Efficient Decomposition of Large Fuzzy Functions and Relations

Paul Burkey +, Marek Perkowski

+ Portland State University, Dept. Electrical Engineering, Portland, Oregon 97207, paburkey@ee.pdx.edu, Dept. Electr.l Engn. and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), 373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Korea, mperkows@ee.kaist.ac.kr

Abstract
This paper presents a new approach to decomposition of fuzzy functions. A tutorial background on fuzzy logic representations is first given to emphasize next the simplicity and generality of this new approach. Ashenhurst-like decomposition of fuzzy functions was discussed in [3] but it was not suitable for programming and was not programmed. In our approach, fuzzy functions are converted to multiple-valued functions and decomposed using an mv decomposer. Then the decomposed multiple-valued functions are converted back to fuzzy functions. This approach allows for Curtis-like decompositions with arbitrary number of intermediate fuzzy variables, that have been not presented for fuzzy functions by the previous authors. Extension of the method to fuzzy relations is also shown. The new approach is suitable for Machine Learning.

1. INTRODUCTION

Decomposition of a function is a process of creating an equivalent composition of other, simpler functions. For example, if x and y are sets of variables and $F(x, y) = H(G(x), y)$, then the term to the right is a composition of functions G and H that is equivalent to the function F. Thus, the complexity of F is reduced by representing function F in terms of functions G and H. The formulation of functional decomposition is very simple, but it is a very complex problem to solve when we deal with large data and want to find composing functions of the smallest total complexity. One problem is in determining how to group the input variables x and y for functions G and H. This process of selecting the input variables to G and H is called *variable partitioning* or *variable grouping*. The input variables going to G are called the bound set and those going to H are called the *free set*. The need for the introduction of fuzzy functions and multiple-valued functions is to extend the domain of binary functions. The world can not always be conveniently represented in binary terms so the concepts of fuzzy-valued, multiple-valued, and continuous-valued functions have been introduced. They find many applications other than circuit design, primarily Artificial Intelligence (AI), Machine Learning (ML), Fault Diagnosis, industrial control, Data Mining, Robotics, Knowledge Discovery from Data Bases (KDD), Multi-Objective Optimization, and many others. Binary functions have only two values, either 0 or 1, while a multiple-valued function can have many values. In fuzzy functions, the values are continuous in the range from 0 to 1. Decomposing fuzzy logic functions is a difficult problem because fuzzy logic is non-disjoint.

The definition, operations, identities and differences between the fuzzy logic and binary logic will be explained in the sections 2 and 3. Fuzzy maps and S-maps are then introduced. Next the steps to perform fuzzy logic decomposition using fuzzy maps as in the Kandel and Francioni method [3] will be briefly mentioned and some difficulties pointed out (section 4). Then the new approach based on converting a fuzzy function to a multiple-valued function [1, 2] and decomposing the multiple-valued function will be explained (sections 5 and 6). We developed previously several decomposing programs and made use of them to decompose large multiple-valued functions and relations [12 - 24]. These programs allow to deal with hundreds of variables, tens of thousands of terms, and solve efficiently difficult real-life problems from ML and KDD. The presented approach converts a multi-output fuzzy function to a multi-output three-valued function to be given as an input data to one of our decomposers. Finally, the method of converting the multiple-valued functions back to a fuzzy function will be explained in section 6 in order to prove that the network is a correct decomposition of the initial function. Section 8 presents extension of this method to fuzzy relations. Experimental results are in section 9, and section 10 concludes the paper.

1.1. Background on Fuzzy Logic

A fuzzy set, defined as A, is a subset of the universe f discourse U, where A is characterized by a membership function μ_A (x). The membership function μ_A (x) is associated with each point in U and is the "grade of membership" in A. The membership function μ_A (x) is assumed to range in the interval [0, 1], 0 corresponding to non-membership and 1 corresponding to full membership. The ordered pairs form the set {(x, μ_A (x))} to represent he fuzzy set member and the grade of membership [4].

A.1. Operations: The fuzzy set operations [5] are defined as follows. Intersection operation of two fuzzy sets uses the symbols: $\cap$, $\wedge$, *, AND, or min. Union operation of two fuzzy sets uses the symbols $\cup$, $\vee$, +, OR, or max. Equality of two sets is defined as $A = B \leftrightarrow \mu_A(x) = \mu_B(x)$ for all $x \in X$. Containment of two sets is defined as $A \subseteq B \leftrightarrow \mu_A(x) \leq \mu_B(x)$ for all $x \in X$. Complement of a set A is defined as A' where $\mu_{A'}(x) = 1 - \mu_A(x)$ for all $x \in X$. We will use also notation with bar on top of the argument to denote negation. Intersection of two sets is defined as $A \cap B$ where $\mu_{A \cap B}(x) = \min\{ \mu_A(x), \mu_B(x) \}$ for all $x \in X$ where $C \subseteq A$, $C \subseteq B$ then $C \subseteq A \cap B$. Union of two sets is defined as $A \cup B$ where $\mu_{A \cup B}(x) = \max\{ \mu_A(x), \mu_B(x) \}$ for all $x \in X$ where $D \supseteq A$, $D \supseteq B$ then $D \supseteq A \cup B$. **Example 1:** An example of fuzzy operations: Let X = {1,2,3,4} and consider the following fuzzy sets A and B: A = {(3, 0.8), (5, 1), (2, 0.6)} and B = {(3, 0.7), (4, 1), (2, 0.5)}. Then $A \cap B$ = {(3, 0.7), (2, 0.5)}, $A \cup B$ = {(3, 0.8), (4, 1), (5, 1), (2, 0.6)}, A' = {(1, 1), (2, 0.4), (3, 0.2), (4, 1), (5, 0)}.

A.2. Identities: The identities use fuzzy variables which are the same as elements in a fuzzy set. The definition of an element in a fuzzy set, {(x, μ_A (x))}, is the same as a fuzzy variable x and this form will be used in the remainder of the paper. Fuzzy functions are made up of fuzzy variables. The identities for fuzzy algebra [6] are: Idempotency: X + X = X* X = X . Commutativity: X + Y = Y + X, X * Y = Y * X . Associativity: (X + Y) + Z = X + (Y + Z) , (X * Y) * Z = X * (Y * Z) . Absorption: X + (X * Y) = X, X * (X + Y) = X . Distributivity: X + (Y * Z) = (X + Y) * (X + Z) , X * (Y + Z) = (X * Y) + (X * Z) . Complement: (X') ' = X . DeMorgan's Laws: (X + Y)' = X' * Y' , (X * Y)' = X' + Y' .

A.3. Transformations: Some transformations of fuzzy sets: x' b + xb = (x + x') b ≠ b , xb + xx'b = xb(1 + x) = xb , x'b + xx'b = x'b (1 + x) = x'b , a + xa = a (1 + x) = a , a + x'a = a (1 + x') = a , a + xx'a = a , a + 0 = a , x * 0 = x , x + 1 = x , x * 1 = x . **Example 2:** a + xa + x' b + xx' b = a(1+x) + x'b(1+x) = a + x' b
Example 3: a + xa + x'a + xx'a = a (1 + x + x' + xx') = a

1.2. **Differences between Boolean Logic and Fuzzy Logic.**

In Boolean logic the value of a variable and its inverse are always disjoint *(X * X' = 0)* and *(X + X' = 1),* because the values are either zero or one. However, in fuzzy logic the membership functions can be either disjoint or non-disjoint. The membership function is determined by the grade of membership and can be any value in the interval [0, 1]. Fuzzy membership functions can be any function that can be realized in the interval from zero to one. For simplicity, the term "grade of membership" of a variable in a set will be replaced by the term "fuzzy variable". An example of a fuzzy non-linear membership function *X* is shown in Fig. 1a with its inverse membership function shown in Fig. 1b. The fuzzy intersection of variables *X* and its complement *X'* is not empty, or is not always equal to zero because the membership functions are non-disjoint. From the membership functions in Figures 2a and 2b the intersection of fuzzy variable *X* and its complement *X'* is shown in Fig. 3a. From the membership functions in Figures 2a and 2b the union of fuzzy variable *X* and its complement *X'* is shown in Fig. 3b.

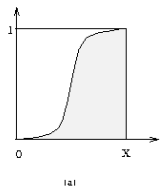
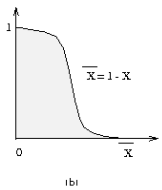
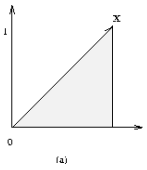


Figure 1. Non-linear membership function and its inverse.





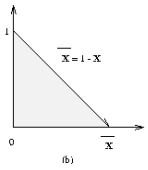
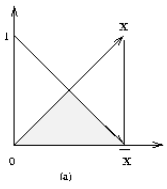


Figure 2. Linear membership function X and inverse



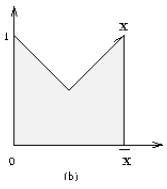


Figure 3. (a) Intersection X * X' ≠ 0. (b) Union X + X' ≠ 1.

2. GRAPHICAL REPRESENTATIONS OF FUZZY FUNCTIONS FOR DECOMPOSITION.

In Karnaugh maps [8] the symbols "1", "0" and "–" (used to denote a don't care) are used to describe minterms and cubes of a binary function and each cell corresponds to a minterm. In contrast, in fuzzy maps the whole terms are represented as cells in the map. Since there is only a finite number of unique terms in a fuzzy function, a symbol I can be used to show if a term is present [7].

Fuzzy Maps. As presented by Schwede and Kandel [9], the fuzzy map may be regarded as an extension of the Veitch diagram [10], which forms the basis for the Karnaugh map. Fuzzy maps pictorially describe the set of all fuzzy implicants which represent a fuzzy function. A K-map of n variables can be represented by 2^n areas (cells) in the map corresponding to care minterms (values 1 and 0) and don't care minterms (values -). A fuzzy map of n variables can be represented by 4^n areas (cells) in the map. The symbol I is used in the map to represent a term existing in the fuzzy function, $F(x_1, x_2, ..., x_n)$. For two variable fuzzy map, the columns are labeled $x_1\ x_1', x_1, x_1', 1$ and the rows are labeled $x_2\ x_2', x_2, x_2', 1,$ as shown in Fig. 4. The column and row headings are conventionally replaced with quaternary numbers representing the binary headings. There are four combinations for each variable x_i , i = 1,2,…,n variables, to be represented in the headings of the rows and columns, as shown in Fig. 5a.

1. This heading is vacuous in x_i . The pair $x_i x'_i$ is denoted by 00 and is represented by 0
2. This heading includes x'_i but not x_i . The pair $x_i x'_i$ is denoted by 01 and represented by 1.
3. This heading includes x_i but not x'_i . The pair $x_i x'_i$ is denoted by 10 and represented by 2.
4. This heading includes x_i and x'_i . The pair $x_i x'_i$ is denoted by 11 and represented by 3.

The construction of fuzzy maps of max, OR, +, as union, and min, AND, *, as intersection, is shown in Fig. 5. The place where I is to be placed is easy to determine. The function of union $f(X_1, X_2) = X_1 + X_2$ is shown in Fig. 5a with the X_1 term that is denoted by the I in the last row because X_2 is vacuous in this term, while the X_2 term is denoted by I in the second row because X_1 is vacuous. In Fig. 5b the function intersection $f(X_1, X_2) = X_1 * X_2$ is shown by placing an I in the column X_1 and row X_2 . Fuzzy map representation has important properties which distinguish them from Boolean maps. As in Boolean maps one can form a cube to reduce the function by circling the ones. In fuzzy maps, the placement of I can show a reduction of the fuzzy logic function. Also another placement of I can show the expansion of the fuzzy logic function. Functions $x_1\ x'_1\ x_2 + x_1\ x'_1\ x'_2 + x_1\ x'_1$ and $x_1\ x'_1$ are equivalent, but have symbols I placed differently. Reduction and transforming to a canonical form of a function correspond then to moving symbols I across the map.

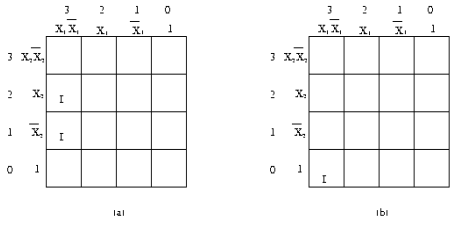


Figure 4. Fuzzy Maps with n=2, (a) $f(x_1,x_2) = x_1\ x'_1\ x_2 + x_1\ x'_1\ x'_2$; (b) $x_1\ x'_1$

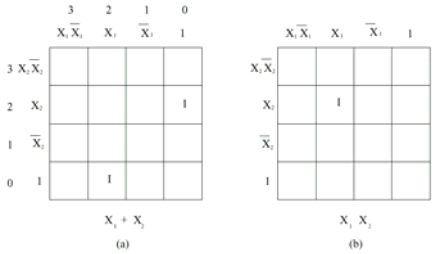


Figure 5. Max and Min representations using fuzzy maps for two variables (a) MAX, (b) MIN

The lattice of two fuzzy variables is shown in Fig. 7a with the most reduced terms on top. The lattice shows the relationship of all the possible terms. The lattice also shows which two terms can be reduced to a single term. In the corresponding fuzzy map of two variables shown in Fig. 7b the highest level is 1 and the lowest is 5. This fuzzy map shows the level in the lattice.

The Subsumtion Rule. The subsume rule is a way to reduce a fuzzy logic function, because rules (X * X' = 0) and (X + X' = 1) are not valid for fuzzy logic. The subsumtion rule is based on the fact that $X * X' \leq 0.5$ and $X + X' \geq 0.5$ and on the rule a + xa = a (1+x) = a. The subsumtion rule is $\alpha\ x_i\ x'_i\ \beta + \alpha'\ x_i\ x'_i\ \beta = x_i$

$x'_i\ \beta$ where α and β can be one or more than one variable [9]. Fig. 7.explains the subsumtion operation on maps of two fuzzy variables, x_1 and , x_2 . In each map, a cell marked with I denotes a term, and the cells marked with i denote all the cells subsumed by cell I. Subsumtion operations for all possible product terms of two variables are shown in Fig. 7.

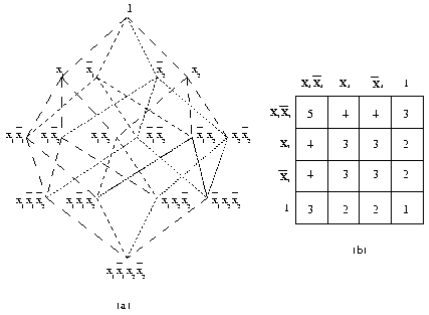


Figure 6. Representations for two variable functions (a) Lattice, (b) Level Map

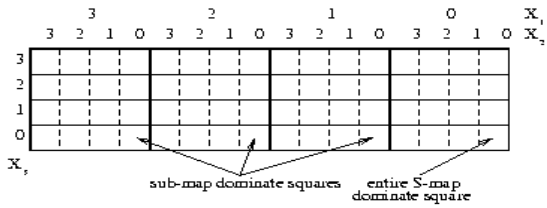


Figure 7. Subsumtion operation for all terms of two variables

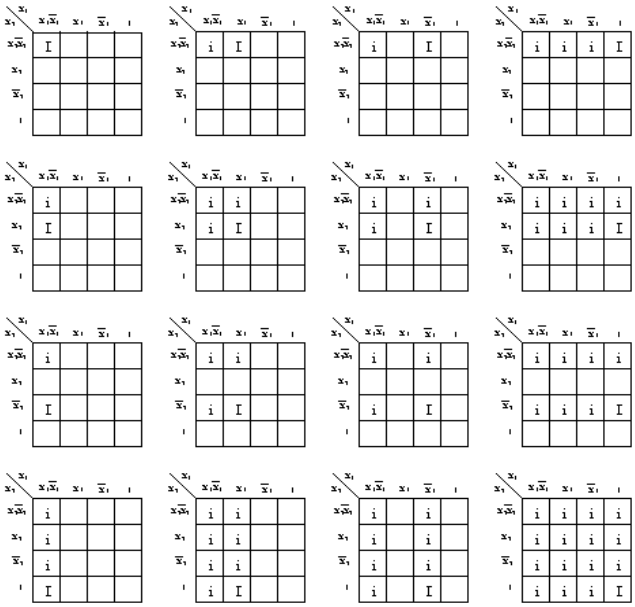


Figure 8. S-map for n =3

S-Maps. S-maps are another way to arrange two-variable fuzzy maps for *n* variables [9]. To construct an *n*-variable S-map, whole one or two variable fuzzy maps are treated as though they were squares of an S-map on *n-1* or *n-2* variables. This method is just iterated for *n* variable S-maps. These subsets of the logical space are called sub-maps and are a very important feature of S-maps. As in fuzzy maps, the binary headings for the columns and rows are converted to a quaternary representation as shown in Fig. 8. The sub-map boundaries are indicated by the vertical solid lines. The same manipulations used on a two-variable map can be used on an *n*-variable S-map. On S-maps, entire sub-map sized patterns behave as single cells in two-variable map [3]. Both fuzzy maps and S-maps have been used in the past to decompose fuzzy switching functions: Fuzzy map is used to find if a decomposition exists. S-map is used to determine the decomposition, or to calculate the predecessor function *G* and successor function *H*.

3. KANDEL'S AND FRANCIONI'S APPROACH TO FUZZY LOGIC DECOMPOSITION.

The approach of Kandel and Francioni [3] was based on graphical representations and it required to reduce the decomposed function to a canonical form. Thus, it was quite inefficient and difficult to program, which was perhaps the reason that it was never implemented in Francioni's Ph. D. Thesis. We are not aware of any other decomposer of fuzzy functions. Below we will briefly present reduction of fuzzy functions to canonical forms to add tutorial value and emphasize the difficulties of Kandel's/Francioni's approach.

Function Form Needed to Decompose a Fuzzy Logic Function in [3]. As a standard, a fuzzy logic function needs to be in a canonical sum-of-products form as the input to decomposition or other minimization procedure. The steps to get a fuzzy logic function into the canonical form are the following, and will be explained next [11]:

1. Represent the fuzzy logic function in sum-of-products form.