A FORMALIZATION OF SHESTAKOV'S DECOMPOSITION

J. J. Lou and J. A. Brzozowski*
Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

Research Report CS-98-03
February 25, 1998

ABSTRACT

We consider two methods for the decomposition of Boolean and multi-valued functions into functions with fewer arguments. Shestakov's method (which we call *double decomposition*) expresses a function $f(x_1, \ldots, x_n)$ as a composite function $h(g_1(u_1, \ldots, u_r), g_2(v_1, \ldots, v_s))$, where the union of $U = \{u_1, \ldots, u_r\}$ and $V = \{v_1, \ldots, v_s\}$ is the set $X = \{x_1, \ldots, x_n\}$. The independently developed method of Łuba and Selvaraj (which we call *single decomposition*) expresses a function $f(x_1, \ldots, x_n)$ as a composite function $h(u_1, \ldots, u_r, g(v_1, \ldots, v_s))$. The latter method has been formalized by Brzozowski and Łuba using "blankets," which are generalizations of set systems. In this paper, we use the same blanket approach to formalize Shestakov's decomposition method. We compare the two methods, and verify that double decomposition is equivalent to two steps of single decomposition. Recently, Brzozowski and Lou extended the single decomposition methods to multi-valued functions. Using the same approach, we also extend Shestakov's method to the multi-valued case.

1. Introduction

Decomposition of Boolean and multi-valued functions has been studied by many authors since the 1950's [1–11]. For a more detailed discussion of this work and its applications see [3]. The present paper is concerned mainly with the methods of Łuba and Selvaraj [7] and Shestakov [10, 11].

We use the following conventions. Ordered n-tuples are denoted by unsubscripted letters and their components by subscripted letters, for example, $x = (x_1, \ldots, x_n)$. If $s = (s_1, \ldots, s_n)$ is an ordered n-tuple, then $S = \{s_1, \ldots, s_n\}$ is the corresponding set. Also, if $S = \{s_1, \ldots, s_n\}$ is a set of elements explicitly represented in that order, then $s = (s_1, \ldots, s_n)$ is the corresponding n-tuple.

*This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant No. OGP0000871 and a postgraduate scholarship.

Łuba and Selvaraj developed an approach to Boolean function decomposition using set systems and their generalizations [7]. They express a function $f(x)$ in the form $h(u, g(v))$, where $x = (x_1, \ldots, x_n)$, $u = (u_1, \ldots, u_r)$, $v = (v_1, \ldots, v_s)$, $U \cup V = X$, and g is a Boolean function. Very similar methods were discovered independently by Shestakov [11], who expresses a Boolean function $f(x)$ in the form $h(g_1(u), g_2(v))$, where u, v, and x are as above, $U \cup V = X$, and g_1 and g_2 are Boolean functions. The method of Łuba and Selvaraj has been formalized by Brzozowski and Łuba in [3], and extended to the multi-valued case by Brzozowski and Lou in [2]. In this paper, we formalize Shestakov's method using blanket algebra as it is described in [3], and also extend it to the multi-valued case. The reader is assumed to be familiar with the work in [2, 3]; here we give only a very brief summary of the main ideas in those papers.

The following statements are given only informally; they are defined below. If a function $f(x)$ can be expressed as $h(u, g(v))$ then the pair (h, g) is a *single separation* of f. A single separation is a *single decomposition* if each of h and g has fewer inputs than f. Similarly, if $f(x)$ can be expressed as $h(g_1(u), g_2(v))$, then (h, g_1, g_2) is a *double separation* of f. It is a *double decomposition* of f if each of h, g_1, and g_2 has fewer inputs than f. We will show later that a double separation is equivalent to the composition of two single separations.

2. Set Functions and Blankets

We will be considering both Boolean and multi-valued functions in the remaining sections. In the following, E is a finite nonempty set, the underlying set of possible values for each variable, and $D = 2^E - \{\emptyset\}$. The set D is partially ordered by set inclusion. In the Boolean case $E = \{0, 1\}$.

For $k > 0$, an element of D^k will be called a *set vector*. The inclusion partial order is extended to D^k: For $t, t' \in D^k$,

$$t \subseteq t' \text{ if and only if } t_i \subseteq t'_i \text{ for all } i,\ 1 \leq i \leq k.$$

The *intersection* of two elements t and t' from D^k is the component-by-component set intersection. This intersection $t \cap t' = (t_1 \cap t'_1, \ldots, t_k \cap t'_k)$ is empty if one (or more) of its components is empty; otherwise, it is an element of D^k. A subset T of D^k is said to be *compatible* if $\bigcap_{t \in T} t \neq \emptyset$. If T is compatible then the greatest lower bound of T exists and is equal to $\bigcap_{t \in T} t$. We denote the *glb* of T as shown below to stress the fact that the *glb* is equal to the intersection only if the intersection is nonempty.

$$glb\, T = \bigsqcap_{t \in T} t.$$

We will be using a matrix notation for the representation of multi-valued functions. A $\rho \times (n + m)$ *set matrix* M is a matrix with elements from D. Each row t of such a matrix is a set vector of length $n + m$, i.e., an $(n + m)$-tuple, where the first n

components, denoted by $t_\downarrow$, are input values and the last m components, denoted by $t^\uparrow$, are output values. We associate an ordered n-tuple x of input variables with the first n columns of the matrix; also X is the corresponding set of input variables. Similarly, y and Y are the tuple and set of output variables associated with the last m columns on M.

In the Boolean case a row vector t of M represents a "function cube", since its entries are $\{0\}$s, $\{1\}$s, and $\{0,1\}$s, representing logic 0s, 1s, and "don't cares," respectively. This terminology is also extended to the multi-valued case, where all the entries are singleton sets—representing particular values, the entire set E of all the possible domain values—representing the complete don't care, or nonempty proper subsets of E—representing partial don't cares.

The set of rows of a set matrix M is denoted by F. A set matrix is said to be *consistent* if it satisfies the following condition: For any set T of rows from F,

$$\text{if } \{t_\downarrow \mid t \in T\} \text{ is compatible, then } \{t^\uparrow \mid t \in T\} \text{ is compatible.}$$

From now on we consider only consistent set matrices.

We refer to an element $(e_1, \ldots, e_n)$ of E^n as a *minterm*. On the other hand, a set vector of the form $e = (\{e_1\}, \ldots, \{e_n\}) \subseteq D^n$, is called a *singleton vector*. The set of all such singleton vectors will be denoted by Σ^n. By a slight abuse of terminology, we also refer to the elements of Σ^n as minterms. To simplify our notation, we write $(e_1, \ldots, e_n)$, or even $e_1 \ldots e_n$, for e when this does not cause confusion. The interpretation of a (consistent) set matrix M is as follows. It defines a multi-valued function $f = f_M$, called *set function*, which is of the form $f : \Sigma^n \to D^m$. For any set vector $e \in \Sigma^n$,

$$f(e) = \bigsqcap_{F_{X \supseteq e}} t^\uparrow,$$

where $F_{X \supseteq e} = \{t \in F \mid t_\downarrow \supseteq e\}$, and, if $F_{X \supseteq e}$ is empty, $f(e)$ is defined to be $(E, \ldots, E)$ (m Es). In other words, the *glb* of the empty set of vectors is interpreted as m Es.

A minterm $e \in \Sigma^n$ is *involved* in a function cube $t \in F$, if $t_\downarrow \supseteq e$. We call e *relevant to* f if e is involved in some vector $t \in F$.

By convention, to simplify matrix M we leave out any row which has $(E, \ldots, E)$ (m Es) as its output.

In the following, we briefly give the definitions and notation used in connection with blankets; for more details see [3].

A *blanket* on a set S is a collection $\beta = \{B_1, \ldots, B_k\}$ of nonempty and distinct subsets of S, called *blocks*, whose union is S. We write $\beta = \{B_i\}$ when it is possible to avoid referring to the number of blocks in β.

The *product* $\beta * \beta'$ of two blankets is defined by

$$\beta * \beta' = ne\{B_i \cap B_j \mid B_i \in \beta \text{ and } B_j \in \beta'\}.$$

where $ne\{S_i\} = \{S_i\} - \{\emptyset\}$ for a set $\{S_i\}$ of subsets of S.

For two blankets β and β' on S, we write $\beta \leq \beta'$ if for each B_i in β there exists a B_j in β' such that $B_i \subseteq B_j$.

Assume the rows in F are numbered $1, \ldots, \rho$. We will be dealing with blankets on F, but we will refer to them as blankets on the set $\{1, \ldots, \rho\}$, for convenience. For an r-element subset U of the input set X, we define $t_{\downarrow}^{U}$ to be the r-tuple of input values from t corresponding to the input variables in U. Define the input blanket β_U corresponding to U to be $\beta_U = ne\{F_{U \supseteq e}\}$, where $e \in \Sigma^r$, and

$$F_{U \supseteq e} = \{t \in F \mid t_{\downarrow}^{U} \supseteq e\}.$$

We also define the output blanket β_Y of f as $\beta_Y = ne\{F_{Y \supseteq e}\}$, where $e \in \Sigma^m$ and

$$F_{Y \supseteq e} = \{t \in F \mid t^{\uparrow} \supseteq e\}.$$

3. Double Separation of Boolean Functions

Let $X = \{x_1, ..., x_n\}$ be the set of input variables of a Boolean function f. Let U and V be two subsets of X such that $U \cup V = X$. Without loss of generality, we relabel the variables $x_1, ..., x_n$ so that $U = \{x_1, ..., x_r\}$ and $V = \{x_{n-s+1}, ..., x_n\}$. Thus for an n-tuple x, the tuple of the first r components is x^U and the tuple of the last s components is x^V. In the next three sections, we assume $E = \{0, 1\}$ and hence $\Sigma = \{\{0\}, \{1\}\}$.

Definition 3.1. *Let f be a Boolean function, with $n > 0$ inputs and $m > 0$ outputs, and let (U, V) be as above. Assume that f is specified by a set F of function cubes. Let g_1 be a Boolean function with r inputs and p outputs, let g_2 be a Boolean function with s inputs and q outputs, and let h be a Boolean function with $p + q$ inputs and m outputs. The triple (g_1, g_2, h) is a* double separation *of f with respect to (U, V), if, for every minterm $b \in \Sigma^n$ relevant to f, $g_1(b^U) \in \Sigma^p$, $g_2(b^V) \in \Sigma^q$, the vector $(g_1(b^U), g_2(b^V))$ is relevant to h, and*

$$f(b) \supseteq h(g_1(b^U), g_2(b^V)). \tag{1}$$

The separation described in [3] is a special case of the above separation where function g_1 is the identity function.

Definition 3.2. *Under the conditions stated above, let W be a subset of X with k elements. Let g be a Boolean function with k inputs. Then a blanket β on F* corresponds to g with respect to W *if the following condition is satisfied: Two tuples t and u of F appear together in some block of β if there exist two minterms d and e relevant to f such that $d \subseteq t_{\downarrow}^{W}$, $e \subseteq u_{\downarrow}^{W}$, and $g(d) = g(e)$.*

Theorem 3.3. *Let $f(x)$ be a Boolean function with the set F of rows and let (U, V) be a pair of subsets of X satisfying $U \cup V = X$. For every pair $(\beta_{g_1}, \beta_{g_2})$ of blankets satisfying the conditions stated below, there is a double separation (g_1, g_2, h) such that β_{g_1} corresponds to g_1 with respect to U, and β_{g_2} corresponds to g_2 with respect to V.*

- $\beta_U \leq \beta_{g_1}$,
- $\beta_V \leq \beta_{g_2}$, *and*
- $\beta_{g_1} * \beta_{g_2} \leq \beta_Y$.

Proof. Suppose β_{g_1} and β_{g_2} satisfying the conditions of the theorem are given. Following the method in [3], we can construct g_1 and g_2 from β_{g_1} and β_{g_2}, respectively. See the examples in Section 4 for more details.

Next we construct the truth table for a function h with $p+q$ inputs and m outputs. Consider $x = x_1 x_2$ with $x_1 \in \Sigma^p$ and $x_2 \in \Sigma^q$. Define $F_{g_1 = x_1}$ to be the block of β_{g_1} that is assigned the value x_1, and $F_{g_2 = x_2}$ to be the block of β_{g_2} that is assigned the value x_2. If $F_{g_1 = x_1} \cap F_{g_2 = x_2} = B \neq \emptyset$, assign block B to x. Otherwise omit x from the table for h. By the condition $\beta_{g_1} * \beta_{g_2} \leq \beta_Y$, we know that B is contained in some block $\hat{B}$ of the output blanket β_Y. Thus we know that $\bigcap_{t \in B} t^{\uparrow}$ exists. Assign $h(x)$ this value.

The proof that the above construction is correct closely follows the proof in [3] and is omitted.

The converse of the above theorem is also true if U and V are disjoint; the proof is again similar to the one in [3]. As we pointed out earlier, the separation in [3] is a special case of the separation described here. Thus the counterexample for the converse of the theorem in that paper also serves as a counterexample here when U and V are not disjoint.

Now we shall prove that double separation is equivalent to two single separations performed in sequence.

Proposition 3.4. *Let f be a Boolean function and let (g_1, g_2, h) be a double separation of f. Let r, q and m be as before. Then there exists a function h' with $r + q$ inputs and m outputs such that (g_2, h') is a single separation of f and (g_1, h) is a single separation of h'.*

Proof. We define h' as follows. For $x = cx'$ where $c \in \Sigma^r$ and $x' \in \Sigma^q$, if there exists a minterm b relevant to f such that $b^U = c$ and $g_2(b^V) = x'$, then we know that $x_1 = g_1(c)$ exists and $h(x_1, x')$ also exists. In this case, we define $h'(x) = h(x_1, x')$. Otherwise x is not present in the table for h'. Now for every minterm b relevant to f,

$$f(b) \supseteq h(g_1(b^U), g_2(b^V)) = h'(b^U, g_2(b^V)).$$

Thus (g_2, h') is a single separation of f.

Let the set of outputs of g_2 be V'. Then the input set of h' is $U \cup V'$ and for every minterm d relevant to h',

$$h'(d) = h'(d^U, d^{V'}) = h(g_1(d^U), d^{V'}).$$

Thus (g_1, h) is a single separation of h'. In fact the $\supseteq$ in the definition is replaced by strict equality here.

4. Examples of Separations

Let f be a Boolean function with five inputs and two outputs specified by the matrix in Table 1. Note that Φ is a short hand for $\{0,1\}$ and the $\{\}$ is omitted for all other entries. This example is taken from [11].

Table 1: Matrix defining f.

Row	x_1	x_2	x_3	x_4	x_5	y_1	y_2
1	0	0	1	0	1	0	1
2	0	1	1	Φ	Φ	1	1
3	1	Φ	0	0	1	Φ	0
4	Φ	1	Φ	1	0	Φ	1
5	1	1	1	0	Φ	0	1
6	0	Φ	0	Φ	1	0	0
7	0	0	1	1	1	1	0
8	1	Φ	0	1	1	Φ	1

Example 4.1. *First we use Shestakov's method. Suppose we have* $U = \{x_1, x_2, x_3\}$, *and* $V = \{x_3, x_4, x_5\}$. *Then*

$$\beta_U = \{\overset{000}{\overline{6}};\ \overset{001}{\overline{1,7}};\ \overset{010}{\overline{4,6}};\ \overset{011}{\overline{2,4}};\ \overset{100}{\overline{3,8}};\ \overset{110}{\overline{3,4,8}};\ \overset{111}{\overline{4,5}}\},$$

$$\beta_V = \{\overset{001}{\overline{3,6}};\ \overset{010}{\overline{4}};\ \overset{011}{\overline{6,8}};\ \overset{100}{\overline{2,5}};\ \overset{101}{\overline{1,2,5}};\ \overset{110}{\overline{2,4}};\ \overset{111}{\overline{2,7}}\},$$

and

$$\beta_Y = \{\overset{00}{\overline{3,6}};\ \overset{01}{\overline{1,4,5,8}};\ \overset{10}{\overline{3,7}};\ \overset{11}{\overline{2,4,8}}\},$$

where, for clarity, we omit the $\{\}$ *for each block* B_i *and write* $\overline{B_i}$ *instead. Also, the minterms corresponding to each block are indicated above each block. Now, let* $\beta_{g_1} = \{\overline{2,3,4,8};\ \overline{1,4,5,6,7}\}$, $\beta_{g_2} = \{\overline{3,6};\ \overline{1,2,4,5};\ \overline{6,8};\ \overline{2,7}\}$. *We can check that these two blankets satisfy the conditions of Theorem 3.3. We encode* β_{g_1} *with one variable and* β_{g_2} *with two variables. One such encoding is as follows:*

$$\beta_{g_1} = \{\overset{0}{\overline{2,3,4,8}};\ \overset{1}{\overline{1,4,5,6,7}}\},$$

$$\beta_{g_2} = \{\overset{00}{\overline{3,6}};\ \overset{01}{\overline{1,2,4,5}};\ \overset{10}{\overline{6,8}};\ \overset{11}{\overline{2,7}}\}.$$

Using this encoding, we construct the tables for g_1, g_2 *and then* h. *The matrices for the functions are given in Tables 2–4.*

Table 2: Finding function g_1.

x_1	x_2	x_3	β_U	β_{g_1}	z_1
0	0	0	$\overline{6}$	$\overline{1,4,5,6,7}$	1
0	0	1	$\overline{1,7}$	$\overline{1,4,5,6,7}$	1
0	1	0	$\overline{4,6}$	$\overline{1,4,5,6,7}$	1
0	1	1	$\overline{2,4}$	$\overline{2,3,4,8}$	0
1	0	0	$\overline{3,8}$	$\overline{2,3,4,8}$	0
1	1	0	$\overline{3,4,8}$	$\overline{2,3,4,8}$	0
1	1	1	$\overline{4,5}$	$\overline{1,4,5,6,7}$	1

To get g_1, we find, for each input combination, the corresponding block B of β_U. Then we get the block of β_{g_1} that contains B and assign the output to be the encoding of that block of β_{g_1}. For example, for $x_1x_2x_3 = 000$, the block of β_U is $\overline{6}$, which is contained in $\overline{1,4,5,6,7}$. Hence $g_1(000)$ is assigned the value 1 here. The rest of the construction of g_1 and g_2 follows similarly.

Table 3: Finding function g_2.

x_3	x_4	x_5	β_V	β_{g_2}	z_2	z_3
0	0	1	$\overline{3,6}$	$\overline{3,6}$	0	0
0	1	0	$\overline{4}$	$\overline{1,2,4,5}$	0	1
0	1	1	$\overline{6,8}$	$\overline{6,8}$	1	0
1	0	0	$\overline{2,5}$	$\overline{1,2,4,5}$	0	1
1	0	1	$\overline{1,2,5}$	$\overline{1,2,4,5}$	0	1
1	1	0	$\overline{2,4}$	$\overline{1,2,4,5}$	0	1
1	1	1	$\overline{2,7}$	$\overline{2,7}$	1	1

To construct function $h = h(z_1, z_2, z_3)$, we first find block B of β_{g_1} corresponding to input z_1 and block B' of β_{g_2} corresponding to inputs z_2z_3. Next we calculate the product $\hat{B} = B{*}B'$. The output is then the greatest lower bound of the outputs of the rows contained in $\hat{B}$. For example, for $z_1z_2z_3 = 001$, the corresponding block of β_{g_1} is $B = \overline{2,3,4,8}$ and the corresponding block of β_{g_2} is $B' = \overline{1,2,4,5}$. The product of the two blocks is $\hat{B} = \overline{2,4}$. Hence the output is $glb\{11, \Phi 1\} = 11$. The rest of the table for h follows similarly.

Example 4.2. *Now we use the method described in Proposition 3.4 to construct a function h' for single separation.*

In Table 5, we list the possible input combinations $(x_1, x_2, x_3, z_2, z_3)$ to function h', then $z_1 = g_1(x_1, x_2, x_3)$, and finally the two outputs of $h(z_1, z_2, z_3)$ which are the

Table 4: Matrix defining h.

z_1	z_2	z_3	$\beta_{g_1} * \beta_{g_2}$	y_1	y_2
0	0	0	$\overline{3}$	Φ	0
0	0	1	$\overline{2,4}$	1	1
0	1	0	$\overline{8}$	Φ	1
0	1	1	$\overline{2}$	1	1
1	0	0	$\overline{6}$	0	0
1	0	1	$\overline{1,4,5}$	0	1
1	1	0	$\overline{6}$	0	0
1	1	1	$\overline{7}$	1	0

outputs of h'.

Note that those input combinations for h' *which do not correspond to a minterm* b *are omitted. For example, according to the tables for* g_1, g_2, *and* h, *the input combination* 00001 *is possible and it would correspond to minterm* $b = 00010$*; however* b *is not relevant to* f, *and thus input* 00001 *is not in the table for* h'.

We can check that the pair (g_2, h') *is a single separation of* f, *and that the pair* (g_1, h) *is a single separation of* h'.

5. Decomposition Method Using Tables

We now describe Shestakov's method for finding the blankets β_{g_1} and β_{g_2}; this is the "table reduction" method [11]. The relation between this method and the construction we used earlier will also be shown. This method can be used to do both single and double separations. We shall use it to derive a single separation first then proceed to derive a double separation. Note that our single separation will be of the form $h(g_1(u), v)$, so the usual roles of U and V are reversed.

The function specified by Table 1 will be used. We label the blocks of β_U and β_V by the integer corresponding to the binary input combination. Hence, for example, block B_5 is missing from β_U as $(x_1, x_2, x_3) = (1, 0, 1)$ is not present in the matrix defining f.

$$\beta_U = \{\overset{B_0}{\overline{6}};\ \overset{B_1}{\overline{1,7}};\ \overset{B_2}{\overline{4,6}};\ \overset{B_3}{\overline{2,4}};\ \overset{B_4}{\overline{3,8}};\ \overset{B_6}{\overline{3,4,8}};\ \overset{B_7}{\overline{4,5}}\},$$

$$\beta_V = \{\overset{B'_1}{\overline{3,6}};\ \overset{B'_2}{\overline{4}};\ \overset{B'_3}{\overline{6,8}};\ \overset{B'_4}{\overline{2,5}};\ \overset{B'_5}{\overline{1,2,5}};\ \overset{B'_6}{\overline{2,4}};\ \overset{B'_7}{\overline{2,7}}\}.$$

We construct a *block multiplication table* where the rows are labeled B_i and columns are labeled B'_j. In order to define the entries in the table, we need the following definitions.

Table 5: Matrix defining h'.

x_1	x_2	x_3	z_2	z_3	z_1	y_1	y_2
0	0	0	0	0	1	0	0
0	0	0	1	0	1	0	0
0	0	1	0	1	1	0	1
0	0	1	1	0	1	0	0
0	0	1	1	1	1	1	0
0	1	0	0	0	1	0	0
0	1	0	0	1	1	0	1
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	1
0	1	1	1	1	0	1	1
1	0	0	0	0	0	Φ	0
1	0	0	1	0	0	Φ	1
1	1	0	0	0	0	Φ	0
1	1	0	0	1	0	1	1
1	1	0	1	0	0	Φ	1
1	1	1	0	1	1	0	1

Definition 5.1. *Let W be a subset of X with k elements. Let B be a block in the blanket β_W. Then $b \in \Sigma^k$* exemplifies *B if $b \subseteq \bigcap_{t\in B} t_{\downarrow}^{W}$ and $b \not\subseteq \bigcap_{t\in B'} t_{\downarrow}^{W}$ for any other block B' of β_W.*

For example, if $W = U$ for the function of Table 1, then 000 exemplifies B_6, but 010 does not.

Definition 5.2. *Given a block B_i from β_U and a block B'_j from β_V, we define $B_i \wedge B'_j = B_i \cap B'_j$ if*

- *$B_i \cap B'_j \neq \emptyset$,*
- *there exists a minterm b relevant to f such that b^U exemplifies B_i and b^V exemplifies B'_j.*

Otherwise, we define $B_i \wedge B'_j = \emptyset$.

Intuitively, the operation $\wedge$ is the block intersection if there is a common minterm which exemplifies both blocks, and empty otherwise. In fact, we can show that in the case $U \cap V = \emptyset$, this operation is equivalent to block intersection. Entry (i, j) in the table $T_{i,j}$ is now defined to be $B_i \wedge B'_j$. The block multiplication table for our function is presented in Table 6. For clarity, empty sets are represented by $-$ and we omit the $\{\}$ for all other entries.

Table 6: Block Multiplication Table.

	B'_1	B'_2	B'_3	B'_4	B'_5	B'_6	B'_7
B_0	6	–	–	6	–	–	–
B_1	–	–	–	–	1	–	7
B_2	6	4	6	–	–	–	–
B_3	–	–	–	2	2	2, 4	2
B_4	3	–	8	–	–	–	–
B_6	3	4	8	–	–	–	–
B_7	–	–	–	5	5	4	–

Note that $T_{2,6}$ is empty instead of $B_2 \cap B'_6 = \{4\}$. This is because all minterms that exemplify B_2 satisfy $x_3 = 0$, while all minterms that exemplify B'_6 satisfy $x_3 = 1$; thus the minterm b in our definition does not exist and we have $B_3 \wedge B'_6 = \emptyset$. Now we replace $T_{i,j}$ with $\bigcap_{t \in T_{i,j}} t^{\uparrow}$ if $T_{i,j}$ is not empty. If $T_{i,j}$ is empty, it is replaced by (Φ, Φ). For clarity, we drop the parentheses and denote each entry by tuples. Also we use $-$ to denote the complete don't care, which is $\Phi\Phi$ in this case. We call Table 7 the *row compatibility table.*

Table 7: Row Compatibility Table.

	B'_1	B'_2	B'_3	B'_4	B'_5	B'_6	B'_7
B_0	00	–	00	–	–	–	–
B_1	–	–	–	–	01	–	10
B_2	00	$\Phi 1$	00	–	–	–	–
B_3	–	–	–	11	11	11	11
B_4	$\Phi 0$	–	$\Phi 1$	–	–	–	–
B_6	$\Phi 0$	$\Phi 1$	$\Phi 1$	–	–	–	–
B_7	–	–	–	01	01	$\Phi 1$	–

Definition 5.3. *A subset $S = \{B_{i_1}, \ldots, B_{i_k}\}$ of $\{B_i\}$ is called* compatible *if, in each column of the row compatibility table, the entries in rows $i_1, \ldots, i_k$ are compatible, i.e., they have a greatest lower bound.*

We now find a blanket β_{B_i} on the set $\{B_i\}$ where the blocks are compatible sets. Each block of this blanket is a set of blocks in β_U. We obtain a blanket β on F by replacing each block with the union of its elements. We then use blanket β as β_{g_1} to obtain a single separation. Note this is equivalent to Łuba and Selvaraj's method [7], where blocks are merged in β_U to get β such that $\beta_V * \beta \leq \beta_Y$. For some graph theoretical methods to obtain β, see [3]. Here we just use the brute force method of taking a block B_i of β_U, trying to merge it with an existing block in β_{B_i} and creating a new block for β_{B_i} whenever B_i is not compatible with any existing block. For example, B_0, B_1 and B_2 are compatible, while B_3 is not compatible with the first three, so it's in a separate block. Now B_4 is compatible with B_3 so they are in the same block. The rest of the merging is done similarly.

In our example, one possible blanket β_{B_i} on the set $\{B_i\}$ is

$$\beta_{B_i} = \{\overline{B_0, B_1, B_2, B_7};\ \overline{B_3, B_4, B_6}\}.$$

The corresponding blanket β_{g_1} with one possible encoding is

$$\beta_{g_1} = \{\overset{0}{\overline{1, 4, 5, 6, 7}};\ \overset{1}{\overline{2, 3, 4, 8}}\}.$$

Note that this is the same blanket as we used in the last section. We can check that $\beta_{g_1} * \beta_V \leq \beta_Y$. Thus this blanket corresponds to a single separation of f.

Denote the block encoded 0 by δ_0 and the block encoded 1 by δ_1. We construct the *column compatibility table* as follows. The rows of the table are labeled δ_l and the columns labeled B'_j. The entry (l, j) of the table is the greatest lower bound of entries (i_k, j) in the row compatibility table where $\delta_l = \{B_{i_k}\}$. This table is shown in Table 8.

Table 8: Column Compatibility Table.

	B'_1	B'_2	B'_3	B'_4	B'_5	B'_6	B'_7
δ_0	00	Φ1	00	01	01	Φ1	10
δ_1	Φ0	Φ1	Φ1	11	11	11	11

Now we use similar procedure as before to find compatible columns.

Definition 5.4. *A subset $S' = \{B'_{i_1}, \ldots, B'_{i_k}\}$ of $\{B'_i\}$ is called* compatible *if in each row of the column compatibility table, the entries in columns $i_i, \ldots, i_k$ are compatible, i.e., they have a greatest lower bound.*

From our table, we obtain the following blanket on the set $\{B'_j\}$.

$$\beta_{B'_j} = \{\overline{B'_1};\ \overline{B'_2, B'_4, B'_5, B'_6};\ \overline{B'_3};\ \overline{B'_7}\}.$$

The corresponding blanket β_{g_2} with one possible encoding is

$$\beta_{g_2} = \{\overset{00}{\overline{3, 6}};\ \overset{01}{\overline{1, 2, 4, 5}};\ \overset{10}{\overline{6, 8}};\ \overset{11}{\overline{2, 7}}\}.$$

As we can see, the blanket β_{g_2} we derived is also the same as the one we used in the previous section. We label the four blocks $\delta'_0, \ldots, \delta'_3$. Using similar methods as we used in the construction of the column compatibility table, we merge columns this time and the result is shown in Table 9. We can get function h directly from this table: each input combination of h is formed by concatenating the encodings of δ_i and δ'_j while the output tuple is the entry (i, j) of the table. As usual, we can omit an input combination from h if the output tuple is a complete don't care (denoted by $-$ in our tables).

Table 9: Final Table.

	δ'_0	δ'_1	δ'_2	δ'_3
δ_0	00	01	00	10
δ_1	Φ0	11	Φ1	11

Note that we could have used Table 7 as our column compatibility table and merged the columns first. This is equivalent to exchanging the sets U and V. If we do this in our example, we get

$$\beta_{B'_j} = \{\overline{B'_1, B'_4, B'_5, B'_6};\ \overline{B'_2, B'_7};\ \overline{B'_3}\}.$$

This gives

$$\beta_{g_2} = \{\overset{00}{\overline{1,2,3,4,5,6}};\ \overset{01}{\overline{2,4,7}};\ \overset{10}{\overline{6,8}}\}.$$

Table 10: New Row Compatibility Table.

	δ'_0	δ'_1	δ'_2
B_0	00	$-$	00
B_1	01	10	$-$
B_2	00	Φ1	00
B_3	11	11	$-$
B_4	Φ0	$-$	Φ1
B_6	Φ0	Φ1	Φ1
B_7	01	$-$	$-$

Table 10 is our new row compatibility table. We can get β_{B_i} and then β_{g_1} from this table. They are as follows:

$$\beta_{B_i} = \{\overline{B_0, B_2};\ \overline{B_1, B_7};\ \overline{B_3};\ \overline{B_4, B_6}\}.$$

Table 11: New Final Table.

	δ'_0	δ'_1	δ'_2
δ_0	00	Φ1	00
δ_1	01	10	–
δ_2	11	11	–
δ_3	Φ0	Φ1	Φ1

$$\beta_{g_1} = \{\overset{00}{\overline{4,6}};\ \overset{01}{\overline{1,4,5,7}};\ \overset{10}{\overline{2,4}};\ \overset{11}{\overline{3,4,8}}\}.$$

We can now deduce our new final table from β_{g_1}. The result is shown in Table 11. Note this new table is 4×3 as opposed to the 2×4 table we obtained earlier. Thus the order in which we reduce our table is important. In our example, the set of tables for our second double separation would have larger total size than the tables for our first double separation.

We now prove that the blankets β_{g_1} and β_{g_2} obtained by the above procedure give us a double separation of f. The proof that β_{g_1} gives us a single separation is similar and so it is omitted.

Theorem 5.5. *The table reduction method results in a double separation of the original function f.*

Proof. Let b be a minterm relevant to f. Then b^U exemplifies a block B_i in the blanket β_U and b^V exemplifies a block B'_j in the blanket β_V. We know that $B_i \cap B'_j \neq \emptyset$, as the row that contains b is in the intersection. Further, by definition, we have $B_i \wedge B'_j = B_i \cap B'_j$. In our final table obtained using the procedure, there is an entry corresponding to the value $h(g_1(b^U), g_2(b^V))$. This entry is the greatest lower bound of some output vectors, one of which is the entry (i, j) in the block compatibility table. Since that entry is simply $f(b)$, we get $f(b) \supseteq h(g_1(b^U), g_2(b^V))$ as required.

6. Decomposition of Multi-Valued Functions

We now extend the method in the previous section to decomposition of multi-valued functions. In this section, unlike in previous sections, the set E has more than two elements. The following is an example of a set function, where $E = \{0, 1, 2\}$. As usual, for clarity, we drop $\{\}$ from the entries.

In the following, we shall state the multi-valued version of Theorem 3.3 and then work through a double decomposition of f using the table method.

Theorem 6.1. *Let $f(x)$ be a set function specified by a set F of set vectors, and let (U, V) be a pair of subsets of X satisfying $U \cup V = X$. For every pair of blankets β_{g_1}*

Table 12: Set Function f.

Row	x_1	x_2	x_3	x_4	x_5	y_1	y_2
1	1	0,1	0	0	0,1	1	0
2	1,2	1	0	1,2	2	0	1,2
3	0,1	0	0	0,1	0	1,2	0
4	2	0,2	1,2	2	2	0,1	2
5	0,1	0	0,2	1	1	2	1,2
6	0	0	0,1	0	2	1	1
7	0,1,2	1	0,2	0	2	0	0
8	0	2	0	0	2	1	2

and β_{g_2} satisfying the conditions stated below, there is a double separation (g_1, g_2, h) of f such that β_{g_1} corresponds to g_1 with respect to U, and β_{g_2} corresponds to g_2 with respect to V.

- $\beta_U \leq \beta_{g_1}$, *and*
- $\beta_V \leq \beta_{g_2}$, *and*
- $\beta_{g_1} * \beta_{g_2} \leq \beta_Y$.

The proof of this theorem is similar to the proof of the corresponding theorem in [2] and is omitted.

Consider the set function f specified by Table 12. Let $U = \{x_1, x_2, x_3\}$ and $V = \{x_3, x_4, x_5\}$, then

$$\beta_U = \{\overset{B_1}{\overline{3,5,6}};\ \overset{B_2}{\overline{6}};\ \overset{B_3}{\overline{5}};\ \overset{B_4}{\overline{7}};\ \overset{B_5}{\overline{8}};\ \overset{B_6}{\overline{1,3,5}};\ \overset{B_7}{\overline{1,2,7}};\ \overset{B_8}{\overline{4}};\ \overset{B_9}{\overline{2,7}}\},$$

$$\beta_V = \{\overset{B_1'}{\overline{1,3}};\ \overset{B_2'}{\overline{1}};\ \overset{B_3'}{\overline{6,7,8}};\ \overset{B_4'}{\overline{3}};\ \overset{B_5'}{\overline{5}};\ \overset{B_6'}{\overline{2}};\ \overset{B_7'}{\overline{6}};\ \overset{B_8'}{\overline{4}};\ \overset{B_9'}{\overline{7}}\},$$

and

$$\beta_Y = \{\overset{00}{\overline{7}};\ \overset{01}{\overline{2}};\ \overset{02}{\overline{2,4}};\ \overset{10}{\overline{1,3}};\ \overset{11}{\overline{6}};\ \overset{12}{\overline{4,8}};\ \overset{20}{\overline{3}};\ \overset{21,22}{\overline{5}}\}.$$

Note that the tuples on top of each block of β_Y are the tuples that exemplify that block. We omit such tuples from β_U and β_V; instead, we show the labels to be used in subsequent tables.

Just as before, we first construct the block multiplication table with rows labeled B_i and columns labeled B_j', with entries $B_i \wedge B_j'$. This is shown in Table 13.

We then replace the entries by corresponding outputs to get the row compatibility table shown in Table 14.

Table 13: Block Multiplication Table.

	B'_1	B'_2	B'_3	B'_4	B'_5	B'_6	B'_7	B'_8	B'_9
B_1	3	–	6	3	5	–	–	–	–
B_2	–	–	–	–	–	–	6	–	–
B_3	–	–	–	–	5	–	–	–	–
B_4	–	–	7	–	–	–	–	–	7
B_5	–	–	8	–	–	–	–	–	–
B_6	1, 3	1	–	3	5	–	–	–	–
B_7	1	1	7	–	–	2	–	–	–
B_8	–	–	–	–	–	–	–	4	–
B_9	–	–	7	–	–	2	–	–	–

Table 14: Row Compatibility Table.

	B'_1	B'_2	B'_3	B'_4	B'_5	B'_6	B'_7	B'_8	B'_9
B_1	$\{1,2\}\{0\}$	–	$\{1\}\{1\}$	$\{1,2\}\{0\}$	$\{2\}\{1,2\}$	–	–	–	–
B_2	–	–	–	–	–	–	$\{1\}\{1\}$	–	–
B_3	–	–	–	–	$\{2\}\{1,2\}$	–	–	–	–
B_4	–	–	$\{0\}\{0\}$	–	–	–	–	–	$\{0\}\{0\}$
B_5	–	–	$\{1\}\{2\}$	–	–	–	–	–	–
B_6	$\{1\}\{0\}$	$\{1\}\{0\}$	–	$\{1,2\}\{0\}$	$\{2\}\{1,2\}$	–	–	–	–
B_7	$\{1\}\{0\}$	$\{1\}\{0\}$	$\{0\}\{0\}$	–	–	$\{0\}\{1,2\}$	–	–	–
B_8	–	–	–	–	–	–	–	$\{0,1\}\{2\}$	–
B_9	–	–	$\{0\}\{0\}$	–	–	$\{0\}\{1,2\}$	–	–	–

Two rows in the row compatibility table are considered to be compatible if the *glb* of the corresponding entries exist. Using the simple algorithm described earlier to merge blocks, we get

$$\beta_{B_i} = \{\overline{B_1, B_2, B_3, B_6, B_8};\ \overline{B_4, B_7, B_9};\ \overline{B_5}\}.$$

The blanket on F corresponding to β_{B_i} is

$$\beta_{g_1} = \{\overset{0}{\overline{1,3,4,5,6}};\ \overset{1}{\overline{1,2,7}};\ \overset{2}{\overline{8}}\}.$$

We construct the column compatibility table by merging the rows appearing in the same block of β_{B_i}. The result is shown in Table 15.

Table 15: Column Compatibility Table.

	B'_1	B'_2	B'_3	B'_4	B'_5	B'_6	B'_7	B'_8	B'_9
δ_0	{1}{0}	{1}{0}	{1}{1}	{1, 2}{0}	{2}{1, 2}	–	{1}{1}	{0, 1}{2}	–
δ_1	{1}{0}	{1}{0}	{0}{0}	–	–	{0}{1, 2}	–	–	{0}{0}
δ_2	–	–	{1}{2}	–	–	–	–	–	–

As before, we partition the columns into compatible blocks. The result of using our simple brute force algorithm is as follows:

$$\beta_{B'_j} = \{\overline{B'_1, B'_2, B'_4};\ \overline{B'_3, B'_7, B'_9};\ \overline{B'_5, B'_6};\ \overline{B'_8}\}.$$

This corresponds to

$$\beta_{g_2} = \{\overset{00}{\overline{1,3}};\ \overset{01}{\overline{6,7,8}};\ \overset{02}{\overline{2,5}};\ \overset{10}{\overline{4}}\}.$$

To get functions g_1, g_2 and h, we use similar methods to the one used in the previous section. The tables for these functions are shown in Tables 16, 17 and 18, respectively.

Consider the following blankets:

$$\beta_{g'_1} = \{\overline{1,3,5,6};\ \overline{1,2,7};\ \overline{4,8}\},$$

$$\beta_{g'_2} = \{\overline{1,3,4};\ \overline{6,7,8};\ \overline{2,5}\}.$$

We can check that these blankets satisfy the conditions of Theorem 6.1. As we can see both blankets have only three blocks. Thus by using these blankets, both g_1 and g_2 would have one output and h would have only two inputs. This new decomposition would be an improvement over our current one. This shows that two blankets β_{g_1} and $\beta_{g'_1}$, even if they are of the same size, may lead to two blankets β_{g_2} and $\beta_{g'_2}$ with different sizes.

Table 16: Matrix defining g_1.

x_1	x_2	x_3	z_1
0	0	0	0
0	0	1	0
0	0	2	0
0	1	0	1
0	1	2	1
0	2	0	2
1	0	0	0
1	0	2	0
1	1	0	1
1	1	2	1
2	0	1	0
2	0	2	0
2	1	0	1
2	1	2	1
2	2	1	0
2	2	2	0

Table 17: Matrix defining g_2.

x_4	x_5	x_6	z_2	z_3
0	0	0	0	0
0	0	1	0	0
0	0	2	0	1
0	1	0	0	0
0	1	1	0	1
0	1	2	0	2
0	2	2	0	2
1	0	2	0	1
1	2	2	1	0
2	0	2	0	1
2	1	1	0	2
2	2	2	1	0

Table 18: Matrix defining h.

z_1	z_2	z_3	f_1	f_2
0	0	0	1	0
0	0	1	1	1
0	0	2	2	1, 2
0	1	0	0, 1	2
1	0	0	1	0
1	0	1	0	0
1	0	2	0	1, 2
2	0	1	1	2

7. Conclusions

From Theorem 3.4, we see that all separation results achievable by the double-separation method are achievable with repeated application of the single-separation method. The latter method is more flexible as we are free to choose the sets U and V for the second separation and perhaps get a better result.

References

1. R. L. Ashenhurst, The Decomposition of Switching Functions, *Proc. of International Symp. Theory of Switching Functions*, 1959.

2. J. A. Brzozowski and J. J. Lou, Blanket algebra for multiple-valued function decomposition, manuscript available at `http://maveric0.uwaterloo.ca/publication.html`, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, June 1997, to be published by World Scientific.

3. J. A. Brzozowski and T. Łuba, Decomposition of Boolean Functions Specified by Cubes, Research Report CS–97–01, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, January 1997.

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

5. S. Grygiel et al., Cube Diagram Bundles: A New Representation of Strongly Unspecified Multiple-Valued Functions and Relations, *Proc. IEEE International Symposium on Multiple-Valued Logic,* Antigonish, Nova Scotia, Canada, pp. 287–292, May 1997.

6. T. Łuba, Decomposition of Multiple-Valued Functions, *Proc. 25th International Symposium on Multiple-Valued Logic,* Bloomington, Indiana, pp. 256–261, May 1995.

7. T. Łuba and H. Selvaraj, A General Approach to Boolean Function Decomposition and its Applications in FPGA-Based Synthesis, VLSI Design, Vol.3, Nos 3-4, pp.289-300, 1995.

8. M. Perkowski et al., Decomposition of Multiple-Valued Relations, *Proc. IEEE International Symposium on Multiple-Valued Logic,* Antigonish, Nova Scotia, Canada, pp. 13–18, May 1997.

9. J. P. Roth and R. M. Karp, Minimization over Boolean Graphs, *IBM Journal of Research and Development,* Vol. 6, pp. 227–238, April 1962.

10. E. Shestakov, Decomposition of Systems of Completely Defined Boolean Functions by Argument Covering, *Automatic Control and Computer Sciences,* Vol. 28, No. 1, pp. 12–20, 1994.

11. E. Shestakov, Decomposition of Systems of Incompletely Defined Boolean Functions by Argument Cover, *Automatic Control and Computer Sciences,* Vol. 28, No. 6, pp. 4–15, 1994.