Portland State University

[PDXScholar](#)

Enhancements to Crisp Possibilistic Reconstructability Analysis

Anas Al-Rabadi
Portland State University

Martin Zwick
Portland State University, zwick@pdx.edu

Enhancements to Crisp Possibilistic Reconstructability Analysis

Anas N. Al-Rabadi[1] and Martin Zwick[2]
(1) ECE Department (2) Systems Science Ph.D. Program @Portland State University
[alrabadi@ece.pdx.edu] [1], [zwick@sysc.pdx.edu] [2]

KEYWORDS

Reconstructability Analysis, Ashenhurst-Curtis Decomposition, Boolean Functions, NPN-Classification, Log-Functionality Complexity Measure.

ABSTRACT

Modified Reconstructibility Analysis (MRA), a novel decomposition within the framework of set-theoretic (crisp possibilistic) Reconstructibility Analysis, is presented. It is shown that in some cases while 3-variable NPN-classified Boolean functions are not decomposable using *Conventional Reconstructibility Analysis (CRA)*, they are decomposable using *Modified Reconstructibility Analysis (MRA)*. Also, it is shown that whenever a decomposition of 3-variable NPN-classified Boolean functions exists in both MRA and CRA, MRA yields simpler or equal complexity decompositions. A comparison of the corresponding complexities for Ashenhurst-Curtis decompositions, and Modified Reconstructibility Analysis (MRA) is also presented. While both AC and MRA decompose some but not all NPN-classes, MRA decomposes more classes, and consequently more Boolean functions. MRA for many-valued functions is also presented, and algorithms using two different methods (intersection and union) are given. A many-valued case is presented where CRA fails to decompose but MRA decomposes.

1 INTRODUCTION

One general methodology for understanding a complex system is to decompose it into less complex sub-systems. Decomposition is used in many situations; for example, in *logic synthesis* (e.g., Al-Rabadi 2002; Ashenhurst 1953; Ashenhurst 1956; Ashenhurst 1959; Curtis 1963; Curtis 1962; Files 2000; Grygiel 2000; Muroga 1979) where the number of inputs to the gates is high and cannot be mapped to a standard library and in *machine learning* where data is noisy or incomplete (Files 2000, Grygiel 2000). The primary criteria for evaluating the quality of the decomposition process are the amount of *information (or the loss of information, i.e., error)* existing in the decomposed system and the *complexity* of this decomposed system. The *objective* is obvious: *decompose the complex system (data) into the least-complex most-informative*

(least-error) model. Simplicity is desired since, according to the Occam Razor principle, *the simpler the model is, the more powerful it is for generalization.* Least error is desired since one wants to retain as much information as possible in the decomposed system, when compared to the original data. Decomposition processes can be generally dichotomized into *lossless (no error) versus lossy decompositions.* In this paper, a comparison of two types of lossless decompositions are considered: the disjoint *Ashenhurst-Curtis decomposition,* and *set-theoretic (crisp possibilistic) Modified Reconstructibility Analysis (MRA) decomposition.*

The remainder of this paper is organized as follows: Section 2 presents necessary methodological background. CRA, MRA, and AC, complexity results are presented in Section 3. Many-valued MRA is presented in Section 4. Conclusions and future work are discussed in Section 5.

2 BINARY LOGIC FUNCTIONS CLASSIFICATION, COMPLEXITY MEASURES, AND DECOMPOSITIONS

This Section introduces the basic background of the NPN-classification of three-variable 2-valued logic functions, Ashenhurst-Curtis (AC), and Reconstructibility Analysis (RA) decomposition methods, and the complexity measures utilized here, to compare the efficiencies of AC and RA decompositions.

2.1 NPN-Classification of Binary Logic Functions

There exist many classification methods to cluster logic functions into families of functions (Muroga 1979). Two important operations that produce equivalence classes of logic functions are *negation* and *permutation* (Muroga 1979). Accordingly, the following classification types result:

1. *P-Equivalence class: a family of identical functions obtained by the operation of permutation of variables.*

2. *NP-Equivalence class: a family of identical functions obtained by the operations of negation or permutation of one or more variables.*

3. *NPN-Equivalence class: a family of identical functions obtained by the operations of negation or permutation of one or more variables, and also negation of function.*

NPN-Equivalence classification will be used in this work. Figure 1 lists 3-variable Boolean functions, for the non-degenerate classes (i.e., the classes depending on all three variables).

Class	Representative Function	Number of Functions
1	$F = x_1x_2 + x_2x_3 + x_1x_3$	8
2	$F = x_1 \oplus x_2 \oplus x_3$	2
3	$F = x_1 + x_2 + x_3$	16
4	$F = x_1(x_2 + x_3)$	48
5	$F = x_1 x_2' x_3 + x_1' x_2' x_3'$	8
6	$F = x_1' x_2 x_3 + x_1 x_2' + x_1 x_3$	24
7	$F = x_1(x_2 x_3 + x_2' x_3')$	24
8	$F = x_1 x_2 + x_2 x_3 + x_1' x_3$	24
9	$F = x_1' x_2 x_3 + x_1 x_2' x_3 + x_1 x_2 x_3'$	16
10	$F = x_1 x_2' x_3' + x_2 x_3$	48

Figure 1. NPN-Equivalence classes for non-degenerate Boolean functions of three binary variables (Muroga 1979) for a total of 218 Boolean functions, where (,) means negation of a variable, $\oplus$ is Boolean exclusive Sum-Of-Product, + is Boolean OR, and product is Boolean AND.

Example 1. The following steps produce the sets of all possible Boolean functions that are included in class #1 in Figure 1 for the representative function $F_0 = x_1x_2 + x_2x_3 + x_1x_3$.

(1) Negation of variables (N): $\{F_1 = x_1' x_2 + x_2 x_3 + x_1' x_3, F_2 = x_1 x_2' + x_2' x_3 + x_1 x_3, F_3 = x_1 x_2 + x_2 x_3' + x_1 x_3',$

$F_4 = x_1' x_2' + x_2' x_3 + x_1' x_3, F_5 = x_1' x_2 + x_2 x_3' + x_1' x_3', F_6 = x_1 x_2' + x_2' x_3' + x_1 x_3', F_7 = x_1' x_2' + x_2' x_3' + x_1' x_3'\}$

(2) Permutation of variables (P): does not produce a different function

(3) Negation of functions (N): $\{F_9 = x_1' x_2' + x_2' x_3' + x_1' x_3'\}$

F_7 and F_9 are the same, which gives 8 distinct functions.

2.2 Complexity Measures

Decomposability means complexity reduction. Many complexity measures exist for the purpose of evaluating the efficiency of the decomposition of complex systems into simpler sub-systems. Such complexity measures include: the *Decomposed Function Cardinality* (DFC) complexity measure (Abu-Mostafa 1988), and the *Log-Functionality (LF)* complexity measure (Grygiel 2000). In DFC, complexity is a *count* of the total number of possible functions realizable by the decomposed structure, while LF counts the number of *non-redundant* functions realizable by the decomposed structure. The complexity of the decomposed structures is always less or equal to the complexity of the original Look-Up-Table (LUT) that represents the mapping of the non-decomposed structure. That is, if a decomposed structure has higher complexity than the original structure, then the original structure is said to be non-decomposable. Although the DFC measure is easier and more familiar, LF is a better measure because it more properly deals with non-disjoint systems (Grygiel 2000). Consequently, the LF measure will be used in this paper. The DFC

and LF complexity measures are illustrated using Figure 2, which exemplifies AC decomposition, as follows:
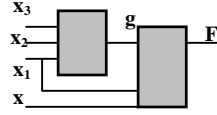

Figure 2. Generic non-disjoint decomposition.

In Figure 2, for the first block , the total number of possible functions for three 2-valued input variables is $_2 2^3$ = 256. Also, for the second block, the total number of possible functions is similarly 256. The total possible number of functions for the whole structure is equal to 256·256 = 65,536. The DFC measure is defined as:

$$DFC_j = O_j \cdot 2^{I_j} \tag{1}$$

$$C_{DFC} = \sum_j DFC_j \tag{2}$$

where O_j is the number of outputs to block j, I_j is the number of inputs to the same block, Equation (1) is the complexity for block j, and Equation (2) is the complexity for the total decomposed structure. For instance, the DFC for Figure 2 is: C_{DFC} = 1·2^3 + 1·2^3 = 16. It was shown in (Grygiel 2000) that, for Figure 2, the *Log-Functionality complexity measure (C_{LF})* for Boolean functions can be expressed as follows:

$$C_{LF} = \log_2(C_F) \tag{3}$$

where:

$$C_F = (C'_F)^{p_{X_3}} \quad , \quad C'_F = \sum_{i=0}^{p_{Y_1}-1} P(p_{Y_2}^{\frac{p_{X_2}}{p_{X_3}}}, p_{Y_1}-i) S(\frac{p_{X_1}}{p_{X_3}}, p_{Y_1}-i)$$

$$P(n,k) = \frac{n!}{(n-k)!}, \quad S(n,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i \binom{k}{i}(k-i)^n, \quad \binom{k}{i} = \frac{k!}{i!(k-i)!}.$$

$$X_1 = \{x_1, x_2, x_3\}, \quad X_2 = \{x_1, x_4\}, \quad X_3 = X_1 \cap X_2 = \{x_1\}, \quad Y_1 = g, \quad Y_2 = F$$

$$p_{X1} = \prod_{x_i \in X_1}|x_i|, \quad p_{X2} = \prod_{x_i \in X_2}|x_i|, \quad p_{X3} = \prod_{x_i \in X_3}|x_i|, \quad p_{Y1} = \prod_{y_i \in Y1}|y_i|, \quad p_{Y2} = \prod_{y_i \in Y2}|y_i|$$

where X_1 is the set of input variables to the first block, X_2 is the set of input variables to the second block, X_3 is the set of overlapping variables between sets X_1 and X_2, P_{Xi} is the product of cardinalities of the input variables in set X_i, Y_1 is the output of first block, Y_2 is the output of second block, and P_{Yi} is the product of cardinalities of output variables in set Y_i. For example, the LF for Figure 2 is:

$$X_1 = \{x_1, x_2, x_3\}, X_2 = \{x_1, x_4\}, X_3 = X_1 \cap X_2 = \{x_1\}$$
$$\therefore p_{X1} = 2 \cdot 2 \cdot 2 = 8, p_{X2} = 2 \cdot 2 = 4, p_{X3} = 2,$$
$$p_{Y1} = 2, p_{Y2} = 2,$$
$$C_F' = \sum_{i=0}^{1} P(2^2, 2-i) S(4, 2-i) = 88$$
$$\therefore C_F = 7,744 \Rightarrow C_{LF} = \log_2(7,744) = 12.92.$$

Figure 2 shows a four input function, where the variable sets for the first and second blocks are not disjoint. In this paper we are concerned only with 3-input functions, and in this case the AC decomposition results in a structure shown in Figure 3. Note that the variable sets for the two blocks with outputs g and F are necessarily disjoint, because if the two blocks shared one input variable, F would have three inputs and the decomposed structure would be more complex than the original non-decomposed 3-input function.

Example 2.
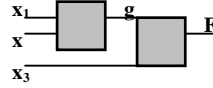
Figure 3. A decomposed structure.

The Log-Functionality complexity measure of the structure in Figure 3 is obtained as follows:

Each sub-block in Figure 3 has a total of $2^{2^2} = 16$ possible Boolean functions. Figure 4 illustrates all of the possible 16 two-variable Boolean functions per sub-block in Figure 3.
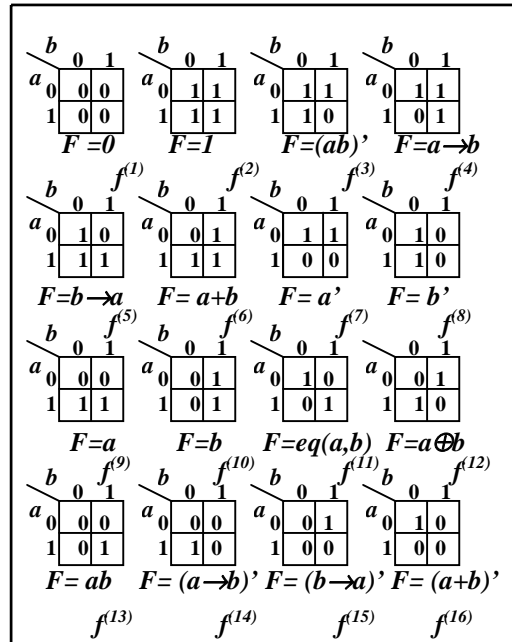
Figure 4. Maps of all 16 possible Boolean functions of two variables.

By allowing g and F in Figure 3 to take on all possible maps from Figure 4, one obtains the following count of total non-repeated (non-redundant) 3-variable functions, as follows: $C_F = 88 \Rightarrow C_{LF} = 6.5$. This answer agrees with the result of Equation (3) (Grygiel 2000).

Example 3. RA produces decompositions for 3-variable functions that resemble the structures shown in Figure 5.

(a)   *(b)*
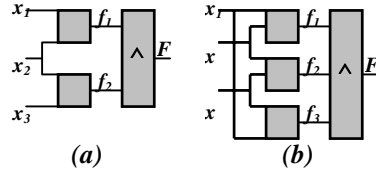
Figure 5. Some RA decomposed structures.

The Log-Functionality complexity measure for the structures in Figure 5, is obtained as follows: Figure 6 represents a tree that generates all possible functions for the structures 5a and 5b, respectively (Al-Rabadi 2002; Al-Rabadi et al. 2002). (Superscripts of functions denote the specific edge between two nodes in the tree).
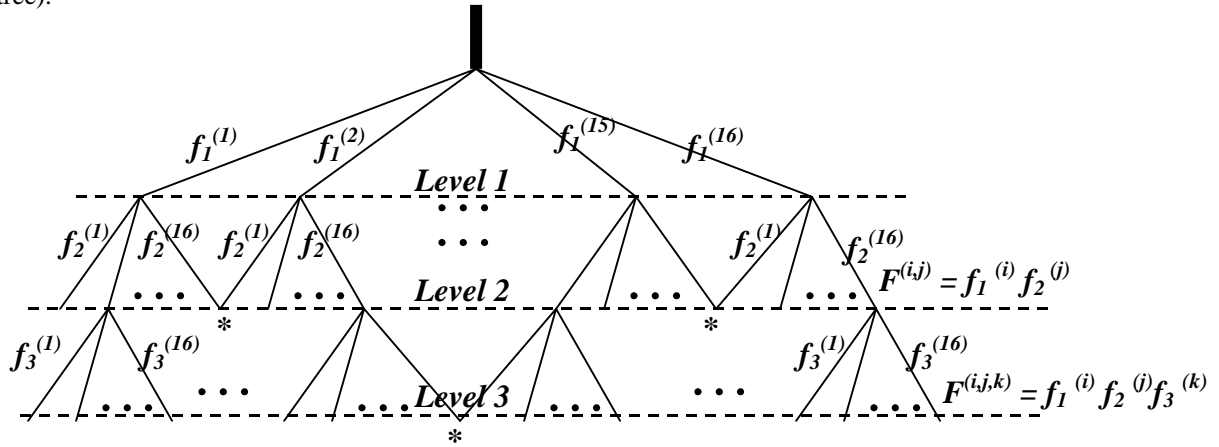
Figure 6. All possible combinations of sub-functions $f_1^{(i)}$, $f_2^{(j)}$, and $f_3^{(k)}$ in Figures 5a and 5b, respectively. *Log-functionality complexity measure* represents the count of all possible *non-redundant* functions, that is all *different* $F^{(i,j)}$ for Figure 5a, and all *different* $F^{(i,j,k)}$ for Figure 5b. *Where two nodes of the tree are superposed (*), they are counted only once.* At every node, there are 16 possible 2-variable Boolean functions.

Utilizing this methodology of removing redundant functions, one obtains the following results for Log-Functionality: for Figure 5a, the total number of irredundant sub-functions is $C_F = 100 \Rightarrow \therefore C_{LF} = log_2 (100) = 6.6$, and for Figure 5b, the total number of irredundant sub-functions is $C_F = 152 \Rightarrow \therefore C_{LF} = log_2 (152) = 7.2$. (In later tables, C_{LF} values of 4.3 and 6.5 are also reported, for functions $F = x_1 + x_2 + x_3$ and $F = x_1(x_2 + x_3)$, respectively.) The following example illustrates the use of Figure 6 to eliminate the

redundant terms in the process of computing the log-functionality complexity measure.

Example 4. Utilizing Figure 5a, if one choose the following maps from Figure 4: $f_1 = f^{(13)}$ and $f_2 = f^{(6)}$ then the function $F^{(13,6)} = f^{(13)}f^{(6)}$ will produce the same result as $F^{(13,11)} = f^{(13)}f^{(11)}$ since $(x_1 \cdot x_2)(x_1+x_2) = (x_1 \cdot x_2)(x_1 \equiv x_2)$, and consequently the two paths will lead to the same node in Level 2 in Figure 6. Also, we can demonstrate the same calculation for Figure 5b as follows: let $f_1 = f^{(8)}(b)$, $f_2 = f^{(2)}$, and

$f_3 = f^{(8)}(c)$ then the function $F^{(8,2,8)} = f^{(8)}(b)f^{(2)}f^{(8)}(c)$ will produce the same result as the function

$F^{(5,3,7)} = f^{(5)}(b,c)f^{(3)}(a,b)f^{(7)}(c)$ where $\{f_1 = f^{(5)}(b,c), f_2 = f^{(3)}(a,b), f_3 = f^{(7)}(c)\}$ and consequently the two paths will lead to the same node in Level 3 in Figure 6.

2.3 Ashenhurst-Curtis Decomposition

Ashenhurst-Curtis (AC) decomposition (Al-Rabadi 2002; Ashenhurst 1953, Ashenhurst 1956, Ashenhurst 1959, Curtis 1962, Curtis 1963, Files 2000, Grygiel 2000) is one of the major techniques for the decomposition of functions commonly used in the field of logic synthesis. The main idea of AC decomposition is to decompose logic functions into simpler logic blocks using the compression of the number of cofactors in the corresponding representation. This compression is achieved through exploiting the logical compatibility (i.e., redundancy) of cofactors (i.e., column multiplicity). As a result of AC decomposition (as a result of column compression), intermediate constructs (latent variables) are created, and learning is achieved as a result of these variables (Files 2000, Grygiel 2000). A general algorithm of the AC decomposition utilizing Karnaugh map (K-map) representation (Muroga 1979), for instance, is as follows:

(1) Partition the input set of variables into free set and bound set, and label all the different columns.

(2) Decompose the bound set and create a new K-map for the decomposed bound set (utilizing minimum graph coloring, maximum clique, or some other algorithm to combine similar columns into a single column). Each cell in the new K-map represents a labeled column in the original K-map.

(3) Encode the labels in the cells of the new K-map using minimum number of intermediate binary variables. These intermediate variables are shown as g and h in Example 5 (Figure 7). Express the intermediate variables as functions of the bound set variables.

(4) Produce the decomposed structure, i.e., a K-map specifying the function (F) in terms of the intermediate variables and the free set variables.

In general, steps (1) and (3) determine the optimality of the AC decomposition (i.e., whether the resulting decomposed blocks are of minimal complexity or not).

Example 5. For the following logic function $F = x_2x_3 + x_1x_3 + x_1x_2$, let the sub-set of variables $\{x_2, x_3\}$ be the *Bound Set*, and the sub-set of variables $\{x_1\}$ be the *Free Set*. The following is the disjoint AC decomposition of F (where $\{-\}$ means don't care):
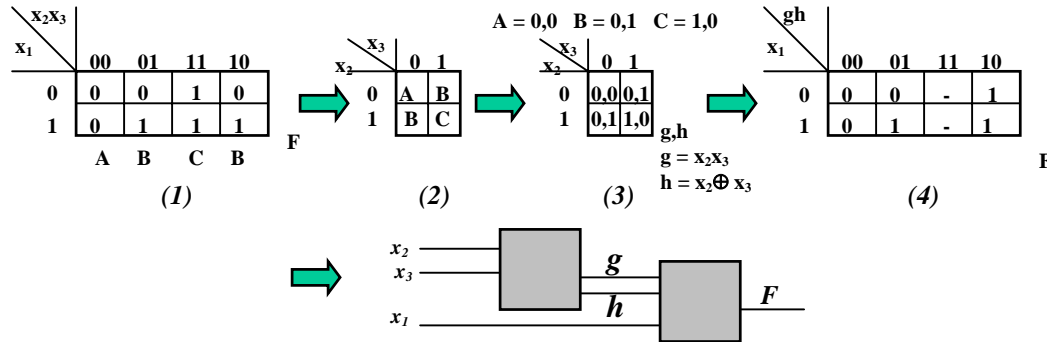
Figure 7. AC decomposition. Steps (1)-(4) are discussed in the text.

In Example 5, the first block of the decomposed structure has two outputs (intermediate variables g and h). The DFC measure of the decomposed structure is $= 2 \cdot 2^2 + 1 \cdot 2^3 = 16$, while the DFC of the original LUT is $= 1 \cdot 2^3 = 8$. This shows the inadequacy of DFC as a measure of complexity because the decomposition produces a more complex structure than the non-decomposed LUT. By contrast, LF for the decomposed structure in Figure 7 is 8, which does not exceed the complexity of the LUT. Thus, for AC decomposition of Boolean functions with 3-variables, if the first block of the decomposed structure has two outputs, then the decomposed structure is at least as complex as the LUT, and consequently, for the purpose of this paper, the decomposition is rejected. For other NPN functions AC decomposition produces only one output in the first block. These decompositions are not rejected, and are listed in Figure 12.

2.4 Reconstructability Analysis: Conventional RA versus Modified RA for the Binary Case

Reconstructability Analysis (RA) is a decomposition technique for qualitative data (Klir 1996; Klir 1985; Klir and Wierman 1998; Krippendorff 1986; Zwick 2001). RA data is typically either a set theoretic relation or mapping or it is a probability or frequency distribution. The former case is the domain of "set-theoretic" RA or more precisely crisp possibilistic RA. The latter is the domain of "information-theoretic"

RA, or more precisely probabilistic RA. The RA framework can apply to other types of data (e.g., fuzzy data) via generalized information theory (Klir and Wierman 1998).

In this paper, we are concerned only with crisp possibilistic RA. RA decomposition can also be lossless or lossy. In this paper, we are concerned only with lossless decomposition, i.e., with decomposition which produces no error. This paper introduces an innovation in set-theoretic RA, which we call "modified" RA (or MRA) (Al-Rabadi 2001; Al-Rabadi 2002; Al-Rabadi et al. 2002) as opposed to the conventional set-theoretic RA (or CRA). This innovation is illustrated by Example 6.

Example 6. For the logic function: $F = x_1 x_2 + x_1 x_3$

Figure 8 illustrates decomposed structures using both CRA and MRA decompositions, respectively. In Figure 8, while CRA decomposes for all values of Boolean functions, MRA decomposes for an arbitrarily chosen value of the Boolean functions (e.g., for value "1"). ***The completely specified Boolean function can be retrieved if one knows the MRA decomposition for the Boolean function being equal either to "1" or to "0"*** (Al-Rabadi 2001; Al-Rabadi 2002; Al-Rabadi et al. 2002).
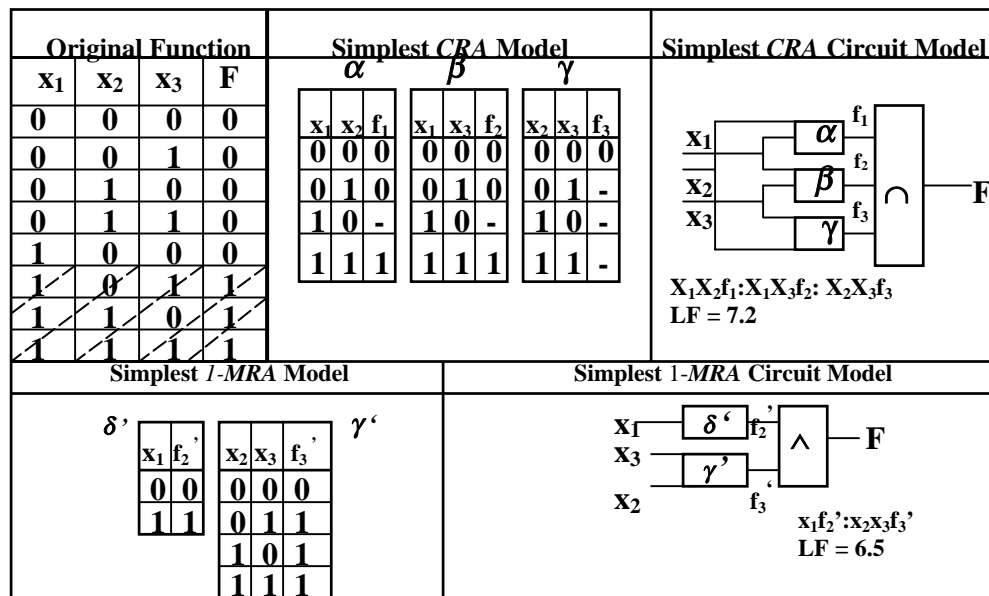
Figure 8. Conventional versus Modified RA decompositions for the Boolean function: $F = x_1 x_2 + x_1 x_3$.

CRA decomposition (Zwick 1995; Zwick and Shu 1995) is illustrated in the upper half of Figure 8, while MRA decomposition is illustrated in the lower half of the figure. MRA decomposition yields much simpler logic circuit than the corresponding CRA decomposition, while retaining complete information about the decomposed logic function. For CRA in Figure 8, the calculated function for model =

$x_1x_2f_1:x_1x_3f_2:x_2x_3f_3$ (i.e., α:β:γ) is defined as follows: $\mathbf{x_1x_2x_3F}_{\mathbf{model}} \equiv (\mathbf{x_1x_2f_1} \otimes \mathbf{x_3}) \cap (\mathbf{x_1x_3f_2} \otimes \mathbf{x_2}) \cap (\mathbf{x_2x_3f_3} \otimes \mathbf{x_1})$.

For lossless CRA decomposition, this equals the original function $\mathbf{x_1x_2x_3F}$ that is shown at the top left of Figure 8. (For *lossy* CRA $\mathbf{x_1x_2x_3F}_{\mathbf{model}}$ would not be equivalent to $\mathbf{x_1x_2x_3F}$). The CRA model can be interpreted by the circuit shown at the top right of Figure 8, where different projections of F are labeled f_1, f_2, and f_3. MRA simplifies the decomposition problem by focusing, in the original function F, on the three shaded tuples ("cubes") for which F=1. The procedure used to obtain the 1-MRA in Figure 8 is as follows (Al-Rabadi 2001; Al-Rabadi 2002; Al-Rabadi et al. 2002):

(1) Decompose the Boolean function of value "1" into the simplest lossless CRA decomposition.

(2) For a particular model, get the projections.

(3) Assign value "1" (for 1-MRA) to the tuples in the resulted projection. Add all tuples that are missing in the projections which will have the functional value "0".

(4) Perform the AND operation for 1-MRA in the output block to obtain the total functionality.

Steps (2-)-(4) are illstrated as follows:

x_1	x_2	x_3	F
1	0	1	1
1	1	0	1
1	1	1	1

→

x_1	x_2	x_3
1	0	1
	1	0
	1	1

→

x_1	f_2'	x_2	x_3	f_3'
0	0	0	0	0
1	1	0	1	1
		1	0	1
		1	1	1
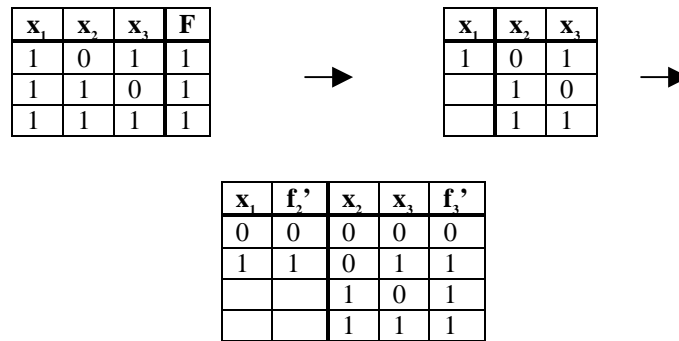
The output function in step (4) is the logical AND of the two subfunctions, i.e., $F = f_2'(x_1) \wedge f_3'(x_2,x_3)$. Set-theoretically, this can be represented as $F = (x_1 \otimes (1 \cup x_1') \otimes 0) \cap (x_2x_3 \otimes (1 \cup (x_2x_3)') \otimes 0)$. From Figure 8, one observes that MRA possess two main advantages over CRA for the decomposition of Boolean functions (which will be further demonstrated in Figure 10): (1) The resulting decomposed structures from MRA are less complex that the corresponding decomposed structures from CRA, and (2) The resulting decomposed structures from MRA are directly realizable in Boolean-based circuits, while the resulting decomposed structures from CRA are not realizable in Boolean-based circuits, but in ternary-valued logic circuits, and thus the resulting logic circuits from MRA are directly implementable using the current technologies. The idea of 0-MRA versus 1-MRA is illustrated in Example 7.

Example 7. For the logic function: $F = x_1x_2 + x_1x_3$

Figure 9 illustrates the simplest model using both 1-MRA and 0-MRA.

In this example, *The completely specified Boolean function can be retrieved if one knows the MRA decomposition for the Boolean function being equal either to "1" (that is 1-MRA) or to "0" (that is 0-MRA).* The procedure used to obtain the 0-MRA in Figure 9 is as follows (Al-Rabadi 2002):

(1) Decompose the Boolean function of value "0" into the simplest lossless CRA decomposition.

(2) For a particular model, get the projections.

(3) Assign value "0" (for 0-MRA) to the tuples in the resulted projection. Add all tuples that are missing in the projections which will have the functional value "1".

(4) Perform the OR operation for 0-MRA in the output block to obtain the total functionality.

Steps (2-)-(4) are illustrated as follows:

x_1	x_2	x_3	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0

→

x_1	x_2	x_1	x_3
0	0	0	0
0	1	0	1
1	0	1	0

→

x_1	x_2	$f_1^{''}$	x_1	x_3	$f_2^{''}$
0	0	0	0	0	0
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	1	1	1
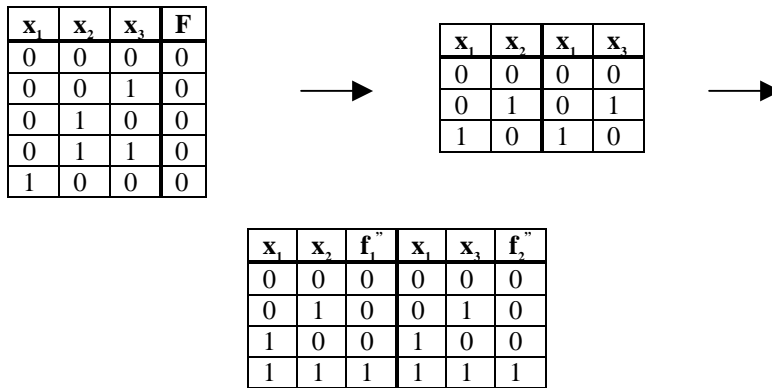
The output function in step (4) is the logical OR of the two sub-functions as follows:

$F = f_1^{''}(x_1,x_2) \vee f_2^{''}(x_1,x_3)$.

Simplest *0-MRA* Model	Simplest *0-MRA* Circuit Model
$\alpha^{''}$ $\beta^{''}$ $\begin{array}{ccc} x_1 & x_2 & f_1^{''} \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{array}$ $\begin{array}{ccc} x_1 & x_3 & f_2^{''} \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{array}$	 $x_1x_2f_1^{''}$: $x_1x_3f_2^{''}$ LF = 6.6
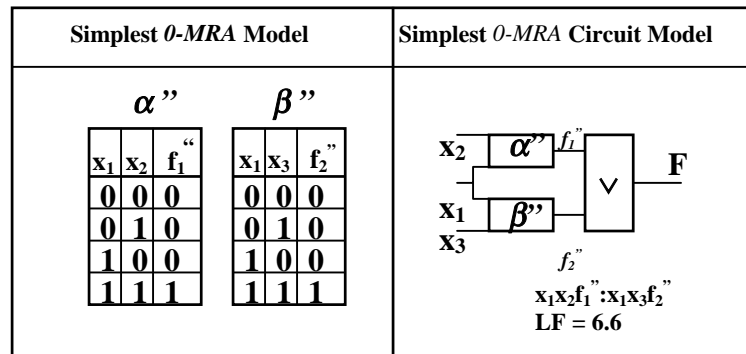
Figure 9. 0-MRA versus 1-MRA decompositions for the Boolean function $F = x_1x_2 + x_1x_3$.

As can be observed from Figure 9, 1-MRA produces less complex decomposed structure than 0-MRA. The 0-MRA (LF = 6.6) decomposition should be compared to the 1-MRA decomposition (LF = 6.5) which is shown in Figure 8.

3 RESULTS: COMPARING MRA TO OTHER DECOMPOSITIONS

The following Sections compare the complexities of the decomposed structures using MRA with complexities of the decomposed structures using CRA (Section 3.1) and AC (Section 3.2).

3.1 Complexity of MRA versus CRA Decompositions

Figure 10 compares MRA and CRA decompositions of all NPN-classes of 3-variable Boolean functions.

NPN-Representative Function	MRA	Simplest Modified RA model (0-MRA or 1-MRA)	Simplest Conventional RA model	C (LUT)	C_{LF} (CRA)	C_{LF} (MRA)
Class 1 (8) $F = x_1x_2 + x_2x_3 + x_1x_3$	1	[x₁ x₂ f₁: 000 / 011 / 101 / 111] [x₂ x₃ f₂: 000 / 011 / 101 / 111] [x₁ x₃ f₃: 000 / 011 / 101 / 111]	[x₁ x₂ f₁: 000 / 01- / 10- / 111] [x₂ x₃ f₂: 000 / 01- / 10- / 111] [x₁ x₃ f₃: 000 / 01- / 10- / 111]	8	7.2	7.2
Class 2 (2) $F = x_1 \oplus x_2 \oplus x_3$	*	non-decomposable	non-decomposable	8	8	8
Class 3 (16) $F = x_1 + x_2 + x_3$	0	[x₁ f₁: 00 / 11] [x₂ f₂: 00 / 11] [x₃ f₃: 00 / 11]	non-decomposable	8	8	4.3
Class 4 (48) $F = x_1(x_2 + x_3)$	1	[x₁ f₁: 00 / 11] [x₂ x₃ f₂: 000 / 011 / 101 / 111]	[x₁ x₂ f₁: 000 / 010 / 10- / 111] [x₂ x₃ f₂: 000 / 010 / 10- / 111] [x₁ x₃ f₃: 000 / 01- / 10- / 11-]	8	7.2	6.5
Class 5 (8) $F = x_1x_2x_3 + x_1'x_2'x_3'$	1	[x₁ x₂ f₁: 001 / 010 / 100 / 111] [x₁ x₃ f₂: 001 / 010 / 100 / 111]	non-decomposable	8	8	6.6
Class 6 (24) $F = x_1'x_2x_3 + x_1x_2' + x_1x_3'$	*	non-decomposable	non-decomposable	8	8	8
Class 7 (24) $F = x_1(x_2x_3 + x_2'x_3')$	1	[x₂ x₃ f₁: 001 / 010 / 100 / 111] [x₁ f₂: 00 / 11]	non-decomposable	8	8	6.5
Class 8 (24) $F = x_1x_2 + x_2x_3 + x_1'x_3$	1	[x₁ x₂ f₁: 001 / 011 / 100 / 111] [x₁ x₃ f₂: 000 / 011 / 101 / 111]	[x₁ x₂ f₁: 00- / 01- / 100 / 111] [x₁ x₃ f₂: 000 / 011 / 10- / 11-]	8	6.6	6.6
Class 9 (16) $F = x_1'x_2x_3 + x_1x_2'x_3 +$ $x_1x_2x_3'$	*	non-decomposable	non-decomposable	8	8	8
Class 10 (48) $F = x_1x_2'x_3' + x_2x_3$	1	[x₁ x₃ f₁: 000 / 011 / 101 / 111] [x₂ x₃ f₂: 001 / 010 / 100 / 111]	non-decomposable	8	8	6.6

Figure 10. Conventional RA (CRA) versus Modified RA (MRA) for the decomposition of all NPN-classes of 3-variable Boolean functions (Compare the right-most two columns), where in the second column from left "0" means 0-MRA, "1" means 1-MRA, and * means 0-MRA or 1-MRA.

Figure 10 shows that in five NPN classes (classes 1, 2, 6, 8, 9) totaling 74 functions MRA and CRA give equivalent complexity decompositions, but in the remaining five classes (classes 3, 4, 5, 7, 10) totaling 144 functions MRA is superior in complexity reduction. This is summarized in Figure 11.
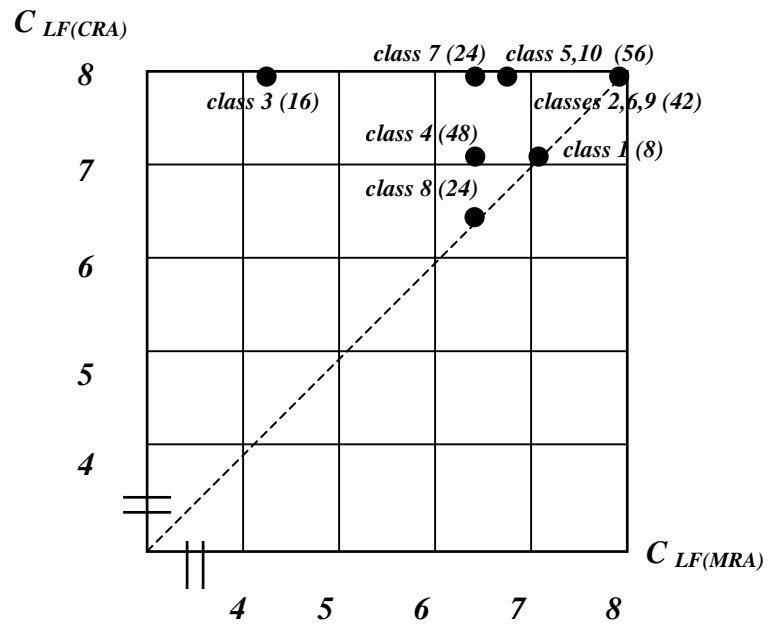
Figure 11. Comparison of the Log-Functionality complexity measure between modified RA (MRA) and conventional RA (CRA) of 3-variable NPN-classified Boolean functions.

3.2 Complexity of MRA verus AC Decompositions

Utilizing the methods described above, one obtains the following results in Figure 12 for the decomposition of 3-variable NPN-classified Boolean functions (Figure 1) using MRA and AC decompositions.

14

NPN-Representative Function	Simplest Modified RA model (0-MRA or 1-MRA)	Simplest AC circuit	Simplest Modified RA circuit	DFC (SOP)	C_{data} (LUT)	C_{LF} (MRA)	C_{LF} (AC)
Class 1 (8) $F = x_1x_2 + x_2x_3 + x_1x_3$	$x_1\,x_2\,f_1$ / $x_2\,x_3\,f_2$ / $x_1\,x_3\,f_3$ truth tables	non-decomposable	RA circuit diagram	20	8	7.2	8
Class 2 (2) $F = x_1 \oplus x_2 \oplus x_3$	non-decomposable	$g = x_2 \oplus x_3,\ F = x_1 \oplus g$	-	8	8	8	6.5
Class 3 (16) $F = x_1 + x_2 + x_3$	$x_1\,f_1$ / $x_2\,f_2$ / $x_3\,f_3$ truth tables	$g = x_2 + x_3,\ F = x_1 + g$	RA circuit diagram	8	8	4.3	6.5
Class 4 (48) $F = x_1(x_2 + x_3)$	$x_1\,f_1$ / $x_2\,x_3\,f_2$ truth tables	$g = x_2 + x_3,\ F = x_1 g$	RA circuit diagram	12	8	6.5	6.5
Class 5 (8) $F = x_1x_2x_3 + x_1'x_2'x_3'$	$x_1\,x_2\,f_1$ / $x_1\,x_3\,f_2$ truth tables	non-decomposable	RA circuit diagram	20	8	6.6	8
Class 6 (24) $F = x_1'x_2x_3 + x_1x_2' + x_1x_3'$	non-decomposable	$g = x_2 x_3,\ F = x_1 \oplus g$	-	24	8	8	6.5
Class 7 (24) $F = x_1(x_2x_3 + x_2'x_3')$	$x_2\,x_3\,f_1$ / $x_1\,f_2$ truth tables	$g = x_2 \oplus x_3,\ F = x_1 g'$	RA circuit diagram	20	8	6.5	6.5
Class 8 (24) $F = x_1x_2 + x_2x_3 + x_1'x_3$	$x_1\,x_2\,f_1$ / $x_1\,x_3\,f_2$ truth tables	non-decomposable	RA circuit diagram	20	8	6.6	8
Class 9 (16) $F = x_1'x_2x_3 + x_1x_2'x_3' + x_1x_2x_3'$	non-decomposable	non-decomposable	-	32	8	8	8
Class 10 (48) $F = x_1x_2'x_3' + x_2x_3$	$x_1\,x_3\,f_1$ / $x_2\,x_3\,f_2$ truth tables	non-decomposable	RA circuit diagram	16	8	6.6	8

Figure 12. AC versus MRA for the decomposition of all NPN-classes of 3-variable Boolean functions. (Compare the right-most two columns.) Note that *all* AC decompositions have the same structure, while MRA decompositions have *four* different circuit topologies.

Figure 12 shows that in three NPN classes (4, 7, 9) totaling 88 functions MRA and AC decompositions give equivalent complexity decompositions. In two remaining classes (2, 6), totaling 26 functions, AC decomposition is superior, but in five classes (1, 3, 5, 8, 10), totaling 104 functions, MRA is superior. This is summarized in Figure 13.

$C_{LF(CRA)}$ Classes 5,9,10 (72) Classes 2,3,6,7 (66) Class 4 (48) Class 1 (8) Class 8 (24) $C_{LF(AC)}$ (a)

$C_{LF(MRA)}$ Classes 2,6 (26) Class 9 (16) Class 1 (8) Classes 4,7 (72) Classes 5,8,10 (80) Class 3 (16) $C_{LF(AC)}$ (b)
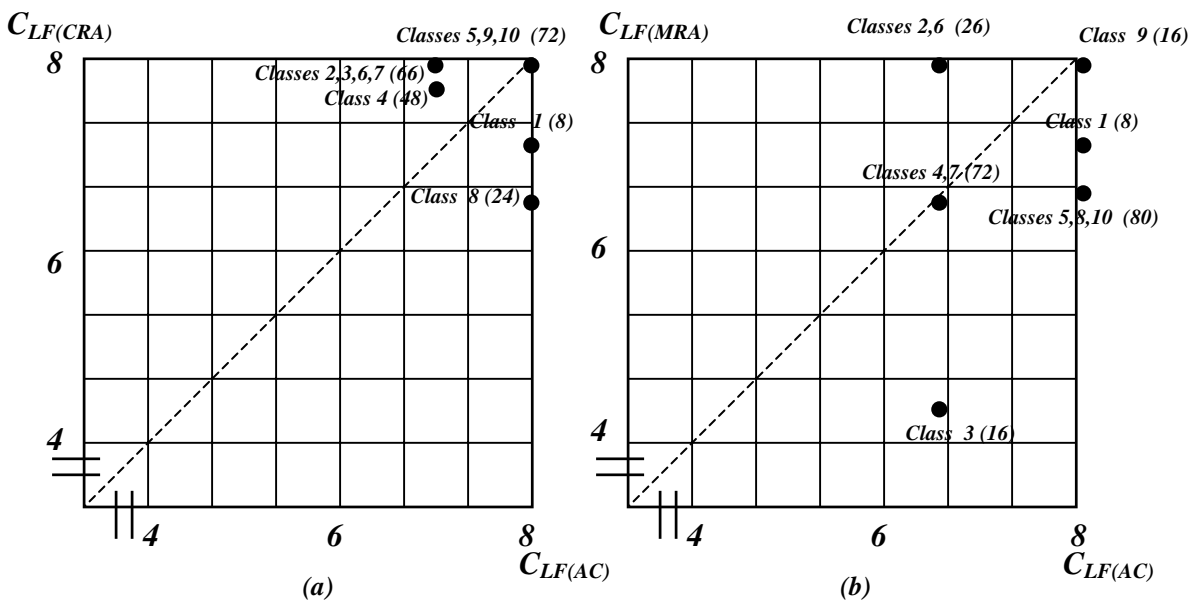
Figure 13. Comparison of the Log-Functionality complexity measure between CRA versus AC decompositions (a), and MRA versus AC decompositions (b), of 3-variable NPN-classified Boolean functions.

We can also summarize these results, by comparing decomposability versus non-decomposability for the various approaches as shown in Figure 14.

(a) MRA

	ND	D
ND	**1** (9: 16)	**4** (1,5,8,10: 88)
AC D	**2** (2,6: 26)	**3** (3,4,7: 88)

(b) AC

	ND	D
ND	**3** (5,9,10: 72)	**4** (2,3,6,7: 66)
CRA D	**2** (1,8: 32)	**1** (4: 48)

(c) MRA

	ND	D
ND	**3** (2,6,9: 42)	**4** (3,5,7,10: 96)
CRA D	**0** (0)	**3** (1,4,8: 80)

Figure 14. Comparison of the Decomposability (**D**) versus Non-Decomposability (**ND**) for (a) AC versus MRA, (b) CRA versus AC, and (c) CRA versus MRA, respectively. The number of classes are indicated, and in parantheses also the number of functions.

Fom Figure 14, one concludes that for NPN-classified 3-variable Boolean functions, MRA is superior to AC (88 versus 26), AC is superior to CRA (66 versus 32), and MRA is superior to CRA (96 versus 0).

4 MANY-VALUED MRA

This section presents MRA for many-valued functions and relations.

4.1 General Approach

Data are in general many-valued. Consequently, if MRA can decompose relations between many-valued variables it can have practical applications in machine learning and data mining. Many-valued MRA (Al-Rabadi 2001; Al-Rabadi 2002; Al-Rabadi and Zwick 2002) can be implemented with two equivalent algorithms: intersection-based and union-based. Both algorithms begin with the same two steps: (1) partition the many-valued truth table into sub-tables, each contain only single function value (e.g., $T = T_0 \cup T_1 \cup T_2$ for the corresponding output values O_0, O_1, and O_2 respectively), and (2) perform CRA on all sub-tables and obtain every M_j decomposition of T_j. Figure 15 illustrates the general pre-processing procedure for the two many-valued MRA algorithms, which will be explained in more detail below.

Original 3-valued table

| 0 |
| 1 |
| 2 |

Step (1): Separate one-valued tables

| 0 | 1 | 2 |

Step (2): CRA decompositions of all one-valued tables

Step (3): Application of MRA algorithm

Intersection
Algorithm

Union
Algorithm
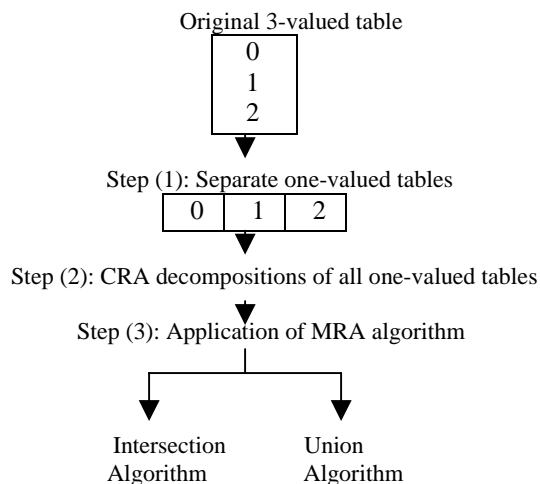
Figure 15. Steps for many-valued MRA.

For an "n"-valued completely specified function one needs (n-1) values to define the function. We thus do all n decompositions and use for our MRA model the (n-1) simplest of these. For example, obtain the simplest lossless MRA decomposition for value "0" of the function (denoted as the 0-MRA decomposition), for value "1" (1-MRA decomposition), and for value "2" (2-MRA decomposition). By selecting the simplest two models from these 0-MRA, 1-MRA, and 2-MRA decompositions, one can generate the complete function.

In the intersection method, first the CRA decompositions (M_j) are expanded to include the full set of variable and function values, and these "expanded" decompositions are then intersected to yield the original table.

In the union method the reconstructed function (T^*) is the union of all the sub-table decompositions,

$$T^* = \bigcup_{j=0}^{n-1} M_j \otimes O_j$$, where $\otimes$ is the set-theoretic Cartesian product. The union procedure can also be done

with (n-1) decompositions.

4.2 Complete Examples

The following are two examples which illustrate many-valued Modified Reconstructability Analysis of 3-valued functions. In the first example MRA can decompose the function for only two values, and one has no choice but to use both in the MRA model. In the second example, the function is decomposable for all three of its values, and the two simplest decompositions are chosen to define the model.

In discussing the second example, we show that this approach is generalizable to set-theoretic relations, in addition to mappings.

Example 8. We will generate the MRA decomposition for the ternary function specified by the following ternary map:

x_1x_2 \ x_3	0	1	2
00	0	0	0
01	1	1	0
02	1	1	1
10	0	0	2
11	0	0	2
12	1	1	1
20	0	2	0
21	1	1	0
22	2	2	0

F

The following is the intersection algorithm for many-valued MRA for the ternary function in Example 8.

Step 1: decompose the ternary chart of the function into three separate tables each for a single function value. This will produce the following three sub-tables.

Value "0"	Value "1"	Value "2"
000	010	102
001	011	112
002	020	201
012	021	220
100	022	221
101	120	
110	121	
111	122	
200	210	
202	211	
212		
222		
D0	**D1**	**D2**

Step 2: Perform CRA for each sub-table.

Step 2a: The simplest error-free 0-MRA decomposition is the original "0"-subtable itself since it is not decomposable.

Step 2b: 1-MRA decomposition of D1 is as follows:

Table 1		Table 2	
X_1	X_2 :	X_2	X_3
0	1	1	0
0	2	1	1
1	2	2	0
2	1	2	1
		2	2
D11		**D12**	

Step 2c: The 2-MRA decomposition of D2 is as follows:

Table 3		Table 4	
X_1	X_3 :	X_2	X_3
1	2	0	2
2	1	1	2
2	0	0	1
		2	0
		2	1
D21		**D22**	

THE INTERSECTION ALGORITHM

Step 3.1: Select the two simplest error-free decomposed models; these are 1-MRA and 2-MRA decompositions. MRA thus gives the decomposition model of D11:D12:D21:D22 from which the original function can be reconstructed as follows.

Step 3.2: Note that, for Tables 1 and 2, the MRA decomposition is for the value "1" of the logic function. Therefore, the existence of the tuples in the decomposed model implies that the function has value "1" for those tuples, and the non-existence of the tuples in the decomposed model implies that the function does not have value "1" but "0" or "2" for the non-appearing tuples. This is shown in Tables 1' and 2', respectively. Similarly, for Tables 3 and 4, for the value "2". The existence of the tuples in the decomposed model implies that the function has value "2" for those tuples, and the non-existence of the tuples in the decomposed model implies that the function does not have value "2" but "0" or "1" for the non-appearing tuples. This is shown in Tables 3' and 4', respectively.

Table 1'	Table 2'		Table 3'	Table 4'
$X_1\ X_2\ F_1$:	$X_2\ X_3\ F_2$		$X_1\ X_3\ F_3$:	$X_2\ X_3\ F_4$
0 0 0,2	0 0 0,2		0 0 0,1	0 0 0,1
0 1 1,0,2	0 1 0,2		0 1 0,1	0 1 2,0,1
0 2 1,0,2	0 2 0,2		0 2 0,1	0 2 2,0,1
1 0 0,2	1 0 1,0,2		1 0 0,1	1 0 0,1
1 1 0,2	1 1 1,0,2		1 1 0,1	1 1 0,1
1 2 1,0,2	1 2 0,2		1 2 2,0,1	1 2 2,0,1
2 0 0,2	2 0 1,0,2		2 0 2,0,1	2 0 2,0,1
2 1 1,0,2	2 1 1,0,2		2 1 2,0,1	2 1 2,0,1
2 2 0,2	2 2 1,0,2		2 2 0,1	2 2 0,1

In Tables 1' and 2' (i.e., the decomposition for value "1" of the function), the existence of value "1" (of sub-relations F_1 and F_2) means that the value "1" appeared in the original non-decomposed function for the corresponding tuples that appear in each table, but ***does not imply*** that the values "0" or "2" (of sub-relations F_1 and F_2) did not exist in the original non-decomposed function for the same tuples. Therefore "0" and "2" are added to "1" as allowed values. In the remaining tuples, however, only "0" and "2" are allowed since the value "1" did not occur. Similarly, in Tables 3' and 4', the existence of the value "2" (of sub-relations F_3 and F_4) means that the value "2" appeared in the original non-decomposed function for the corresponding tuples that appear in each table, but ***does not imply*** that values "0" or "1" did not exist in the original non-decomposed function for the same tuples. Therefore "0" and "1" are added to "2" as allowed values. In the remaining tuples, however, only "0" and "1" are allowed since the value "2" did not occur. Set-theoretically, obtaining tables 1', 2', 3', and 4' from tables 1, 2, 3, and 4 is described as follows:

Table 1': (D11⊗(0,1,2))∪(D11′⊗(0,2))
Table 2': (D12⊗(0,1,2))∪(D12′⊗(0,2))
Table 3': (D21⊗(0,1,2))∪(D21′⊗(0,1))
Table 4': (D22⊗(0,1,2))∪(D22′⊗(0,1))

where ′ here means complement of a set.

Step 3.3: Tables 1', 2', 3', and 4' are used to obtain the block diagram in Figure 16, where the following set-theoretic Equations govern the outputs of the levels in the circuit shown in the figure:

$F = F5 \cap F6$
$F5 = F1 \cap F2$
$F6 = F3 \cap F4$

where F1 is given by Table 1', F2 by Table 2', F3 by Table 3', and F4 by Table 4', respectively.

Figure 16. The decomposed structure resulting from the many-valued MRA decomposition.

The intermediate sub-functions, F5 and F6 are shown in the following maps, respectively.

x_1 \ x_2x_3	00	01	02	10	11	12	20	21	22
0	0,2	0,2	0,2	1	1	0,2	1	1	1
1	0,2	0,2	0,2	0,2	0,2	0,2	1	1	1
2	0,2	0,2	0,2	1	1	0,2	0,2	0,2	0,2

$$F_5 = F_1 \cap F_2$$

x_1 \ x_2x_3	00	01	02	10	11	12	20	21	22
0	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
1	0,1	0,1	2	0,1	0,1	2	0,1	0,1	0,1
2	0,1	2	0,1	0,1	0,1	0,1	2	2	0,1

$$F_6 = F_3 \cap F_4$$

Note that in Figure 16 the intersection blocks in the second level and the intersection block at the third (output) level, are general and do not depend on the function being decomposed. Only the tables at the first level depend upon this function.

THE UNION ALGORITHM

Steps 1 and 2 are the same as in the intersection algorithm.

Step 3.1: Using the decomposition model D11:D12:D21:D22 obtain D1 and D2 by standard CRA as follows:

D1 = (D11⊗x3)∩(D12⊗x1)
D2 = (D21⊗x2)∩(D22⊗x1)
D0 = (D1∪D2)′

where D1 is the decomposition for function value "1", D2 for function value "2", and x1, x2, and x3 ∈ {0,1,2}.

Step 3.2: Perform the set-theoretic operations to obtain the total function from the decomposed sub-functions.

x1x2x3F = (D1⊗1)∪(D2⊗2)∪((D1∪D2)′⊗(1∪2)′)
        = (D1⊗1)∪(D2⊗2)∪((D1∪D2)′⊗0)

Alternatively, one can use all three decompositions:

x1x2x3F = (D0⊗0)∪(D1⊗1)∪(D2⊗2)

The function value of (x1,x2,x3) is determined by the block diagram of Figure 17, where G performs the following operation:

F = 0 if (x1x2x3) ∈ D0
F = 1 if (x1x2x3) ∈ D1
F = 2 if (x1x2x3) ∈ D2
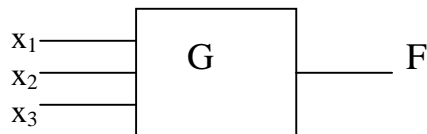
Figure 17. Block diagram for the union algorithm of MRA of Example 8.

Note that the logic function in Example 8 is non-decomposable using CRA but decomposable using MRA. We now consider an example where CRA does decompose, and also where MRA decomposes for all three values.

Example 9. Let us generate the MRA decomposition for the ternary function specified by the following ternary map:

X₁X₂ \ X₃	0	1	2
00	0	0	0
01	1	1	1
02	1	1	1
10	0	0	2
11	0	0	2
12	1	1	1
20	0	2	0
21	1	1	1
22	2	2	0

Utilizing the intersection-based algorithm, one obtains the following results for MRA for the ternary function in Example 9.

Step 1: decompose the ternary chart of the function into three separate tables each for a single function value. This will produce the following three sub-tables.

Value "0"	Value "1"	Value "2"
000	010	102
001	011	112
002	012	201
100	020	220
101	021	221
110	022	
111	120	
200	121	
202	122	
222	210	
	211	
	212	
D0	**D1**	**D2**

Step 2: Perform CRA for each sub-table.

Step 2a: The 0-MRA decomposition of D0 is as follows:

Table 1	Table 2	Table 3
X_1X_2 :	X_2X_3 :	X_1X_3
0 0	0 0	0 0
1 0	0 1	0 1
1 1	0 2	0 2
2 0	1 0	1 0
2 2	1 1	1 1
	2 2	2 0
		2 2
D01	**D02**	**D03**

Step 2b: The 1-MRA decomposition of D1 is as follows:

Table 4		Table 5
X_1	X_2	:	X_3

X_1	X_2	: X_3
0	1	0
0	2	1
1	2	2
2	1	
D11		**D12**

Step 2c: The 2-MRA decomposition of D2 is as follows:

Table 6   Table 7

X_1	X_3	: X_2	X_3
1	2	0	2
2	1	1	2
2	0	0	1
		2	0
		2	1
D21		**D22**	

THE INTERSECTION ALGORITHM

Step 3.1: Select the two simplest decomposed models, namely the 1-MRA and 2-MRA decompositions.

These are at a lower level in the lattice of structures than 0-MRA.

Step 3.2: Analogously to Example 8, one obtains the following expanded tables:

Table 4'   Table 5'   Table 6'   Table 7'

X_1 X_2 F_1	: X_3 F_2	X_1 X_3 F_3	: X_2 X_3 F_4
0 0 0,2	0 1,0,2	0 0 0,1	0 0 0,1
0 1 1,0,2	1 1,0,2	0 1 0,1	0 1 2,0,1
0 2 1,0,2	2 1,0,2	0 2 0,1	0 2 2,0,1
1 0 0,2		1 0 0,1	1 0 0,1
1 1 0,2		1 1 0,1	1 1 0,1
1 2 1,0,2		1 2 2,0,1	1 2 2,0,1
2 0 0,2		2 0 2,0,1	2 0 2,0,1
2 1 1,0,2		2 1 2,0,1	2 1 2,0,1
2 2 0,2		2 2 0,1	2 2 0,1

Set-theoretically, obtaining tables 4', 5', 6', and 7' from tables 4, 5, 6, and 7 is described as follows:

Table 4': (D11⊗(0,1,2))∪(D11'⊗(0,2))
Table 5': (D12⊗(0,1,2))∪(D12'⊗(0,2))
Table 6': (D21⊗(0,1,2))∪(D21'⊗(0,1))
Table 7': (D22⊗(0,1,2))∪(D22'⊗(0,1))

Step 3.3: Tables 4', 5', 6', and 7' are used to obtain the block diagram in Figure 18, where the following

set-theoretic Equations govern the outputs of the levels in the circuit shown in the figure:

F = F5 ∩ F6
F5 = F1 ∩ F2
F6 = F3 ∩ F4

where F1 is given by Table 4', F2 by Table 5', F3 by Table 6', and F4 by Table 7', respectively.
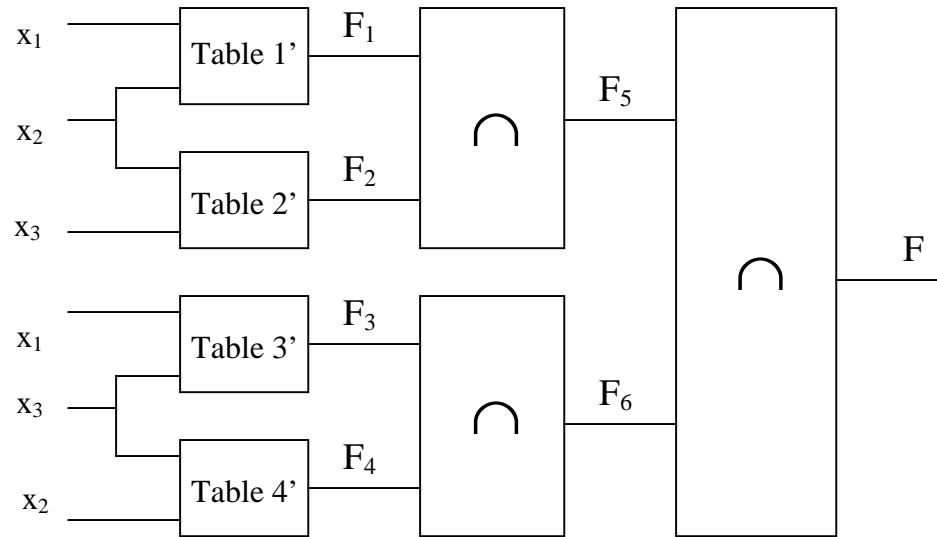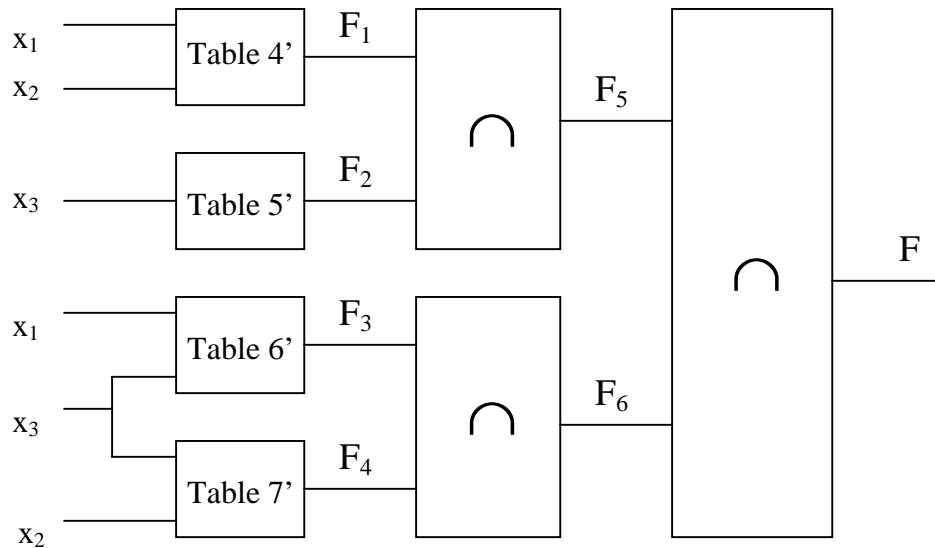
Figure 18. The decomposed structure resulting from the many-valued MRA decomposition.

The intermediate sub-functions, F5 and F6 are shown in the following maps, respectively.

x_1 \ x_2x_3	00	01	02	10	11	12	20	21	22
0	0,2	0,2	0,2	1	1	1	1	1	1
1	0,2	0,2	0,2	0,2	0,2	0,2	1	1	1
2	0,2	0,2	0,2	1	1	1	0,2	0,2	0,2

$$F_5 = F_1 \cap F_2$$

x_1 \ x_2x_3	00	01	02	10	11	12	20	21	22
0	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
1	0,1	0,1	2	0,1	0,1	2	0,1	0,1	0,1
2	0,1	2	0,1	0,1	0,1	0,1	2	2	0,1

$$F_6 = F_3 \cap F_4$$

THE UNION ALGORITHM

Steps 1 and 2 are the same as in the intersection algorithm.

Step 3.1: Using the decomposition model D01:D02:D11:D12:D21:D22 obtain D0, D1, and D2 by standard

methods as follows:

D0 = (D01⊗x3)∩(D02⊗x1)∩(D03⊗x2)
D1 = (D11⊗x3)∩(D12⊗x1x2)
D2 = (D21⊗x2)∩(D22⊗x1)

where D0 is the decomposition for function value "0", D1 is for function value "1", D2 for function value

"2", and x1, x2, and x3 ∈ {0,1,2}.

Step 3.2: Perform the set-theoretic operations to obtain the total function from the decomposed sub-functions. This can be done using only two of the three decompositions, as in Step (3.2) of the union algorithm in Example 8, or alternatively, one can use all three decompositions as follows:

x1x2x3F = (D0⊗0)∪(D1⊗1)∪(D2⊗2)

The function value of (x1,x2,x3) is determined by the block diagram of Figure 19, where G performs the following operation:

F = 0 if (x1x2x3) ∈ D0
F = 1 if (x1x2x3) ∈ D1
F = 2 if (x1x2x3) ∈ D2
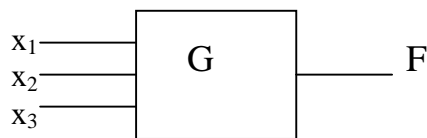
Figure 19. Block diagram for the union algorithm of MRA of Example 9.

The logic function in Example 9 is decomposable using CRA with the lossless CRA model $x_1x_2{:}x_2x_3{:}x_1x_3$. Consequently, unlike the previous example, both many-valued MRA and CRA decompose losslessly. Since both CRA and MRA decompose this function, we would like to be able to compare the complexities of the two decompositions. The complexity measure reported in (Al-Rabadi et al. 2002) could be used, but needs to be extended to many-valued functions.

From the previous discussion, it follows that the extension of many-valued MRA from functions to relations is trivial. One just performs the union algorithm using all n decompositions, e.g., for three values (D0⊗0)∪(D1⊗1)∪(D2⊗2).

5 CONCLUSION

A novel RA-based decomposition is introduced; *the Modified Reconstructability Analysis (MRA)*. It is shown that in 4 out of 10 NPN classes while 3-variable NPN-classified Boolean functions are not decomposable using the *Conventional Reconstructibility Analysis (CRA) decomposition*, they are decomposable using the *Modified Reconstructibility Analysis (MRA) decomposition.* Also, it is shown that whenever a decomposition of 3-variable NPN-classified Boolean functions exists in both MRA and CRA, MRA yields simpler or equal complexity decomposition. While the disjoint AC decomposition and MRA decompose some but not all NPN-classes, MRA decomposes more classes and consequently more Boolean functions than AC. The many-valued MRA decomposition is also presented. Since data are often many-

valued, future work will apply many-valued MRA to real-life data for machine learning, data mining, and data analysis. Future work will also include the investigation of the MRA decomposition of fuzzy functions. The use of gates other than the logical AND, and OR gates (e.g., XOR, NAND) at the final stage of RA-based decompositions to reduce the complexities of the decomposed structures will also be investigated.

6 REFERENCES

Abu-Mostafa, Y. 1988. *Complexity in Information Theory,* Springer-Verlag, New York.

Al-Rabadi, A. N. 2001. "A Novel Reconstructability Analysis For the Decomposition of Boolean Functions". Technical Report #2001/005, Electrical and Computer Engineering Department, Portland State University, Portland, Oregon, 1st July 2001.

Al-Rabadi, A. N. 2002. *Novel Methods for Reversible Logic Synthesis and their Application to Quantum Computing*. Ph.D. Dissertation, Electrical and Computer Engineering Department, Portland State University, Portland, Oregon.

Al-Rabadi, A. N., M. Zwick, and M. Perkowski 2002. "A Comparison of Enhanced Reconstructability Analysis and Ashenhurst-Curtis Decomposition of Boolean Functions". Book of Abstracts of the 12th international World Organization for Systems and Cybernetics (WOSC) Congress and the 4th International Institute for General Systems Studies (IIGSS) workshop, Pittsburgh, Pennsylvania, p. 12, 24-26th March 2002.

Al-Rabadi, A. N. and M. Zwick, 2002. "Modified Reconstructability Analysis for Many-Valued Logic Functions". Book of Abstracts of the WOSC/IIGSS'2002, Pittsburgh, Pennsylvania, p. 90, 24-26th March 2002.

Al-Rabadi, A. N. and M. Zwick, 2002. "Reversible Modified Reconstructability Analysis of Boolean Circuits and its Quantum Computation". Book of Abstracts of the WOSC/IIGSS'2002, Pittsburgh, Pennsylvania, p. 90, 24-26th March 2002.

Ashenhurst, R. L. 1953. "The Decomposition of Switching Functions". *Bell Laboratories' Report*, Vol. 1, pp. II-1-II-37.

Ashenhurst, R. L. 1956. "The Decomposition of Switching Functions". *Bell Laboratories' Report*, Vol. 16, pp. III-1-III-72.

Ashenhurst, R. L. 1959. "The Decomposition of Switching Functions". In: *International Symposium on the Theory of Switching Functions*, pp. 74-116.

Conant, R. 1981. "Set-Theoretic Structural Modeling". *International Journal of General Systems*: 7: 93-107.

Curtis, H. 1963. "Generalized Tree Circuit". *ACM,* pp. 484-496.

Curtis, H. 1963. "Generalized Tree Circuit-The Basic Building Block of an Extended Decomposition Theory". *ACM,* Vol. 10, pp. 562-581.

Curtis, H. A. 1962. *A New Approach to the Design of Switching Circuits*, Princeton, Van Nostrand, NJ.

Files, C. M. 2000. *A New Functional Decomposition Method as Applied to Machine Learning and VLSI Layout,* Ph.D. Dissertation, Electrical and Computer Engineering Department, Portland State University, Portland, Oregon.

Grygiel, S. 2000. *Decomposition of Relations as a new Approach to Constructive Induction in Machine Learning and Data Mining.* Ph.D. Dissertation, Electrical and Computer Engineering Department, Portland State University, Portland, Oregon.

Klir, G. 1985. *Architecture of Systems Problem Solving.* Plenum Press, New York.

Klir, G., editor, 1996. "Reconstructability Analysis Bibliography". *International Journal of General Systems*, 24:225- 229.

Klir, G. and M. J. Wierman, 1998. *Uncertainty-Based Information: Variables of Generalized Information Theory,* Physica-Verlag, New York.

Krippendorff, K. 1986. *Information Theory: Structural Models for Qualitative Data.* Sage Publications, Inc.

Muroga, S. 1979. *Logic Design and Switching Theory,* Wiley, New York.

Shannon, C. E. and W. Weaver 1949. *A Mathematical Theory of Communication,* University of Illinois Press.

Zwick, M. 1995. "Control Uniqueness in Reconstructibility Analysis". *International Journal of General Systems*, 23(2).

Zwick, M. and H. Shu, 1995. "Set-Theoretic Reconstructability of Elementary Cellular Automata". *Advances in System Science and Application, Special Issue I*, PP.31-36.

Zwick, M. 2001. Wholes and Parts in General Systems Methodology. In: *The Character Concept in Evolutionary Biology*, edited by G. Wagner, Academic Press.

Anas N. Al-Rabadi is currently a post-doctoral research fellow in the Systems Science Ph.D. Program at Portland State University. He received his Ph.D. in Electrical and Computer Engineering from Portland State University, Portland, Oregon in 2002 in the field of Advanced Logic Synthesis and Quantum Computing. He received his M.S. in Electrical and Computer Engineering from Portland State University in 1998 in the area of Power Electronics Systems Design and Feedback Control Systems Design. His current research areas include logic synthesis, reversible logic, quantum logic, multiple-valued logic, reconstructability analysis, signal processing, image processing, logic testing, and two-dimensional and three-dimensional regular structures.

Martin Zwick is currently Professor of Systems Science at Portland State University, Portland, Oregon. He received his Ph.D. in Biophysics from MIT in 1968, did post-doctoral work in the Department of Biochemistry of Stanford University, and was Assistant Professor in the Department of Biophysics & Theoretical Biology at the University of Chicago. His research in this period was in mathematical crystallography and macromolecular structure. In the 1970's his interests shifted to systems theory, methodology, and philosophy, and in 1976 he took his present position in the Systems Science Ph.D. Program at PSU. During the years 1984-1989, he was Director of the Program. His current research is primarily in three areas: (1) information and set-theoretic modeling (synchronic and time-series analysis of nominal or nominalized data); (2) "artificial life" (evolutionary simulations, genetic algorithm optimization, chaotic & nonchaotic dynamics in cellular automata); (3) systems philosophy (the metaphysics of "problems"). He also continues research in mathematical crystallography using systems methodologies.