain a result that is different from the set of all minimal sets of attributes.

Another example confirming the absolute superiority of the proposed method is *kaz* function. It is the binary function of 21 arguments, used when testing advanced logic synthesis tools. RSES calculates all the 5574 reducts within 70 minutes. In comparison, the new procedure developed and implemented by the authors calculates the set of all reducts in 234ms.

Presented method was additionally proved on the typical databases of medicine, i.e. *audiology* database, *dermatology* database, *urology* database, *breast cancer* database and *lung cancer* database. Table II shows the computation time for all the minimum sets of attributes.

The experiments performed confirm that logic synthesis algorithms developed for the design of digital systems are much more effective than currently used algorithms in data mining systems.

Undoubtedly, in logic synthesis systems and hardware realizations we are almost always looking for these sets of arguments (reducts) which are both: minimal and least. However, the decision systems depend on all the minimal sets of attributes. For example, when considering a reduct of the least cardinality, it can include an attribute that its implementation is actually expensive. In particular, when we consider calculations for a medical diagnosis, it may be a parameter which express complicated or expensive examination, or a test which may have a negative impact on the health of the patient and it is not possible to carry out. Therefore, the reducts of higher cardinality could be sometimes easier to be applied/used in practice.

V. CONCLUSIONS

The argument reduction problem is of a great importance in logic synthesis. It is the basis of some functional transformations, such as parallel decomposition [18]. Combined with some other design techniques it allows us to reduce the size of implemented circuits.

In this paper we have described an important problem of attribute reduction. This concept, originating from artificial intelligence (namely the theory of rough sets), helps to deal with functional dependencies having redundant input attributes. We have presented a new exact algorithm for attribute reduction which is based on the unate complementation task. Experimental results which have been obtained using this approach proved that it is a valuable method of processing the databases.

REFERENCES

[1] S. Abdullah, L. Golafshan, and Mohd Zakree Ahmad Nazri, "Re-heat simulated annealing algorithm for rough set attribute reduction," *International Journal of the Physical Sciences*, vol. 6, no. 8, pp. 2083–2089, 2011.

[2] J. Bazan, H. S. Nguyen, S. H. Nguyen, P. Synak, and J. Wróblewski, "Rough set algorithms in classification problem," in *Rough Set Methods and Applications: New Developments in Knowledge Discovery in Information Systems*. Heidelberg: Physica-Verlag, 2000, vol. 56, pp. 49–88.

[3] R. Dash, R. Dash, and D. Mishra, "A hybridized rough-PCA approach of attribute reduction for high dimensional data set," *European Journal of Scientific Research*, vol. 44, no. 1, pp. 29–38, 2010.

[4] Z. Feixiang, Z. Yingjun, and Z. Li, "An efficient attribute reduction in decision information systems," in *International Conference on Computer Science and Software Engineering*, Wuhan, Hubei, 2008, pp. 466–469, DOI: 10.1109/CSSE.2008.1090.

[5] A.-R. Hedar, J. Wang, and M. Fukushima, "Tabu search for attribute reduction in rough set theory," *Journal of Soft Computing – A Fusion of Foundations, Methodologies and Applications*, vol. 12, no. 9, pp. 909–918, Apr. 2008, DOI: 10.1007/s00500-007-0260-1.

[6] S. Jing and K. She, "Heterogeneous attribute reduction in noisy system based on a generalized neighborhood rough sets model," *World Academy of Science, Engineering and Technology*, vol. 75, pp. 1067–1072, 2011.

[7] P. Kalyani and M. Karnan, "A new implementation of attribute reduction using quick relative reduct algorithm," *International Journal of Internet Computing*, vol. 1, no. 1, pp. 99–102, 2011.

[8] M. Kryszkiewicz and K. Cichoń, "Towards scalable algorithms for discovering rough set reducts," in *Transactions on Rough Sets I*, ser. Lecture Notes in Computer Science, J. Peters, A. Skowron, J. Grzymała-Busse, B. Kostek, R. Świniarski, and M. Szczuka, Eds. Berlin: Springer Berlin / Heidelberg, 2004, vol. 3100, pp. 120–143, DOI: 10.1007/978-3-540-27794-1_5.

[9] M. Kryszkiewicz and P. Lasek, "FUN: Fast discovery of minimal sets of attributes functionally determining a decision attribute," in *Transactions on Rough Sets IX*, ser. Lecture Notes in Computer Science, J. Peters, A. Skowron, and H. Rybiński, Eds. Springer Berlin / Heidelberg, 2008, vol. 5390, pp. 76–95, DOI: 10.1007/978-3-540-89876-4_5.

[10] D. Nguyen and X. Nguyen, "A new method to attribute reduction of decision systems with covering rough sets," *Georgian Electronic Scientific Journal: Computer Science and Telecommunications*, vol. 1, no. 24, pp. 24–31, 2010.

[11] Z. Pawlak, *Rough Sets. Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, 1991.

[12] X. Pei and Y. Wang, "An approximate approach to attribute reduction," *International Journal of Information Technology*, vol. 12, no. 4, pp. 128–135, 2006.

[13] A. Skowron and C. Rauszer, "The discernibility matrices and functions in information systems," in *Intelligent Decision Support – Handbook of Application and Advances of the Rough Sets Theory*, R. Słowiński, Ed. Kluwer Academic Publishers, 1992.

[14] C. Wang and F. Ou, "An attribute reduction algorithm based on conditional entropy and frequency of attributes," in *Proceedings of the 2008 International Conference on Intelligent Computation Technology and Automation*, ser. ICICTA '08, vol. 1. Washington, DC, USA: IEEE Computer Society, 2008, pp. 752–756, DOI: 10.1109/ICICTA.2008.95.

[15] Y. Yao and Y. Zhao, "Attribute reduction in decision-theoretic rough set models," *Information Sciences*, vol. 178, no. 17, pp. 3356–3373, 2008, DOI: 10.1016/j.ins.2008.05.010.

[16] T. Łuba, R. Lasocki, and J. Rybnik, "An implementation of decomposition algorithm and its application in information systems analysis and logic synthesis," in *Rough Sets, Fuzzy Sets and Knowledge Discovery*, W. Ziarko, Ed. Springer Verlag, 1994, pp. 458–465, Workshops in Computing Series.

[17] T. Łuba and R. Lasocki, "On unknown attribute values in functional dependencies," in *Proceedings of The Third International Workshop on Rough Sets and Soft Computing*, San Jose, 1994, pp. 490–497.

[18] M. Rawski, G. Borowik, T. Łuba, P. Tomaszewicz, and B. J. Falkowski, "Logic synthesis strategy for FPGAs with embedded memory blocks," *Electrical Review*, vol. 86, no. 11a, pp. 94–101, 2010.

[19] H. Selvaraj, P. Sapiecha, M. Rawski, and T. Łuba, "Functional decomposition – the value and implication for both neural networks and digital designing," *International Journal of Computational Intelligence and Applications*, vol. 6, no. 1, pp. 123–138, March 2006, DOI: 10.1142/S1469026806001782.

[20] G. Borowik, T. Łuba, and D. Zydek, "Reduction of knowledge representation using logic minimization techniques," in *ICSEng*, 2011, pp. 482–485, DOI: 10.1109/ICSEng.2011.98.

[21] J. A. Brzozowski and T. Łuba, "Decomposition of boolean functions specified by cubes," *Journal of Multi-Valued Logic & Soft Computing*, vol. 9, pp. 377–417, 2003.

[22] T. Łuba and J. Rybnik, "Rough sets and some aspects in logic synthesis," in *Intelligent Decision Support – Handbook of Application and Advances of the Rough Sets Theory*, R. Słowiński, Ed. Kluwer Academic Publishers, 1992.

[23] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, 1984.

[24] "ROSETTA – A Rough Set Toolkit for Analysis of Data." [Online]. Available: http://www.lcb.uu.se/tools/rosetta/

[25] "RSES – Rough Set Exploration System." [Online]. Available: http://logic.mimuw.edu.pl/ rses/

[26] "ROSE2 – Rough Sets Data Explorer." [Online]. Available: http://idss.cs.put.poznan.pl/site/rose.html