Decomposition of Multi-Valued Functions into Min- and Max-Gates

Christian Lang
Institute for Microelectronic and Mechatronic Systems
Haarbergstr. 67, 99097 Erfurt, Germany
Christian.Lang@erfurt.imms.de

Bernd Steinbach
Institute for Computer Science
Freiberg University of Mining and Technology
09596 Freiberg, Germany
steinb@informatik.tu-freiberg.de

Abstract

This paper presents algorithms that allow the realization of multi-valued functions as a multi-level network consisting of min- and max-gates. The algorithms are based on bi-decomposition of function intervals, a generalization of incompletely specified functions. Multi-valued derivation operators are applied to compute decomposition structures. For validation the algorithms have been implemented in the YADE system. Results of the decomposition of functions from machine learning applications are listed and compared to the results of another decomposer.

1 Introduction

Machine learning seeks to extract knowledge from a given data base. If data is given as a multi-valued logic (MVL) function, it can be structured by decomposition and represented as a network of MVL gates [14]. To keep the results interpretable, the functions of the MVL gates should be understandable by humans. Many decomposition systems use a generalization of Curtis-decomposition where influence on the type of gates is limited [8, 5]. Bi-decomposition allows excellent control over the gates by the choice of the operator functions [1]. The minimum and maximum functions are among the simplest MVL gates. This paper shows properties of MVL functions that allow the decomposition of MVL functions into netlists of two-input min- and max-gates.

For machine learning and logic synthesis, some function values, the so called don't cares, are not specified. Functions containing don't cares are represented by incompletely specified functions (ISF). Values are assigned to the don't cares during the synthesis process. Basic bi-decomposition algorithms for MVL ISFs have already been developed [10, 6]. For multi-level decomposition however, sophisticated computation of the decomposition functions is necessary. In general, ISFs are insufficient to represent intermediate functions during decomposition efficiently [9]. Function intervals [11] are a generalization of ISFs that allow improved control over the don't cares. This paper shows that function intervals are very appropriate to store intermediate results during the min- and max-decomposition (MM-decomposition) process of ISFs.

MVL differential calculus [13] is the MVL extension of Boolean differential calculus [2]. The application of MVL differential calculus allows the development of fast decomposition algorithms for function intervals. The algorithms developed in this paper are a direct extension of the AND- and OR-decomposition algorithms, respectively, for Boolean functions [1, 3].

The paper is organized as follows. Section 2 shows basic properties of the MM-operators, defines differential operators and function intervals. In section 3 the bi-decomposition problem for function intervals is discussed. Properties of function intervals are shown, which are needed for the construction of efficient MM-decomposition algorithms. Results of the bi-decomposition algorithms are presented and compared to the results of another decomposer in section 4. Finally, ideas for further development of bi-decomposition systems are presented in section 5.

2 Basic properties of MVL functions

Throughout this paper, all variables are assumed to be multi-valued with variable cardinality, i.e. a variable a_i has the integer domain $[0, m_{ai} - 1]$ where m_{ai} is the *cardinality* of the variable a_i. For a set of variables $A = \{a_1, \ldots, a_n\}$, an *MVL function* $f(A)$ is a mapping from the Cartesian product of the domains of the variables $a_1 \ldots a_n$ onto the integers from the interval $[0, m_f - 1]$. The integer m_f is the *output cardinality* of the function $f(A)$. A *minterm* A_k is an assignment of constant values to the variables of A.

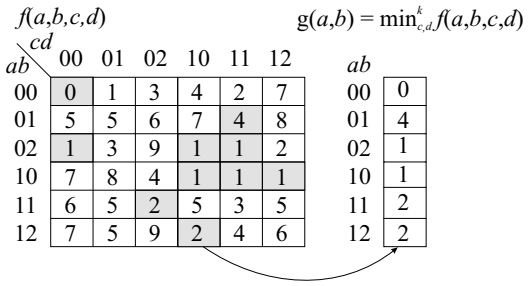

Figure 1. The k-times Minimum operator applied to an MVL function. The values of $g(a,b) = \min_{c,d}{}^k f(a,b,c,d)$ are indicated by shaded fields in the map of $f(a,b,c,d)$.

2.1 Derivation operators

Derivation operators have successfully been applied to decomposition of Boolean functions [1]. Below, two derivation operators for MVL functions, $\min^k$ and $\max^k$ are defined. The letter k distinguishes the k-times derivation operators from the vector derivation operators with the same name. A more complete discussion of differential calculus can be found in [13]. The MVL derivation operators defined below are direct generalizations of the corresponding Boolean operators [2].

Definition 1. The *k-times minimum* $\min_B{}^k f(A,B)$ of the function $f(A,B)$ over the set of variables B is the function

$$\min_B{}^k f(A,B) = \min_{\forall B_i} f(A,B_i) \qquad (1)$$

Similarly, the *k-times maximum* $\max_B{}^k f(A,B)$ of the function $f(A,B)$ over the set of variables B is the function

$$\max_B{}^k f(A,B) = \max_{\forall B_i} f(A,B_i) \qquad (2)$$

The example (Figure 1) shows the function $f(a,b,c,d)$ and its k-times minimum $g(a,b) = \min_{c,d}{}^k f(a,b,c,d)$ over the set of variables $\{c,d\}$. The result of the operator is the minimum value of each row of the map of $f(a,b,c,d)$.

Note, that the results of the k-times MM derivation operations over the set of variables B do not depend on the variables of B anymore. These variables are removed from the original function. This property will be used when applying the differential operators in decomposition tests. The following lemma relates the functions that depend on the sets of variables A and B to the functions that only depend on A.

Lemma 1. *Let $f(A,B)$ and $g(A)$ be two functions. There is $g(A) \leq f(A,B)$ iff $g(A) \leq \min_B{}^k f(A,B)$, and there is $g(A) \geq f(A,B)$ iff $g(A) \geq \max_B{}^k f(A,B)$.*

2.2 Function intervals

There are several generalizations of MVL functions, such as incompletely specified functions and MVL relations. Here, another generalization of MVL functions, the so called function intervals, are introduced because they naturally arise in the decomposition of MVL functions.

Definition 2. A *function interval* $F(X) = [f_l(X), f_u(X)]$ with $f_l(X) \leq f_u(X)$ is the set of MVL functions

$$F(X) = \{f(X) | f_l(X) \leq f(X) \leq f_u(X)\}. \qquad (3)$$

A function interval $F = [f_l(X), f_u(X)]$ can be visualized by a map similar to ISFs. The *lower bound* $f_l(X_i)$ and the *upper bound* $f_u(X_i)$ for the minterm X_i are shown as an interval $[f_l(X_i), f_u(X_i)]$ in the field X_i of the map.

Figure 2(a) shows the map of the function interval $F(a,b) = [f_1(a,b), f_4(a,b)]$, where the lower bound $f_1(a,b)$ and the upper bound $f_4(a,b)$ are displayed in Figure 2(b). The function interval $F(a,b)$ contains exactly the functions $F(a,b) = \{f_1(a,b), f_2(a,b), f_3(a,b), f_4(a,b)\}$ which are also displayed in Figure 2(b).

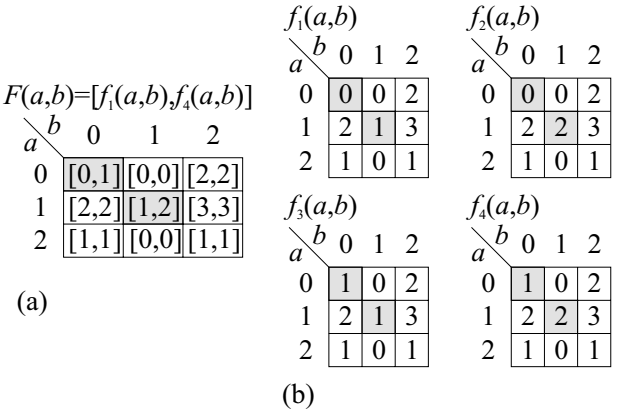

Figure 2. Function intervals. (a) Representation of the function interval $F(a,b) = [f_1(a,b), f_4(a,b)]$ as a map. (b) Enumeration of the member functions of the function interval $F(a,b) = \{f_1(a,b), f_2(a,b), f_3(a,b), f_4(a,b)\}$.

3 Max-decomposition by derivation operators

This section shows properties of function intervals that are needed for the construction of efficient bi-decomposition algorithms with respect to the MM-operators. The max-decomposition problem for function

intervals is defined in section 3.1. Tight lower and upper bounds on the decomposition functions are derived in section 3.2. Then, these bounds are applied to formulate a test condition for the max-decomposability of function intervals in section 3.3. The computation of the decomposition functions is developed in section 3.4.

3.1 The max-decomposition problem

Definition 3. A *bi-decomposition* of a function $f(A, B, C)$ with respect to the sets of variables A and B, and the operator π is a pair of functions $\langle g(A, C), h(B, C)\rangle$ so that

$$f(A, B, C) = \pi(g(A, C), h(B, C)) \tag{4}$$

where A, B, and C are three disjoint sets of variables. The functions $g(A, C)$ and $h(B, C)$ are called *decomposition functions*. The set C is called the *shared set*. A function $f(A, B, C)$ is called *decomposable* with respect to A and B if at least one decomposition exists.

Not all functions are decomposable. To increase the probablility that a bi-decomposition exists, MVL functions with some freedom for optimization, such as ISFs have been decomposed. This paper introduces bi-decomposition algorithms for function intervals.

Definition 4. A *bi-decomposition* of the function interval $F(A, B, C) = [f_l(A, B, C), f_u(A, B, C)]$ is a pair of functions $\langle g(A, C), h(A, C)\rangle$ so that

$$f_l(A, B, C) \leq \pi(g(A, C), h(B, C)) \leq f_u(A, B, C). \tag{5}$$

A function interval is called *bi-decomposable* if at least one bi-decomposition exists.

Now, bi-decomposition of function intervals with respect to the max-operator, or shorter *max-decomposition* is considered. From inequality (5) can be derived that a function interval $F(A, B, C) = [f_l(A, B, C), f_u(A, B, C)]$ is *max-decomposable* iff there exist functions $g(A, C)$ and $h(B, C)$ with

$$\max(g(A, C), h(B, C)) \geq f_l(A, B, C) \tag{6}$$
$$\max(g(A, C), h(B, C)) \leq f_u(A, B, C). \tag{7}$$

3.2 Bounds on the decomposition functions

Similar to ISFs, function intervals provide some freedom in the choice of the function $f(A, B, C) \in F(A, B, C)$ that is decomposed. Any function from the interval can be chosen and decomposed. For efficient multi-level decomposition, it is important to pass as much as possible of this freedom to the next level. Therefore, not only one pair of decomposition functions $\langle g(A, C), h(B, C)\rangle$ is computed, but

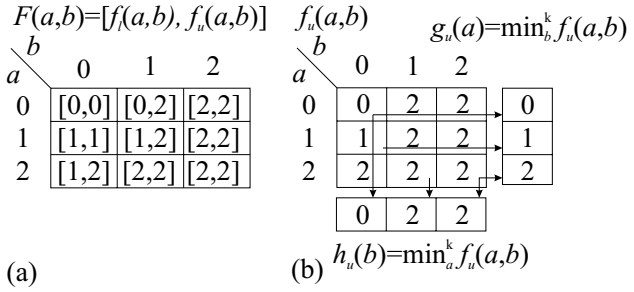

Figure 3. Max-decomposition of a function interval. (a) Function interval $F(a, b)$. (b) Upper bounds $g_u(a)$ and $h_u(b)$ on the max-decomposition functions of $F(a, b)$.

decomposition intervals $G(A, C) = [g_l(A, C), g_u(A, C)]$ and $H(B, C) = [h_l(B, C), h_u(B, C)]$ that contain as many functions as possible.

The development of bounds on the decomposition functions is illustrated by decomposition the function interval $F(a, b)$ shown in Figure 3(a). The function interval is to be max-decomposed with respect to the sets of variables $A = \{a\}$ and $B = \{b\}$. To simplify the example, the shared set C is empty, but the theorems will be formulated for the general case.

A max-decomposition $\langle g(A, C), h(B, C)\rangle$ of the function interval $F(A, B, C) = [f_l(A, B, C), f_u(A, B, C)]$ must satisfy inequality (7). Because of the inequality $g(A, C) \leq \max(g(A, C), h(B, C))$, it can be concluded that $g(A, C) \leq f_u(A, B, C)$.

In Figure 3(b), the values of function $g(a_i)$ must not exceed the values of $f_u(a_i, b)$. Therefore, an upper limit on the function values of $g(a)$ is the minimum value of each row of the map of $f_u(a, b)$.

With above arguments is shown that there are upper limits on the decomposition functions. The questions remains whether these limits are tight, i.e. exist decompositions that reach these limits? The lemma below gives a positive answer.

Lemma 2. *Let $F(A, B, C) = [f_l(A, B, C), f_u(A, B, C)]$ be a max-decomposable function interval and let the pair $\langle g(A, C), h(B, C)\rangle$ be a max-decomposition of the interval $F(A, B, C)$, then there is*

$$g(A, C) \leq g_u(A, C) \tag{8}$$
$$h(B, C) \leq h_u(B, C), \tag{9}$$

where the functions $g_u(A, C)$ and $h_u(B, C)$ are defined by

$$g_u(A, C) = \min_B{}^k f_u(A, B, C) \tag{10}$$
$$h_u(B, C) = \min_A{}^k f_u(A, B, C). \tag{11}$$

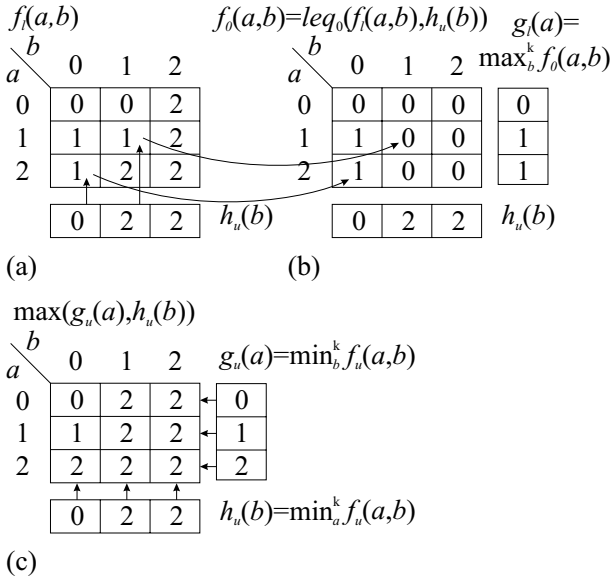

Figure 4. Computation of decomposition functions. (a) Lower bound $f_l(a,b)$ of the function interval $F(a,b)$. (b) Computation of the lower bound $g_l(a)$ of the decomposition set. (c) Test for max-decomposability by comparison of $\max(g_u(a),h_u(b))$ with $f_l(a,b)$.

Furthermore, the pairs of functions $\langle g_u(A,C),h(B,C)\rangle$ and $\langle g(A,C),h_u(B,C)\rangle$ are also max-decompositions of $F(A,B,C)$.

Lemma 2 has an important consequence. In general, there are many max-decomposition functions, say $g_1,g_2,\ldots,g_m$ and $h_1,h_2,\ldots,h_n$. However, not every pair of functions $\langle g_i,h_j\rangle$, $i=1\ldots m$, $j=1\ldots n$ is a max-decomposition. Now, the lemma shows that all pairs $\langle g_u,h_j\rangle$, $j=1\ldots n$ and $\langle g_i,h_u\rangle$, $i=1\ldots m$ are max-decompositions.

The calculation of a lower bound on the decomposition functions is complicated by the fact that the lower bound for the functions $g(A,C)$ depends on the second decomposition function $h(B,C)$. Consider decomposition of $F(a,b)$ from Figure 3(a) with respect to the function $h_u(b)$ shown below the map of $f_l(a,b)$ in Figure 4(a). For the minterm $a,b=1,1$ there is $h_u(b=1)\geq f_l(a,b=1,1)$ and $\max(g_l(a=1),h_u(b=1))\geq f_l(a,b=1,1)$ is satisfied independently of the value of $g_l(a=1)$. For the minterm $a,b=2,0$ there is $h_u(b=0)<f_l(a,b=2,0)$. Therefore, the minimum value of $g_l(a=2)$ should be $f_l(a,b=2,0)=1$. The restrictions on $g_l(a)$ for all minterms b_j are shown in the map of function $f_0(a,b)$ in Figure 4(b). The lower bound $g_l(a)$, after taking the maximum over the variable b, is shown to the right of the map.

The restrictions on the lower bound can be expressed with the help of a special operator function $leq_0(x,y)$ (if x is less than or equal to y, then 0) defined by

$$leq_0(x,y)=\begin{cases}0 & \text{for } x\leq y\\ x & \text{otherwise.}\end{cases} \tag{12}$$

Again, the question remains whether the lower bound is tight, which is positively answered by the following lemma.

Lemma 3. *Let $F(A,B,C)=[f_l(A,B,C),f_u(A,B,C)]$ be a max-decomposable function interval and let the pair $\langle g(A,C),h(B,C)\rangle$ be a max-decomposition of the function interval $F(A,B,C)$, then there is*

$$g(A,C)\geq g_l(A,C) \tag{13}$$

$$g_l(A,C)=\max_B{}^k leq_0(f_l(A,B,C),h(B,C)). \tag{14}$$

Furthermore, the pair of functions $\langle g_l(A,C),h(B,C)\rangle$ is also a max-decomposition of $F(A,B,C)$.

3.3 Max-decomposition test

With the results of the previous section, it is possible to derive a test condition for max-decomposability of function intervals from the upper bounds on the decomposition functions. Lemma 2 has shown tight upper bounds $g_u(A,C)$ and $h_u(B,C)$ on the decomposition functions. Therefore, a decomposition can only exist if these bounds satisfy (6). On the other hand if (6) is satisfied for the upper bounds, the functions $g_u(A,C)$ and $h_u(B,C)$ are a decomposition and the function interval is decomposable. The following theorem gives a formal argument based on this intuition.

Theorem 1. *A function interval $F(A,B,C)$ with the bounds $F(A,B,C)=[f_l(A,B,C),f_u(A,B,C)]$ is max-decomposable iff*

$$f_l(A,B,C)\leq\max(g_u(A,C),h_u(B,C)) \tag{15}$$

$$g_u(A,C)=\min_B{}^k f_u(A,B,C) \tag{16}$$

$$h_u(B,C)=\min_A{}^k f_u(A,B,C). \tag{17}$$

See Figure 4(c) for example. The functions $g_u(a)$ and $h_u(b)$ are displayed together with $\max(g_u(a),h_u(b))$. Comparison with $f_l(a,b)$ in Figure 4(a) shows that (15) is satisfied. Additional comparison with $f_u(a,b)$ from Figure 3(a) reveals that $\langle g_u(a),h_u(b)\rangle$ is indeed a max-decomposition of $[f_l(a,b),f_u(a,b)]$.

3.4 Computation of the decomposition functions

In multi-level decomposition, the test for decomposability is only the first step. Once, a decomposable function interval has been found, decomposition functions $g(A,C)$

and $h(B,C)$ must be computed. The problem is that there are many pairs $\langle g_i(A,C), h_j(B,C)\rangle$ of decomposition functions. In general, it is difficult to choose the best pair of functions, because the effects of this choice on next levels of decomposition are very complex. One solution is to postpone the choice of the decomposition functions by computing function intervals $G(A,C)$ and $H(B,C)$, where the choice of the actual function $g(A,C) \in G(A,C)$ is made later during subsequent decompositions [9]. This situation is very similar to decomposition of ISFs where an ISF is not decomposed into functions, but ISFs. The theorem below applies the results of Lemmas 2 and 3 and shows how to compute the interval $G(A,C)$ of decomposition functions.

Theorem 2. *Let the function interval* $F(A,B,C) = [f_l(A,B,C), f_u(A,B,C)]$ *be max-decomposable. Then, the function* $g(A,C)$ *is a decomposition function of* $F(A,B,C)$ *iff* $g(A,C) \in [g_0(A,C), g_u(A,C)]$, *where*

$$g_0(A,C) = \max_B{}^k leq_0(f_l(A,B,C), h_u(B,C)) \quad (18)$$

$$h_u(B,C) = \min_A{}^k f_u(A,B,C) \quad (19)$$

$$g_u(A,C) = \min_B{}^k f_u(A,B,C). \quad (20)$$

Similar theorems can be developed for min-decomposition by dual arguments.

4 Results of the decomposition system YADE

The results of Theorems 1 and 2 have been applied in a decomposition system called YADE (yet another decomposer). The input to the program is a function interval. YADE returns a netlist of MM-gates that represents a single function from the interval. YADE applies multi-valued decision diagrams (MDD) for the representation of the MVL functions. The MDDs are stored and manipulated by the MDD package [4]. The function intervals are represented by pairs of lower and upper bounds.

4.1 Supplementary algorithms

For a complete decomposition system, there are some additional problems to be solved that are only briefly discussed here.

Elimination of vacuous and inessential variables. It is possible that some or all functions of the given function interval do not depend on several variables. These variables are called inessential or vacuous, respectively [7]. They can be removed before decomposition. The removal algorithm used in YADE is a memorized search algorithm with back-tracking and a fixed timeout for the search.

Table 1. Decomposition of the UCI machine learning functions [12]. Columns: In - number of inputs, Out - number of outputs, G_Y - number of gates by YADE, DFC_Y - DFC by YADE, DFC_F - DFC by FREDMVL, T_Y - computation time by YADE on a 500 MHz Pentium.

$Name$	In	Out	G_Y	DFC_Y	DFC_F	T_Y
balance	4	1	268	2012	375	18 sec
breastc	9	1	95	634	7025	24 sec
flare1	10	3	154	932	1751	11 sec
flare2	10	3	300	8049	1413	37 sec
hayes	4	1	22	128	314	1 sec

Simplification of indecomposable functions. Some function intervals do not contain MM-decomposable functions. These intervals are simplified by separation. Separation simplifies a function interval by splitting it into a larger interval and a decomposable interval.

Selection of variables. First, all pairs of single variables are tested for the sets of variables A and B, respectively. If a decomposition is found, variables from the shared set C are successively moved to either A or B. The smaller of the two sets is tested first. The sequence of variables is the sequence of the input.

Reuse of functions. It is possible to simplify the decomposition process by reusing functions that have already been decomposed. YADE stores all decomposed functions in a database. Each function interval is checked if a realization is found in the database. If found, the interval is not decomposed, but its realization from the database will be reused.

4.2 Results

Several machine learning functions from the UCI database [12] have been decomposed by the YADE system and compared to the results of the decomposer FREDMVL [5], see Table 1. FREDMVL applies a generalization of Curtis decomposition to decompose ISFs into smaller blocks [8]. The cardinalities of the variables are in the range from 2 to 10. The number of gates includes MM-gates as well as the literal functions at the inputs of the decomposition network. The discrete function cardinality (DFC) of the decomposed networks has been computed to measure the complexity. For a function with one output, the DFC is the product of the cardinalities of all input variables. The DFC of a decomposition network is the sum of the DFCs of all gates.

Table 1 indicates that there are significant differences in the complexity of the circuits in both directions. Therefore, an optimized decomposer can be implemented by combining the two approaches. In general, YADE gives better results for functions that are well decomposable because bi-decomposition does not create intermediate input variables. For functions that are difficult to decompose, FREDMVL has the better result because functional decomposition is more flexible in the choice of the decomposition functions. Furthermore, functions can be left undecomposed if the decomposition has larger complexity than the original function. A direct comparison of the computation time is not possible because different hardware was used for FREDMVL and YADE (except for the function balance, YADE was faster than FREDMVL [5]).

5 Conclusion and further work

Derivation operators on MVL functions are well suited to formulate bi-decomposition algorithms for MM-operators. Function intervals are a natural generalization of ISFs that are preferred to describe the decomposition functions that arise from MM-decomposition. The algorithms have been implemented using MDDs as data structure. Several functions from machine learning applications have successfully been decomposed. For some functions, the complexity of the decomposition results obtained by bi-decomposition is significantly smaller than results obtained by functional decomposition.

A very innovative technique for manipulation of MVL relations are implicit methods, which use binary-encoded multi-valued decision diagrams (BEMDD) [7]. A BEMDD encodes MVL variables and MVL function values in several BDD variables. Current research activities indicates that the decomposition methods of this paper can be extended to BEMDDs. Using BEMDDs it is easily possible to decompose MVL relations. The implementation can be based on any available BDD package.

There are several possibilities to reduce the complexity of decompositions found by the YADE program. The separation of functions that are not decomposable is very important. Non-decomposable functions have great influence on the final number of MM-gates. Separation is the fist step in the decomposition of these function, and it seems that this step has great impact on the complexity of the final result.

The complexity of the decompositions and the computation time can be decreased by using some measures for the selection of variables of the decompositions.

Although the MM-operators are very simple, other operators, such as truncated sum and product for instance, would be desirable for the decomposition of machine learning functions. Efficient decomposition algorithms for these operators have yet to be developed.

List of abbreviations

BEMDD	binary-encoded multi-valued decision diagram
DFC	discrete function cardinality
ISF	incompletely specified function
MDD	multi-valued decision diagram
MM	min and max
MVL	multi-valued logic
YADE	yet another decomposer

References

[1] D. Bochmann, F. Dresig, and B. Steinbach. A new decomposition method for multilevel circuit design. In *European Conference on Design Automation*, pages 374–377, Amsterdam, 1991.

[2] D. Bochmann and C. Posthoff. *Binäre dynamische Systeme*. Oldenbourg Verlag, München, 1981.

[3] D. Bochmann and B. Steinbach. *Logikentwurf mit XBOOLE*. Verlag Technik, Berlin, 1991.

[4] C. Files. *MDD package*. Portland State University.

[5] C. Files, R. Drechsler, and M. Perkowski. Functional decomposition of mvl functions using multi-valued decision diagrams. In *27th IEEE International Symposium on Multiple-Valued Logic*, pages 27–32, Halifax, May 1997.

[6] J. Lou and J. Brzozowski. A generalization of shestakov's function decomposition method. In *29th IEEE International Symposium on Multiple-Valued Logic*, pages 66–71, Freiburg, May 1999.

[7] A. Mishchenko, C. Files, M. Perkowski, B. Steinbach, and C. Dorotska. Implicit algorithms for multi-valued input support minimization. In *4th International Workshop on Boolean Problems*, pages 9–20, Freiberg, September 2000.

[8] M. Perkowski. A new representation of strongly unspecified switching functions and its application to multi-level and/or/exor synthesis. In *Second Workshop on Applications of Reed-Muller Expansion in Circuit Design*, pages 143–151, Chiba City, August 1995.

[9] B. Steinbach, C. Lang, and M. Perkowski. Bi-decomposition of discrete function sets. In *4th International Workshop on Applications of the Reed-Muller Expansion*, pages 233–252, Victoria, B.C., 1999.

[10] B. Steinbach, M. Perkowski, and C. Lang. Bi-decomposition of multi-valued functions for circuit design and data mining applications. In *29th IEEE International Symposium on Multiple-Valued Logic*, pages 50–58, Freiburg, May 1999.

[11] N. Takagi and K. Nakashima. Some properties of discrete interval truth valued logic. In *30th IEEE International Symposium on Multiple-Valued Logic*, pages 101–106, Portland, May 2000.

[12] UCI, http://www.ics.uci.edu/˜mlearn/MLRepository.html. *UCI data base*.

[13] S. Yanushkevich. *Logic Differential Calculus in Multi-Valued Logic Design*. Habilitation thesis, Technical University of Szczecin, Szczecin, 1998.

[14] B. Zupan. *Machine Learning Based on Function Decomposition*. PhD thesis, University of Ljubljana, Ljubljana, 1997.