DECOMPOSITION OF MULTIPLE-VALUED RELATIONS

Marek Perkowski, Malgorzata Marek-Sadowska †, Lech Jozwiak +, Tadeusz Luba •, Stanislaw Grygiel, Miroslawa Nowicka •, Rahul Malvi, Zhi Wang, and Jin S. Zhang
Portland State Univ., EE Dept., Portland, OR 97207, Tel: 503-725-5411, mperkows@ee.pdx.edu, † U.C., Santa Barbara, ECE Dept., CA 93106, mms@ece.ucsb.edu, + Faculty of EE., Eindhoven Univ. of Techn.,P.O. Box 513, 5600 MB Eindhoven, The Netherlands, lech@eb.ele.tue.nl, • Warsaw Univ. of Techn., Inst. of Telecomm., Warszawa, Nowowiejska 15/19, Poland, luba@tele.pw.edu.pl

Abstract— This paper presents a new decomposition problem: decomposition of multi-valued (MV) relations, and a method of its solution. Decomposition is non-disjoint and multi-level. A fundamental difference in decomposition of MV functions and MV relations is discussed: the column (cofactor) pair compatibility translates to the group compatibility for functions, but not for relations. This makes the decomposition of relations more difficult. The method is especially efficient for strongly unspecified data typical for Machine Learning (ML). It is implemented in program GUD-MV. [1]

I. Introduction.

Functional Decomposition of switching functions has applications in binary and multiple-valued circuit design, Machine Learning (ML), and Knowledge Discovery from Data Bases (KDD). Despite the fundamental nature of the MV decomposition problem and many possible applications of its solutions, efficient MV decomposers do not exist yet, with the exception of [1]. (The Curtis decomposition of binary functions is presented in detail in [4], Curtis-like decomposition of multi-valued functions based on graph coloring was presented in [6]). In this paper we will focus on a new problem of Curtis-like Decomposition of MV Relations. We present also an efficient computer program for this task. The solution of the MV Relation Decomposition Problem finds numerous applications in Machine Learning, binary circuits and Finite State Machine design.

An example of a relation with binary inputs and a single MV output is shown in Table 1. Observe, that only the care minterms (care cubes) are present in the relation table as its rows. Standard don't cares ("unknown data samples" in ML) are represented by the remaining, implicit, minterms. The values in the column for output variable f include also the so-called *"generalized don't cares"*. For instance, assume the meaning of the values of decision variable f: 0 - a chair, 1 - an armchair, 2 - a desk, 3 - a table, etc. Then, the position {0,1} in the first row will mean "a chair or an armchair", which means, something is known but the answer is not precize. The value 0 means a definite answer "a chair", and a value {0,1,2,3,4} would mean a complete unknown, a standard don't care, denoted by "-". Observe in Table 1 that there is no row for a standard don't care at all (like there is no row for a minterm $abcd$ = 1111). In general, for a single-output relation (like one from Table 1), there will be no row for a standard don't care. In case of relations with two or more output variables, it can happen, however, that one of these variables, say f, is a standard don't care, and another one, g, has a proper subset of its possible values. In such case the row will exist in the table (the standard don't care for f has the meaning of a generalized don't care, with all possible values of this variable). Similar tables can be presented for multi-valued inputs as well [5].

	a	b	c	d	f
0	0	0	0	0	0,1
1	0	1	0	0	1,2
2	1	1	0	0	0
3	0	0	0	1	0,3
4	1	1	0	1	0,3
5	1	0	0	1	0,4
6	1	0	0	0	0,3
7	0	0	1	1	1,3
8	0	1	1	1	0,1
9	0	0	1	0	2,3
10	1	0	1	0	1,4
11	0	1	1	0	2,3

Table 1: *Multivalued Relation*

In multiple-valued systems, the entire classical decomposition approach is considerably more complex than for binary systems because of the associated combinatorial explosion. However, this is not the case for weakly specified relations and functions, and an appropriate decomposition approach can be made efficient by utilizing don't cares. It can be observed that in the area of circuit design the percent of don't cares is not more than 90%. While in Machine Learning, this percent is usually larger than 99%. Arbitrarily, we will define the functions with more than 95% don't cares to be weakly specified (they

[1] This research was partially supported by the Dutch Technology Foundation (STW) under project EEL 55.3958

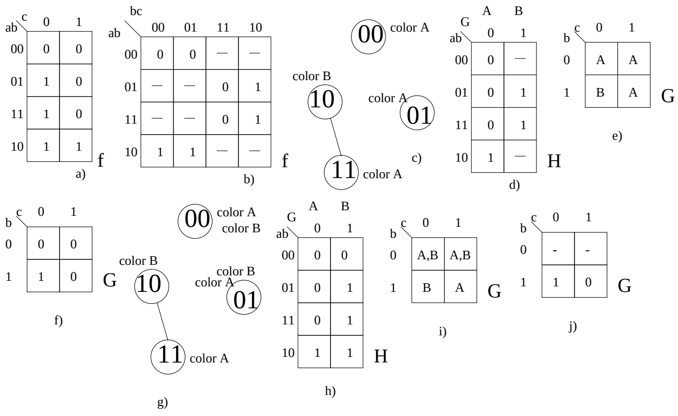

Figure 1: *RVM Maps and Decomposition: (a) The standard map of the function f from Example 1, (b) the RVM map of the function f from Example 1 created from the map in (a). Coloring: (c) the coloring graph with colors of nodes for a Binary Ashenhurst Decomposition of f from Example 2, (d) map of incomplete function H to Example 2, (e) map of function G, (f) encoded map of function G. Multi-Coloring: (g) the multi-coloring graph for f from Example 2, (h) map of function H to Example 2, (i) map of relation G, (j) encoded map of incomplete function G.*

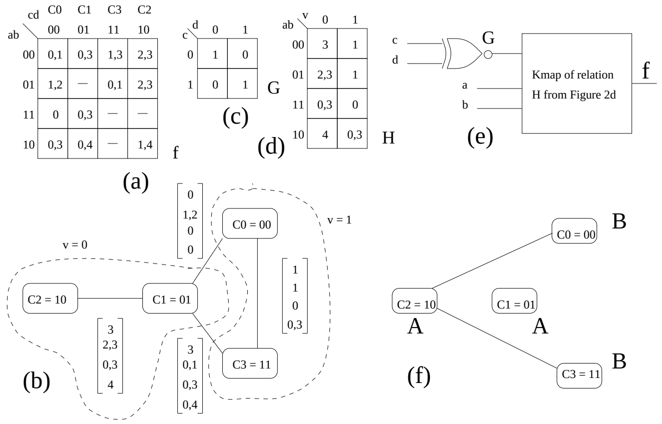

Figure 2: *Compatibility for Decomposition of MV Relations to Example 3. G is a function, H is a relation.*

are called also the strongly unspecified functions). Similarly, we will define the relations with more than 95% don't cares (total, standard and generalized), the weakly specified, or strongly unspecified relations. Observe also, that the more values exist in generalized don't cares, the more is the relation unspecified. The less values exist in generalized don't cares, the relation is more similar to a function.

II. Repeated Variable Maps and Non-Disjoint Decompositions

We discuss Curtis-like decomposition $F = H(G(B \cup C), A \cup C)$. The set X of input variables is partitioned to two sets: *free variables* $A \cup C$ (using Curtis terminology) are direct inputs to the successor block H, and *bound variables* $B \cup C$ are inputs to the predecessor block G of the non-disjoint decomposition. For relation F with $C = \emptyset$ represented as a Karnaugh map with B variables as columns and A variables as rows, the *column multiplicity index* μ is the number of different types of column patterns. By *columns* we will understand the cofactors of F with respect to the variables from the bound set. The problem that we want to formulate and solve in this paper is the following. Given is a multivalued, strongly unspecified relation, with many input variables, and many output variables. Each variable can have a different number of values (from 2 to hundreds). Find the hierarchical decomposition of this function to a DAG (Directed Acyclic Graph), with nodes of the multi-level DAG representing blocks (each block representing a relation or in particular a function), and arrows representing variables and intermediate variables, in such a way that the total cost RC of the network of these blocks will be the minimum. The intermediate signals created in the decomposition may be multivalued, but smaller numbers of values in them are preferable to decrease the cost. Similarly, nonempty sets C may be assumed (non-disjoint decompositions), but smaller sets C are preferable with respect to the total cost. This is a decomposition model more general than known in the literature [6].

First, we explain how to perform non-disjoint decompositions of multi-output relations with use of the Repeated Variable Maps. A multivalued relation can be stored in a tabular representation (such as a Karnaugh map or MV map) in which for every entry there exist one of the following: (1) a single value of the output (as in mv functions), (2) a standard don't care represented by a dash, (3) a generalized don't care represented by a set of values of the output variable. Example of such map for relation from Table 1 is shown in Figure 2a.

Definition 1 *A Repeated Variable Map (RVM) of a multivalued relation is its tabular representation in which entries store the values corresponding to the input combinations assigned to the rows and columns. The row variables correspond to the free set $A \cup C$, and the column variables correspond to the bound set $B \cup C$. Note that the variables from the non-empty set C are repeated (called also shared variables). The entries of RVM whose row and column combination values of C variables are the same, take the values specified for these variables in the original standard map. The entries can be any of the three cases (1),(2),(3) specified above. The entries for which the C-values are different, are set to standard don't cares (represented by dashes). (This is because the same variable cannot have different values simultaneously).*

Example 1. An example of an RVM is shown in Figure 1.

Fig. 1a presents a standard Kmap of a 3-input function f. Assuming b is a repeated variable, the Bound Set $\{b,c\}$ (the columns) and the Free Set $\{a,b\}$ (the rows), one creates the RVM from Fig. 1b. The set of *Rows* of rows is composed of $\{R_0 = \overline{a}\overline{b},\ R_1 = \overline{a}b, R_2 = a\overline{b},\ R_3 = ab\}$. The set of $Columns = \{C_0 = \overline{b}\overline{c},\ C_1 = \overline{b}c, C_2 = b\overline{c},\ C_3 = bc\}$. This map illustrates the principle of repeating variables that leads to a new, strongly unspecified function. In other representations than Karnaugh Maps, a repetition of variable b means creation of two variables: b_{row} and b_{column}. While RVM is only a dydactic concept, the functions with repeated (renamed) variables can be represented in any known data structure that allows for don't cares [5, 2, 3, 1].

Definition 2 *Two single-output MV relations $F1$ and $F2$ are a* relation tautology *if for every entry in their standard maps they have at least one value in common. Two multi-output MV relations $F1$ and $F2$ are a* relation tautology *if they are single-output tautologies for every output separately.*

Obviously this definition is a generalization of the definition of incomplete tautology, where for each entry the functions are either the same, or at least one is a standard don't care.

Definition 3 *We define that MV relation F has a μ-valued* Curtis Decomposition *of the form $H(G(B \cup C), A \cup C)$, with a given bound set $B \cup C$, a given free set $A \cup C$, and with μ values in signal G, when relation F and composition of relations $H(G(B \cup C), A \cup C)$ are a relation tautology.*

Existence of the *MV Curtis Decomposition* for a given bound set $B \cup C$ and free set $A \cup C$ can be checked using Theorem 1.

Theorem 1 *The non-disjoint μ-valued Curtis decomposition exists if the column multiplicity index in the corresponding RVM map is μ. (see proof in [2]).*

Observe, that the MV Ashenhurst decomposition, a generalization of Ashenhurst decomposition for MV functions, which assumes a single binary signal G, is a special case of MV Curtis decomposition with $\mu = 2$. *Disjoint decompositions* are those that decompose function F to two subfunctions G and H that have disjoint sets of inputs variables. Most authors differentiate between *disjoint* and *non-disjoint decompositions,* and most of the MV decompositions reported in literature are disjoint. The RVMs can be used to explain all the decomposition types in a uniform way. As introduced before, if $C = \emptyset$ the decomposition is called *disjoint* and the RVM becomes a standard Karnaugh Map. If $C \neq \emptyset$ the decomposition is *non-disjoint* and the RVM is incompletely specified, even if the original function is completely specified. The process of finding sets A, B and C is called *input variable partitioning*, and we have proposed several efficient algorithms for it [7, 2, 1]. Observe, that addition of each repeated variable increases the map dimension, and all the newly introduced cells are don't cares. For instance, if the original map is completely specified and has 4 variables a, b, c, d, the bound set is $\{a, c, d\}$ and the variable a is a repeated variable, the new 4×8 map will have three variables for columns and two variables for rows (variable a appears in both rows and columns). Half of the entries in this RVM are now don't cares. If variables a and c were repeated, and $\{a, c, d\}$ is the bound set, the new 8×8 map will have three column variables, and three row variables. There will be 75% don't cares in this case. Starting even with a completely specified function, by repeating variables, the function becomes very strongly unspecified. Since "don't cares" represent "design freedom", this fact shows, why it is possible to find a decomposition or to find a better decomposition by introducing more repeated variables. In addition, in ML applications, even the initial data can have more than 99.99% of don't cares. This percent grows with the size of the real life Machine Learning benchmarks. Therefore, it is absolutely crucial to represent and manipulate the weakly specified functions and relations efficiently (we use the data structure introduced in [5]).

III. Column Compatibility for MV Functions and Relations

For relations [3], the decomposition problem has not been discussed in the literature, and thus all the notions below are new.

Definition 4 *Two columns C_1 and C_2 of an MV Relation form a* **pair of compatible columns** *if in each row there exists at least one value that is the same in both columns. In other words, if in any row the intersection of the sets of values of $C_1[R_i]$ and $C_2[R_i]$ is non-empty:* $C_1 \cong C_2 \quad \Leftrightarrow \ (\forall R_i \in Rows)\ [\ [C_1[R_i] \ \cap \ C_2[R_i] \neq \ \emptyset\ ]$

Definition 5 *A set of columns COL forms a* **compatible set of columns of an MV Relation** *iff for any row there exists some nonempty set of values that is included in all columns in COL: $COMP_{Rel}(COL) \quad \Leftrightarrow$ $(\forall R_i \in Rows)\ [\ \bigcap_{C_j \in COL} C_j[R_i] \ \neq \ \emptyset\ ]$*

For instance, columns C_0 and C_3 in the MV relation from Figure 2a are compatible. Columns C_0 and C_2 are not compatible. Although the columns C_0 and C_3, C_0 and C_1, and C_1 and C_3, are pairwise compatible, the set of columns $COL = \{C_0, C_1, C_3\}$ is not a compatible set of columns. Columns that are not compatible are called *incompatible.* The above compatibility notions can be defined for compatible rows in a strictly analogous manner.

Definition 6 *In the* **Column Compatibility Graph** *(CCG) the nodes correspond to columns in the RVM map* *(cofactors of the bound set). If two columns are compatible, there is an edge between the corresponding nodes. The* **Column Incompatibility Graph** *(CIG) has the same set of nodes as the CCG. An edge between two nodes exists, if the corresponding columns are incompatible.*

CCG and CIG are complementary graphs, i.e. their union forms a complete graph, because any two columns are either compatible (exclusive)or incompatible.

Property 1. Compatibility inheritance property. The Column Compatibility Graph satisfies the compatibility inheritance property on a set of columns $COL \subseteq$ $Columns$ if compatibility of all 2-column subsets in COL implies compatibility of the entire set COL: $(\forall COL \subseteq Columns)$ $(\forall C_i, C_j \in COL)$ $[C_i \sim C_j] \Rightarrow COMP(COL)$. For instance, if COL $= \{C_0, C_3, C_7, C_9\}$ and all possible pairs in the set are compatible, i.e. $C_0 \sim C_3, C_0 \sim C_7, C_0 \sim C_9, C_3 \sim C_7, C_3 \sim C_9, C_7 \sim C_9$, then fulfillment of Property 1 implies that $\{C_0, C_3, C_7, C_9\}$ form a compatible set.

Theorem 2 *In an MV function, the relation of compatibility $\approx$ satisfies Property 1. In an MV relation, the relation of compatibility $\cong$ does not satisfy Property 1.*

Proof. *Proof for MV function directly follows from Definitions 4 and 5. To prove for MV relation it is sufficient to find a counter-example, see Example 3.*

Checking the incompatibility of cofactors is what every Curtis-like decomposer does most of the time, so this operation must be efficiently programmed. From the Theorems 1 and 2 it follows that for MV functions we should use proper graph coloring algorithms on CIG to determine the minimum column multiplicity index μ. In proper graph coloring every two nodes linked by an edge are colored with different colors and the total number of colors should be minimal. Nodes assigned the same color form a clique in the CCG graph and correspond to a compatible set of columns. Subsequently they can be combined into one column. The number of colors in the exact minimum coloring, called the *chromatic number* of this graph, is equal to the *column multiplicity index* μ for the given bound set $B \cup C$. *Proper Multi-coloring* is like proper coloring, but a node can be colored with many colors. This corresponds to overlapping cliques in the compatibility graph, and thus to both G and H being relations.

Example 2. The Incompatibility Graph for the function in Fig. 1b with bound set $\{b, c\}$ is shown in Fig. 1c. The coloring is: nodes 00$=C_0$, 01$=C_1$ and 11$=C_3$ with color A; and node 10$=C_2$ with color B. Now columns 00, 01 and 11, colored with color A (a clique in the corresponding Compatibility Graph), can be combined, which creates a map of the successor block H in Fig. 1d. The map of the predecessor block G is also obtained from this clique partitioning, Fig. 1e. After assigning binary codes A $= 0$, B $= 1$, the solution $G = b\overline{c}$, $H = G + a\overline{b}$ $(f = b\overline{c} + a\overline{b})$ is found with blocks H and G from Fig. 1d, and Fig. 1f, respectively. Similarly, solutions $f = (a+b)(\overline{b} + \overline{c})$ and $f = bc \oplus (a+b)$ are found for the same bound set, but with different colorings. One can verify that there is no binary disjoint solution with bound variables a, b, since three different rows exist in the map in Fig. 1a. Thus, the multiplicity index for bound set $\{a, b\}$ is $\mu = 3$. Similarly, for bound sets $\{a, c\}$ and $\{b, c\}$ $\mu = 3$ and there are no binary disjoint decompositions. However, there exists a three-valued decomposition $f = H(G(a, b), c)$ with 3-valued function G, and binary-output function H. Figure 1g presents the graph with multi-coloring, nodes 00 and 01 are colored with colors A and B. Corresponding function H is in Fig.1h. Observe, that it has less don't cares than the H from Fig. 1d obtained from coloring. In contrast, G is now a relation (Fig. 1i). Concluding, by the switching between coloring and multi-coloring procedures, and by controlling the size of sets of nodes colored with single colors, we can constrain any of relations G or H to become functions. We can also investigate trade-offs between percentages of standard don't cares, generalized don't cares and specified transitions in G and H.

In the case of CIG graph for MV Relations, the Proper Graph Coloring or Multi-Coloring cannot be used, since for every group of pairwise compatible nodes one has to check if all these nodes (the columns that correspond to them) satisfy Definition 5. This kind of graph coloring is called *Compatible Graph Coloring*, or *Compatible Graph Multi-Coloring*, respectively. The method to create a combined column is the same for functions and relations. The difference is only in the graph coloring. During node-by-node compatible coloring of a CIG corresponding to a relation the **sets of nodes** colored with the same color are additionally checked for compatibility. This makes compatible coloring slower than the proper coloring, and also more memory is needed to store the combined columns. Every step of Compatible Graph Coloring creates a set of compatible columns for the relation. When the coloring is completed, the minimum set of sets of compatible columns exists. (We compared exact and heuristic algorithms and proved that heuristic multicoloring gives nearly minimum results on binary and MV *benchmark* functions). In each set of compatible columns the columns are combined into a single column, and next new relations G and H are created. Observe, that even if we start from function, this process creates relations during decomposition. These relations are subject to next decompositions. This is one more argument why the decomposition of relations is an important and practical problem.

IV. Curtis-like Decomposition of MV Relations

Our Curtis-like Decomposer can handle both MV Functions and Relations. In the case of a MV relation, the CCG graph can be created with nodes for columns, and edges for pairs of compatible nodes. Two compatible columns C_i and C_j of RVM can be combined, and every combined cell $C_{ij}(R_s) := C_i(R_s) \cap C_j(R_s)$. As shown above, standard maximum cliques cannot be used for MV relations, because, contrary to the standard column compatibility, column C_1 could be compatible with column C_2, column C_2 compatible with column C_3, and column C_3 compatible with column C_1, but columns C_1, C_2, and C_3 are not compatible all together as a set. Therefore, the cliques in the CCG graph must be checked for set compatibility $COMP_{Rel}$. This is equivalent to building a CIG graph, and coloring it using a Compatible Graph (Multi) Coloring algorithm. Such algorithm checks every group of nodes colored with the same color for the set compatibility of all columns corresponding to them.

Example 3. Given a relation with 4 binary variables and a 5-valued output variable from Table 1 the map from Fig. 2 is created. The bound set is $\{c,d\}$. A don't care symbol, "-", stands for a set of values $\{0,1,2,3,4\}$. Every cell that includes a *set* of values with more than one value is a generalized don't care. If at least one cell like this exists in a map, the map describes a relation. Recall, that the interpretation of such a map is that in every cell with many values, any value that simplifies the overall description can be selected. The Column Compatibility Graph is presented in Fig. 2b. The nodes represent the columns from the map in Fig. 2a. A column in brackets shown near the edge between nodes C_i and C_j represents the combined column C_{ij}. As we see, nodes $C_0 = 00$ and $C_2 = 10$ are not compatible, since for instance $\{0,1\} \cap \{2,3\} = \emptyset$ in cofactor $\overline{a}\overline{b}$, thus $C_{02} = \emptyset$. The nodes C_0, C_1 are compatible, so $C_{01} \neq \emptyset$. Although the nodes C_0, C_1 and C_3 are pairwise compatible, the maximum clique from nodes C_0, C_1 and C_3 cannot be used, since $C_{01} \cap C_{03} \cap C_{13} = \emptyset$, which means that the set of columns $\{C_0, C_1, C_3\}$ do not form a compatible set.

The solution obtained from the relation Column Compatibility Graph includes the cliques $\{C_0,C_3\}$ and $\{C_1,C_2\}$. Similarly, the CIG graph can be obtained as a complement of the CCG graph (Fig. 2f) (Multi)-coloring this graph leads to the same solution: columns C_0 and C_3 are colored with color A, and columns C_1 and C_2 are colored with color B. Columns C_0 and C_3 are thus combined to the single column (encoded with v=1 in Fig. 2d). Columns C_1 and C_2 are combined to the column encoded with v=0 in Fig. 2d. We build the map of relation H from Fig. 2d, and from it and the map from Fig. 2a we build the map of relation G (function G in this case) from Figure 2c. This corresponds to the decomposed circuit from Fig. 2e. The relations G and H can be further decomposed or simplified using other methods [3].

Example 4. Figure 3 presents a decomposition of function f (Fig. 3a) to relations G and H, Figs. 3c and 3d, respectively. Clique covering is shown in Fig. 3b. Composition of relations H and G is shown in Fig.3e. The map for this composition is in Fig. 3f. Correctness of decomposition can be verified by finding the intersection of maps from Fig. 3a and Fig. 3f (shown in Fig. 3g). All its

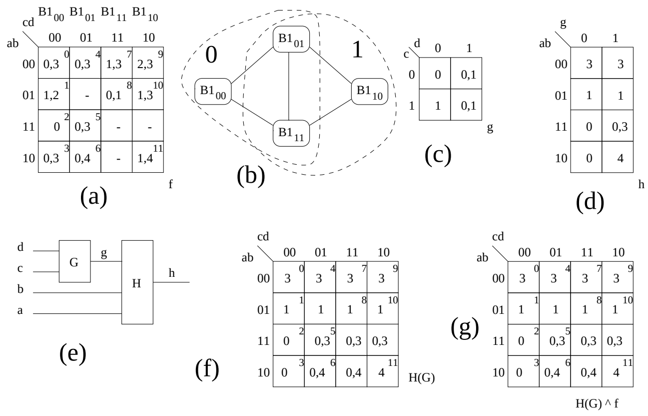

Figure 3: *Compatibility for Decomposition of MV Relations to Example 4. Both G and H are relations.*

entries are non-empty, so $H(G)$ and f are a relation tautology. All solutions of MV-GUD have been verified using the compositional relation tautology verifier we wrote.

Multi-level decomposition consists in decomposing the initial relation into consecutive pairs of relations G and H until the minimum decomposable blocks are obtained. One decomposition step consists in determination of a set of good partitions (X_1, X_2) based on certain heuristic criteria [7, 2], selecting the best one, and performing decomposition. Decomposition is performed only if it results in smaller complexity of a relation, and we use *Relation Cardinality* (RC) as a complexity measure.

Definition 7 Relation Cardinality *(RC) for MV relation with a set of inputs $X = \{x_0, x_1, \ldots, x_n\}$ and set of outputs $Y = \{y_0, y_1, \ldots, y_m\}$ is defined by the following formula: RC $= (\prod_{x_i \in X} m_{x_i}) \sum_{y_j \in Y} \log_2 m_{y_j}$ where: m_{x_i} is multiplicity of variable $x_i \in X$, and m_{y_j} is multiplicity of variable $y_j \in Y$.*

The above definition is based on information theory and RC is directly related to the amount of information the relation could possibly handle. The amount of information is defined, in the simplest case, to be measured by the logarithm of available choices. We use logarithm to the base 2 and express the amount of information in bits. So the value of $I(X) = \sum_{x_i \in X} \log_2 m_{x_i}$ is equal to the amount of information a relation could possibly handle if the relation output is binary. The total number of available choices (relation cardinality) is then equal to $2^{I(X)} = \prod_{x_i \in X} m_{x_i}$. If relation's output is multivalued it is equivalent to $\log_2 m_y$ binary outputs, and the relation itself, equivalent to $\log_2 m_y$ binary output relations (blocks). Definition 7 extends this formula to the general case of multioutput, multivalued relation. RC driven decomposition splits a relation into smaller blocks in such a way that the total RC value, equal to the sum of RCs of decomposed blocks, be minimal. Such procedure follows

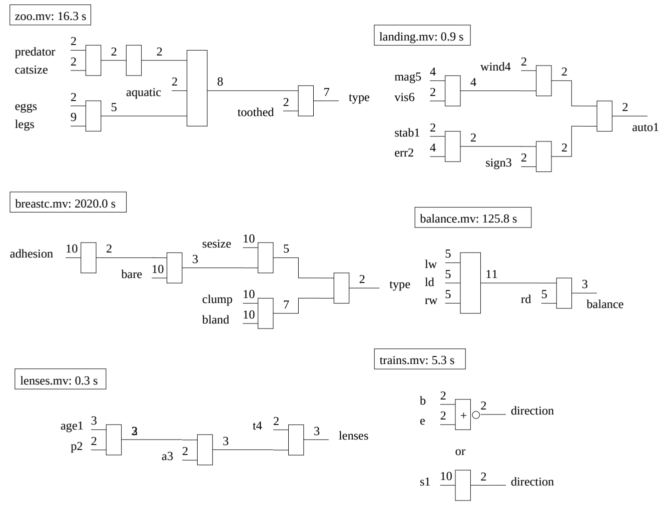

Figure 4: *Solutions for multi-valued ML Benchmarks.*

	input file		output file		
	# of inputs	# of cubes	# of inputs	# of blocks	time[s]
zoo	16	101	6	5	16.3
shuttle	6	15	6	5	0.9
breastc	9	699	5	5	2020.0
balance	4	625	4	2	125.8
lenses	4	24	4	3	0.3
trains	32	10	2(1)	1	5.3

Figure 5: *GUD-MV decomposition results on multi-valued ML benchmarks.*

Occam's Razor principle that we should always accept the simplest solution that correctly fits the data. In our case the cost function defining simplicity of a solution is RC which reduces the number of possible combinations (choices) of variable values without reducing functionality. In case of a tie for RC value, additional criteria are used to select the best block and more unspecified relations are given preference because they lead to simpler circuits.

V. Experimental Results

Table 2 and Figure 4 show the results of decomposition of selected benchmarks from University of California, Irvine ML data base. Decomposed functions are in most cases much smaller then the initial ones and depend on fewer input variables. For testing, we used in particular: **zoo:** Zoo Database. Created and donated by Richard S. Forsyth. **shuttle:** Space Shuttle Autolanding Database. **breastc:** Breast Cancer Database. Donated by the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. **balance:** Balance Scale Weight & Distance Database. **lenses:** Fitting Contact Lenses Database. **trains:** INDUCE Trains Data set. Let us observe that both input, intermediate and output variables can be multivalued, and the numbers of the values differ. Smaller representation is usually equivalent to better generalization properties in ML and KDD. Since most of the ML data sets are only tiny representations of the full data domain for a particular problem, good generalization properties are very important when we want to determine function values for input data not contained in the sample. Also, fewer input variables means that some of the original input variables are vacuous, i.e. they don't provide any essential information for the function value determination and can be removed without affecting the result. Let's take as an example the well known benchmark **trains** from book "Machine Learning" by Michalski. It is considered to be difficult test case for ML programs. Running GUD-MV decomposer on it we obtained two solutions: (1) **direction** $= \neg(b \vee e)$, and (2) $if(s1 = 5 \vee s1 = 6 \vee s1 = 9)$ **then direction = 1; else direction = 0;**. The first solution depends on two binary variables, the second, on one MV variable only. The number of input variables of the initial data set was 32! Similar phenomena, but to a lesser extent, are observed also in controller design (assuming that the don't cares were not artificially treated as binary constants, which is sometimes an industrial practice).

VI. Conclusions

In this paper we formulated a research problem not yet tackled by previous researchers - decomposition of multi-valued relations, and we proposed an efficient method to solve it (some of our test cases are known to be difficult in KDD community, and our solutions have small values of RC - see Fig. 4). The decomposition forms multi-level structures, and is applied to blocks with multiple-valued inputs and multiple-valued outputs. Program GUD-MV is, to our best knowledge, the first decomposer for MV relations ever implemented. Decomposition of relations will find applications in binary circuit and state machine design, Machine Learning and KDD.

References

[1] C. Files, R. Drechsler, and M.A. Perkowski, "Functional Decomposition of MV Functions using Multi-Valued Decision Diagrams," *Proc. ISMVL'97.*

[2] M.A. Perkowski et al. "Full version of this paper".

[3] R. Brayton and F. Somenzi, "An Exact Minimizer for Boolean Relations," Proc. of ICCAD, pp. 316-320, 1989.

[4] H.A. Curtis, "A New Approach to the Design of Switching Circuits," *Princeton,* N.J., Van Nostrand, 1962.

[5] S. Grygiel, M. Perkowski, M. Marek-Sadowska, T. Luba, and L. Jozwiak, "Cube Diagram Bundles, A New Representation of Strongly Unspecified Multiple-Valued Functions and Relations," *Proc. ISMVL'97.*

[6] J.C. Muzio, and T.C. Wesselkamper, "Multiple-Valued Switching Theory," *Adam Hilger,* Boston, MA, 1986.

[7] W. Wan, and M. Perkowski, "A New Approach to the Decomposition of Incompletely Specified Multi-Output Function Based on Graph Coloring and Local Transformations and Its Application to FPGA Mapping," *Proc. Euro-DAC,* pp. 230 - 235, 1992.