Decomposition of Boolean Functions Specified by Cubes.*

J. A. BRZOZOWSKI
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

T. ŁUBA
Institute of Telecommunications
Warsaw University of Technology
Nowowiejska 15/19
00-665 Warsaw, Poland

Abstract We study the problem of decomposing a Boolean function into a set of functions with fewer arguments. This problem has considerable practical importance in VLSI, for example, for designs using field-programmable gate arrays. The decomposition problem is old, and well understood when the function to be decomposed is specified by a truth table, or has one output only. However, modern design tools handle functions with many outputs and represent them by cubes, for reasons of efficiency. We develop a comprehensive theory of serial decompositions for multiple-output, partially specified, Boolean functions represented by cubes. A function $f(x_1, \ldots, x_n)$ has a serial decomposition if it can be expressed as $h(u_1, \ldots, u_r, g(v_1, \ldots, v_s))$, where $U = \{u_1, \ldots, u_r\}$ and $V = \{v_1, ..., v_s\}$ are subsets of the set $X = \{x_1, \ldots, x_n\}$ of input variables, and g and h have fewer input variables than f. The theory uses generalized set systems (which, in turn, are generalized partitions), which we call blankets.

*This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant No. OGP0000871; most of the work was done while T. Luba was a Visiting Professor at the University of Waterloo.

Keywords: blanket, Boolean function, cube, decomposition, disjoint, "don't care," multiple-output, set system
AMS Classification Codes: 06E30 Boolean functions, 94C10 Switching theory, applications of Boolean algebra, Boolean functions.

1 Introduction

Decomposition is a fundamental problem in modern logic synthesis. Its goal is to break a logic circuit into a set of smaller interacting components. Such an implementation is desirable for a number of reasons. In the case of designs using field-programmable gate arrays (FPGAs), particularly those with look-up table structures [3], decomposition is a necessity, since FPGA cells can only accomodate functions with very few inputs and outputs. In PLA- and PLD-oriented designs, a decomposed circuit often leads to a smaller silicon area and shorter signal delays [7]. Consequently, decomposition methods are increasingly exploited in today's logic synthesis systems [22].

Mathematically, decomposition is the process of expressing a function of n variables as a function of functions of fewer variables [8]. In this paper, we consider a function $f(x_1, \ldots, x_n)$ to be decomposable if it can be expressed as $f(x_1, \ldots, x_n) = h(u_1, \ldots, u_r, g(v_1, \ldots, v_s))$, where $U = \{u_1, \ldots, u_r\}$ and $V = \{v_1, \ldots, v_s\}$ are proper subsets of the set $X = \{x_1, \ldots, x_n\}$ of input variables, and g and h have fewer input variables than f.

Numerous decomposition algorithms have been developed. Ashenhurst, in his fundamental paper [1], stated the disjoint decomposition theorem based on decomposition charts, where a decomposition is disjoint if $U \cap V = \emptyset$. Curtis extended Ashenhurst's results to multiple decompositions [8], of the form $f(x_1, \ldots, x_n) = h(u_1, \ldots, u_r, g_1(v_1, \ldots, v_s), \ldots, g_k(w_1, \ldots, w_t))$, where $U = \{u_1, \ldots, u_r\}$, $V = \{v_1, \ldots, v_s\}, \ldots, W = \{w_1, \ldots, w_t\}$ are proper subsets of X such that $U \cup V \cup \ldots \cup W = X$. The methods of Ashenhurst and Curtis were restricted to single-output Boolean functions represented by truth tables rearranged as charts. The use of charts for decomposition of logic networks is applicable only to functions with few variables. To remedy this, Roth and Karp used a more compact representation of a function in the form of a cover of the on-set and a cover of the off-set [21]. However, their method does not deal directly with multiple-output functions.

In the early 1980's, functional decomposition methods received less attention because of the rapid development of synthesis techniques for multi-level logic. A renewed interest in functional decomposition in recent years was caused by the introduction of field programmable gate arrays by Xilinx in 1986, and other companies (AT&T, Actel, Altera) in succeeding years.

There are several approaches to FPGA-based logic synthesis. The most common one relies on breaking the synthesis process into two phases: a technology-independent phase and a technology-mapping phase. The first phase generates a reduced technology-independent abstract logic network. For combinational logic this is a Boolean network, i.e., a directed acyclic graph $G(V, E)$, where each node $v \in V$ represents an arbitrarily complex single-output logic function, and each branch, an interconnection. The second phase maps the abstract logic network onto cells of a given FPGA, and performs technology-dependent optimizations taking the FPGA constraints into account. The architecture based on look-up tables (LUTs) is prevalent among many FPGA architectures. An LUT-based FPGA consists of an array of LUTs, each of which can implement any Boolean function with up to k (typically 4 or 5) inputs. A Boolean network can be directly realized by a one-to-one mapping between nodes and LUTs if every node in the network is feasible, i.e., has up to k input variables. Nodes having more inputs must be decomposed into subnetworks of smaller feasible nodes prior to the actual mapping. This is why decomposition algorithms are usually incorporated into a multilevel synthesis environment [6, 12], where decomposition is usually applied to multi-output functions which result from the node creation or node clustering process in a Boolean network. In methods using this approach, the node function to be decomposed is usually represented by a sum-of-products form, a truth table consisting of the ON-cover terms, or a BDD. Thus there is no need for decomposition of the incompletely specified multiple-output Boolean functions and usage of "don't cares" for optimization of the resulting circuit.

A different approach to FPGA-based technology mapping was proposed in [10, 11, 16, 17, 26], where functional decomposition is applied directly to the original function specification, in order to gain design freedom to construct directly a feasible network of LUTs optimized for a given FPGA. Moreover, the concept of parallel decomposition was introduced in [16] and effectively applied in the so-called balanced decomposition method [17], or decomposition using clusters of single-output functions to find good common subfunctions [11, 26]. Using input variable analysis of each single output of a multi-output function F, parallel decomposition separates F into two or more (multi-output) subfunctions, each of which has as inputs and outputs a subset of the original inputs and outputs of F. Although the decomposition program proposed in [16] uses a mix of serial and parallel decompositions, the crucial process in the whole mapping is serial decomposition.

Our work was motivated by the paper of Łuba and Selvaraj [16]. We formalize and generalize the model used there. An approach somewhat similar to ours has been recently developed independently by Shestakov. Since there are some significant differences, a discussion of this work is outside the scope of the present paper. We refer the interested reader to the literature [13, 14, 23, 24].

The contributions of this paper are as follows:

- We develop a mathematical theory of serial decompositions of Boolean functions using certain generalizations of set systems, which we call *blankets*.[1]
- The functions have multiple outputs, and are incompletely specified.
- The functions are represented in the *consistent-cube* notation, which corresponds to the *fr* format used, for example, in ESPRESSO [2]. Multiple-output functions are treated as single entities.
- Both the disjoint and non-disjoint cases are treated in a single theory.
- The decompositions are carried out using a cube calculus.

The remainder of the paper is structured as follows. Section 2 presents partially specified, multiple-output, Boolean functions and their representations in the *consistent cube* notation. In Section 3 we describe the notion of *separation* of a function, where we express $f(x_1, \ldots, x_n)$ in the form $h(u_1, \ldots, u_r, g(v_1, \ldots, v_s))$, but pay no attention to the number of arguments of functions g and h. In Section 4 we discuss two generalizations of partitions: blankets and set systems. The use of blankets for finding separations of Boolean functions is described in Section 5. The special case of disjoint separations, where the variable sets U and V are disjoint, is discussed in Section 6. In Section 7, we show that the theory developed so far, can also be expressed using set systems, but leads to more restrictive results. The use

[1]The term *rough partition* has also been used [16], but we prefer the shorter term *blanket*. Blankets could also be called covers; however, the word 'cover' has been used in a number of ways, so we prefer to coin a new word.

of cubes in the decomposition process is justified and illustrated in Section 8. Section 9 briefly describes a method for finding the blankets required for the key step in the decomposition algorithm. Section 10, discusses decompositions with constraints on the number of arguments of the components. Section 11 summarizes our results. The proofs of all the theorems are given in the Appendix.

Most of the ideas in this paper were first presented in a technical report [5], but were subsequently significantly revised and improved. Many of the ideas presented here can also be extended to multivalued functions. For some work in this direction see [4, 13, 14].

2 Boolean Functions in Consistent-Cube Notation

We use the convention that a *vector* of variables is denoted by a letter in lower case, and the corresponding *set* of variables, by the same letter in upper case. Thus, if $s = (s_1, \ldots, s_n)$ is a vector of variables, then $S = \{s_1, \ldots, s_n\}$ is the corresponding set. Also, if $S = \{s_1, \ldots, s_n\}$ is a set of variables explicitly represented in that order, then $s = (s_1, \ldots, s_n)$ is the corresponding vector. Suppose $T = \{s_{i_1} \ldots, s_{i_r}\}$ is a subset of S with $i_1 < i_2 < \ldots < i_r$, and $t = (s_{i_1} \ldots, s_{i_r})$ is the corresponding vector. If $b = (b_1, \ldots, b_n)$ is any vector of values, then b^t is the *projection* of b to the variables in T, i.e., $b^t = (b_{i_1}, \ldots, b_{i_r})$. We also use the convention that vectors of values are written without parentheses and commas. For example, suppose $S = \{s_1, s_2, s_3, s_4\}$ and $s = (s_1, s_2, s_3, s_4)$. If $t = (s_1, s_3, s_4)$ and $b = (4, 1, 2, 0)$, then $b^t = (4, 2, 0)$. In simplified notation without commas and parentheses, $b = 4120$ and $b^t = 420$.

We are concerned with incompletely specified multiple-output Boolean functions represented by *function matrices,* as explained below.

Example 1 Table 1 represents a function with four inputs $x_1, \ldots, x_4$ and two outputs y_1, y_2. For the mathematical theory, it is convenient to treat the values assigned to variables as sets. Thus a variable can be assigned $\{0\}$, $\{1\}$, or $\{0, 1\}$, the interpretation being that it has the value 0, 1, or "don't care," respectively. For brevity, we omit the curly brackets from examples, and denote $\{0,1\}$ by Φ. For example, formally, Row 1 should be $(\{0\},\{0\},\{0,1\},\{0\},\{1\},\{1\})$, but is written $00\Phi011$. $\square$

Table 1: Matrix notation F for function f.

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

We now introduce a number of concepts using the example above. Table 1 is a matrix F with 7 rows and 6 columns. The first four components of each row t are its *input projection* t^x, and its last two components are its *output projection* t^y. If t is a row of F, we write $t \in F$. For $t \in F$, we say that input *minterm* (binary vector) b is *involved* in t if the input projection t^x of t contains b, where by containment we mean component-by-component set containment. Let the set of all rows of F involving minterm b be

$$F_{x \supseteq b} = \{t \in F \mid t^x \supseteq b\}. \tag{1}$$

For example, consider Row 4, which is $\Phi\Phi110\Phi$. The input projection of this row is $\Phi\Phi11$, and its output projection is 0Φ. The set of input minterms involved in Row 4 is

$$\{0011, 0111, 1011, 1111\},$$

since $\Phi\Phi11 \supseteq 0011$, etc. We have $F_{x\supseteq 0000} = \{1,3\}$, $F_{x\supseteq 0001} = \{3,7\}$, $F_{x\supseteq 0010} = \{1\}$, $F_{x\supseteq 0011} = \{4\}$, $F_{x\supseteq 0100} = \emptyset$, $F_{x\supseteq 0101} = \{6,7\}$, $F_{x\supseteq 0110} = \{5\}$, $F_{x\supseteq 0111} = \{4,6\}$, $F_{x\supseteq 1000} = \{2,3\}$, $F_{x\supseteq 1001} = \{3\}$, $F_{x\supseteq 1010} = \{2\}$, $F_{x\supseteq 1011} = \{4\}$, $F_{x\supseteq 1100} = \emptyset$, $F_{x\supseteq 1101} = \{6\}$, $F_{x\supseteq 1110} = \{5\}$, $F_{x\supseteq 1111} = \{4,6\}$.

The matrices we consider are *consistent* in the following sense. Consider any set T of rows. If the input projections of the rows in T contain a common minterm, then the output projections of that set of rows also contain a common minterm. In symbols,

$$\bigcap_{t\in T} t^x \neq \emptyset \text{ implies } \bigcap_{t\in T} t^y \neq \emptyset, \tag{2}$$

where intersection of vectors is component-by-component intersection. For example, minterm $(x_1, x_2, x_3, x_4) = 0000$ is involved in Rows 1 and 3, the input projections of which are $00\Phi0$ and $\Phi00\Phi$. Consistency holds for this set of rows, since the output projections of these rows (11 and 1Φ) contain minterm $(y_1, y_2) = 11$. We can check the consistency of a function matrix by considering all the maximal sets of rows containing a common input minterm. Clearly, if consistency is satisfied for a set of rows, then it also holds for any subset of this set of rows.

In our example, we have the following set of maximal subsets (which we call *blocks*) of the form $F_{x \supseteq b}$:

$$\sigma_x = \{\{1,3\}, \{3,7\}, \{6,7\}, \{5\}, \{4,6\}, \{2,3\}\}.$$

We can also find the sets of rows containing common output minterms. Let c be an output minterm; then

$$F_{y \supseteq c} = \{t \in F \mid t^y \supseteq c\}. \tag{3}$$

For Table 1 we have $F_{y \supseteq 00} = \{4,5\}$, $F_{y \supseteq 01} = \{4,6\}$, $F_{y \supseteq 10} = \{2,3,7\}$, $F_{y \supseteq 11} = \{1,3,6,7\}$. The set of maximal subsets of the form $F_{y \supseteq c}$ is therefore

$$\sigma_y = \{\{4,5\}, \{4,6\}, \{2,3,7\}, \{1,3,6,7\}\}.$$

The reader can verify that the consistency condition can now be restated as follows:

$$\textit{Every block of } \sigma_x \textit{ must be contained in some block of } \sigma_y. \tag{4}$$

This holds in our example, and Table 1 is indeed consistent.

Proposition 1 *A set T of k rows of a function matrix is consistent iff every pair or rows is consistent.*

Proof: Suppose rows u and v of T are not consistent. Then $u^y \cap v^y = \emptyset$. Hence, $\bigcap_{t \in T} t^y = \emptyset$, and T cannot be consistent. This shows that, if T is consistent, then every pair in T is also consistent.

Conversely, suppose T is not consistent, but all pairs of rows are consistent. Then there is some minterm b contained in t^x for every t in T, but $\bigcap_{t \in T} t^y = \emptyset$. Consider any component y_i. If the entry in position y_i is Φ or 0 in each row of T, then the intersection of all these entries is nonempty.

Similarly, the intersection is nonempty, if each entry is either 1 or Φ. Since the intersection $\bigcap_{t \in T} t^y$ *is* empty, there must exist some component, say y_i, and two rows, say u and v, such that $u_i, v_i \in \{0,1\}$, and $u_i \neq v_i$. But then the pair $\{u,v\}$ is inconsistent, which is a contradiction. □

Since there are $k(k-1)/2$ pairs of rows in T, consistency can be checked efficiently.

A different approach would be to check for *inconsistency* as follows. Consider a pair of rows, and determine whether the output projections of these rows have a common minterm. If they do, then consistency cannot be violated by these two rows. If they don't, examine the input projections. If they have no minterms in common, then consistency cannot be violated. Otherwise, we have found a violation, the matrix is inconsistent, and no further chacking is required.[2]

Table 2: Truth table of f.

x_1	x_2	x_3	x_4	y_1	y_2
0	0	0	0	1	1
0	0	0	1	1	Φ
0	0	1	0	1	1
0	0	1	1	0	Φ
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	Φ
1	0	1	0	1	0
1	0	1	1	0	Φ
1	1	0	1	Φ	1
1	1	1	0	0	0
1	1	1	1	0	1

The matrix of Table 1 is a compact notation for function f of Table 2. Here, each input takes on a binary value 0 or 1; each output, however, may

[2]The authors thank the anonymous referee for this suggestion.

be 0, 1, or Φ. The value of f is determined as follows. We say that an input minterm b is *relevant* to F if b is involved in some row of F. Recall that $F_{x \supseteq b}$ is the set of all rows of F that involve minterm b; if b is relevant to F, then $F_{x \supseteq b}$ is nonempty. The value $y_i(b)$ of output y_i for b is the intersection of the sets of values appearing in the ith position of the output projections of the rows in $F_{x \supseteq b}$. Since we assume that F is consistent, this intersection is never empty. In our vector notation, we have

$$f(b) = \bigcap_{t \in F_{x \supseteq b}} t^y. \tag{5}$$

In case b is not relevant to F we define the output to be the vector of don't cares.

In our example, the set of rows containing minterm 0000 is $\{1, 3\}$. Hence, the output vector assigned to minterm 0000 is 11, which is the common value in the output projections 11 and 1Φ of Rows 1 and 3. Minterm 0001 is contained only in Row 7; hence, the output vector for that minterm is 1Φ as specified by Row 7. Minterm 0100 does not appear in any row of F; hence, both outputs are don't cares here. By convention, we omit such rows from the table of f. The remaining entries are similarly obtained.

We now generalize the terminology and notation introduced above. Let $D = \{\{0\}, \{1\}, \{0, 1\}\}$. A *function matrix* F is a matrix with h rows and $n + m$ columns with entries from D, where $n > 0$ is the size of the vector $x = (x_1, \ldots, x_n)$ of input variables, $m > 0$ is the size of the vector $y = (y_1, \ldots, y_m)$ of output variables, and $h \geq 0$.

If $t = (t_1, \ldots, t_n, t_{n+1}, \ldots, t_{n+m})$ is any row of F, the *input projection* of t is $t^x = (t_1, \ldots, t_n)$, and the *output projection* is $t^y = (t_{n+1}, \ldots, t_{n+m})$. Let $\Sigma = \{\{0\}, \{1\}\}$. If $V \subseteq X$ has s elements and $d \in \Sigma^s$, then d is *involved* in row t if $t^v \supseteq d$. Also d is *relevant* to F if it is involved in some row of F.

A function matrix is *consistent* if (2) holds. *From now on we consider only consistent function matrices.* Each row of a function matrix is commonly called a "cube." We also refer to a consistent function matrix as a "consistent-cube notation" for a Boolean function f. Each such matrix F specifies a function f from Σ^n to D^m as defined by (5).

3 Separations of Functions

Our main goal is as follows. Given a Boolean function $f(x)$ in consistent-cube notation, decompose it in the form $h(u, g(v))$, where x is the vector of input variables, X is the corresponding set, U and V are two subsets of X satisfying $U \cup V = X$, u and v are the corresponding vectors, and h and g are Boolean functions. See Fig. 1. We refer to such decompositions as *serial decompositions* [16]. In this paper we deal only with serial decompositions; consequently, we refer to them as just *decompositions*, if there is no danger of ambiguity.

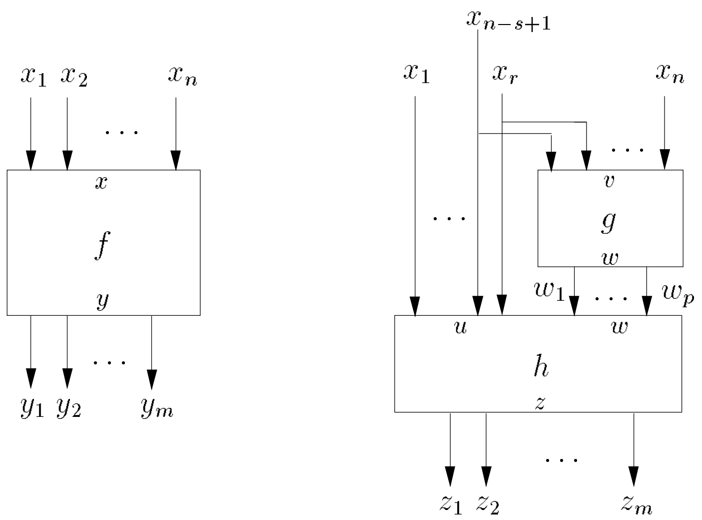

Figure 1: Illustrating decomposition.

One can view $f(x)$ as a specification of a function and $h(u, g(v))$ as its implementation. For any input minterm b of f, the inputs to g are b^v, and the inputs to h are $(b^u, g(b^v))$. For $h(u, g(v))$ to be a decomposition of $f(x)$, we require two conditions:

1. $h(u, g(v))$ should produce for each of its outputs the same value as $f(x)$, whenever $f(x)$ is 0 or 1. If $f(x) = \Phi$, we accept Φ or either binary value in h. Thus, for every minterm b relevant to f we should have

$$f(b) \supseteq h(b^u, g(b^v)). \tag{6}$$

We require that $g(b^v)$ be in Σ^p for all relevant minterms b. Then we guarantee that the inputs to h are well defined for all relevant minterms. This is needed in our method, since the value of a function is defined only for minterms.

2. Functions g and h should be "simpler" than f.

In Sections 3–9, we consider only the first condition; the second will be added in Section 10. A pair (g, h) of functions is called a *separation* if it satisfies the first condition.

Example 2 Consider function f represented by matrix F of Table 1, and functions g and h represented by matrices G and H of Table 3. Function g has inputs x_2, x_3, and x_4 and output w. Function h has inputs x_1, x_4, and w and outputs z_1 and z_2. The reader can verify that both matrices G and H are consistent. We will show that (g, h) is a separation of f.

Table 3: Matrices G and H for Example 2.

Matrix G

x_2	x_3	x_4	w
0	Φ	0	0
Φ	0	1	0
Φ	1	1	1
1	1	Φ	1

Matrix H

x_1	x_4	w	z_1	z_2
Φ	Φ	0	1	Φ
Φ	Φ	1	0	Φ
Φ	1	Φ	Φ	1
0	Φ	0	Φ	1
Φ	0	1	Φ	0
1	0	Φ	Φ	0

Consider minterm 0000 of f. According to Table 2, $f(0000) = 11$. The bits (x_2, x_3, x_4) supplied to function g are 000 for this minterm. In the table of G, 000 appears only in Row 1; hence $w = g(000) = 0$. The bits supplied to h are therefore $(x_1, x_4, w) = 000$. In the table of H, 000 appears in Rows 1 and 4; hence $h(000) = 1\Phi \cap \Phi 1 = 11$. This agrees with the value of f; hence, Condition 1 is satisfied for minterm 0000. Similarly, $f(0001) = 1\Phi$, $g(001) = 0$, and $h(010) = 11$. Since $1\Phi \supseteq 11$, Condition 1 is satisfied. For minterm 0100, Condition 1 is satisfied vacuously, since this minterm is not relevant. The remaining entries are verified in a similar way. □

The notions introduced above are now formalized. Let $X = \{x_1, \ldots, x_n\}$ be the set of input variables of a function $f(x_1, \ldots, x_n)$. Let U and V be two subsets of X such that $U \cup V = X$. Usually, for convenience and without loss of generality, we assume that variables $x_1, \ldots, x_n$ have been relabeled in such a way that $u = (x_1, \ldots, x_r)$ and $v = (x_{n-s+1}, \ldots, x_n)$. Then for any n-tuple x, the first r components are denoted by x^u, and the last s components, by x^v. See Fig. 1.

Definition 1 Let F be a consistent function matrix, with $n > 0$ inputs and $m > 0$ outputs, and let (U, V) be as above. Let G be a consistent function matrix with s inputs and p outputs, and let H be a consistent function matrix with $r + p$ inputs and $m > 0$ outputs. The pair (G, H) is a *separation* of F *with respect to* (U, V), if, for every minterm b relevant to F, $g(b^v)$ is in Σ^p, and

$$f(b) \supseteq h(b^u, g(b^v)).$$

In case (G, H) is a separation of F, we also say that (g, h) is a separation of f. □

The condition that $g(b^v)$ is in Σ^p, i.e., that the outputs of g should be specified for all relevant inputs, ensures that the inputs to function h will be well defined for all relevant minterms.

It is clear that every consistent function matrix F always has at least one separation, because we can choose $p = n - r$, let g be the trivial function that just passes its last $n - r$ input values to its $p = n - r$ outputs, and let h be f. Then $h(b^u, g(b^v)) = f(b)$, and (6) is satisfied. However, it is not at all obvious how to find nontrivial separations. In the next section we introduce the tools that allow us to generate many separations.

4 Blankets

Given a finite set S, by a *cover* of S we mean a collection of subsets of S, called *blocks*, whose union is S. For example, if $S = \{1, 2, 3, 4\}$, then $\pi = \{\{1, 2\}, \{3, 4\}\}$, $\sigma = \{\{1, 2, 3\}, \{1, 4\}\}$, $\beta = \{\{1, 2, 3\}, \{2, 3\}, \{4\}\}$, and $\rho = \{\{1, 2\}, \{1, 2\}, \{3, 4\}\}$ are all covers of S. The last type permits repeated subsets. *We will not consider any covers with repeated subsets in this paper.*

A *blanket* on a set S is a cover $\beta = \{B_1, \ldots, B_k\}$ of nonempty and distinct subsets of S, called *blocks*, whose union is S. For example, if $S = \{1, 2, 3, 4\}$, then β above is a blanket on S. To improve the notation, we often write $\beta = \{\overline{1,2,3};\ \overline{2,3};\ \overline{4}\}$, instead. Also, when it is possible to avoid reference to the number k of blocks in a blanket $\beta = \{B_1, \ldots, B_k\}$, we simply write $\beta = \{B_i\}$.

A *set system* [9] on a set S is a blanket $\{B_i\}$ in which the blocks are maximal in the sense that $B_i \subseteq B_j$ implies $i = j$. A *partition* on a set S is a set system $\{B_i\}$, where the blocks are disjoint, that is, $B_i \cap B_j = \emptyset$, if $i \neq j$. For example, σ above is a set system on S, and π is a partition on S.

Define the "nonempty" operator *ne* as follows. For any set $\{S_i\}$ of subsets of a set S, $ne\{S_i\}$ is the set $\{S_i\}$ with the empty subset removed, if it was originally present. The *product* $\beta{*}\beta'$ of two blankets is defined as follows:

$$\beta{*}\beta' = ne\{B_i \cap B_j \mid B_i \in \beta \text{ and } B_j \in \beta'\}. \tag{7}$$

If β and β' are blankets on S, we write $\beta \leq \beta'$ if each block B_i of β is contained in a block B_j of β'.

The following examples illustrate the product of blankets and the $\leq$ relation. Here $S = \{1, \ldots, 5\}$ for the first two examples, and $S = \{1, 2, 3\}$ for the last two.

$$\{\overline{1,2,3};\ \overline{3,4,5}\}{*}\{\overline{1,3,4};\ \overline{1,5};\ \overline{2,3,4}\} = \{\overline{1,3};\ \overline{1};\ \overline{2,3};\ \overline{3,4};\ \overline{5}\},$$

$$\{\overline{1,2,3};\ \overline{3,4,5}\}{*}\{\overline{1,2,3};\ \overline{3,4,5}\} = \{\overline{1,2,3};\ \overline{3};\ \overline{3,4,5}\},$$
$$\{\overline{1};\ \overline{2};\ \overline{3}\} \leq \{\overline{1,2};\ \overline{2,3}\},$$
$$\{\overline{1,2};\ \overline{2,3}\} \leq \{\overline{1,2};\ \overline{2,3};\ \overline{1,3}\}.$$

Intuitively, blankets on a set $S = \{s_1, \ldots, s_n\}$ carry some information about elements of S. For example, if we know that we are dealing with the first block of the blanket $\{\overline{1,2};\ \overline{2,3};\ \overline{1,3}\}$, then we know that we are dealing with elements 1 and 2 of the set $\{1, 2, 3\}$. The n-block blanket $\{\{1\}, \{2\}, \ldots, \{n\}\}$ carries the maximum information, for if we know a block of this blanket, we know precisely which element is involved. The one-block blanket $\{\{1, 2, \ldots, n\}\}$ carries zero information, since no elements are excluded if we know the block. In general, if $\beta \leq \beta'$, then β carries at least as much information as β'.

If $\beta = \{B_i\}$ is any blanket on S, then

$$max\,\beta = \{B \in \beta \mid B = B_i \text{ for some } i, \text{ and } B \subseteq B_j \text{ implies } B_j = B\}. \tag{8}$$

Note that *max* maps blankets to set systems by removing blocks contained in other blocks. The *product* $\sigma \circ \sigma'$ of two set systems [9] is defined as follows:

$$\sigma \circ \sigma' = max(\sigma * \sigma').$$

For example,

$$\{\overline{1,2,3};\ \overline{3,4,5}\} \circ \{\overline{1,3,4};\ \overline{1,5};\ \overline{2,3,4}\} = \{\overline{1,3};\ \overline{2,3};\ \overline{3,4};\ \overline{5}\}.$$

We will be using blankets on the set of rows of a consistent function matrix F. Assume that the rows of F are numbered $1, \ldots, h$. For simplicity, we denote blankets on the set of *rows* of F by blankets on the set $\{1, \ldots, h\}$ of *row indices.* Suppose V is an r-element subset of the set $X \cup Y$ of input and output variables. Blanket β_v is defined by

$$\beta_v = ne\{F_{v \supseteq d} \mid d \in \Sigma^r\}. \tag{9}$$

For the function of Table 1, if we let $V = Y$, we obtain the *output blanket*

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

where we have indicated the values of the variables y_1, y_2 corresponding to each block. This blanket happens to be a set system and is equal to the σ_y that we found earlier in connection with checking the consistency of F. Similarly, if $V = X$, then

$$\beta_x = \{\overset{0000}{\overline{1,3}};\ \overset{0001}{\overline{3,7}};\ \overset{0010}{\overline{1}}\ ;\ \overset{\Phi 011}{\overline{4}}\ ;\ \overset{0101}{\overline{6,7}};\ \overset{\Phi 110}{\overline{5}}\ ;\ \overset{\Phi 111}{\overline{4,6}}\,;\ \overset{1000}{\overline{2,3}};\ \overset{1001}{\overline{3}}\ ;\ \overset{1010}{\overline{2}}\ ;\ \overset{1101}{\overline{6}}\ \}.$$

Here, we have used the shorthand that $\Phi 011$ represents both 0011 and 1011, etc. Note that there is no block corresponding to 0100 and 1100, since these combinations are not included in any row. One verifies that the σ_x that we found for F in Section 2 is

$$\sigma_x = max\,\beta_x.$$

If $U = \{x_1, x_4\}$ and $V = \{x_2, x_3, x_4\}$, then

$$\beta_u = \{\overset{00}{\overline{1,3,5}};\ \overset{01}{\overline{3,4,6,7}};\ \overset{10}{\overline{2,3,5}};\ \overset{11}{\overline{3,4,6}}\} \text{ and } \beta_v = \{\overset{000}{\overline{1,2,3}};\ \overset{001}{\overline{3,7}};\ \overset{010}{\overline{1,2}};\ \overset{011}{\overline{4}}\ ;\ \overset{101}{\overline{6,7}};\ \overset{110}{\overline{5}}\ ;\ \overset{111}{\overline{4,6}}\}.$$

The following observation relates the notion of consistency of a function to a relation on its blankets.

Proposition 2 *Let F be a function matrix; then the conditions below are equivalent: (a) F is consistent, (b) $\beta_x \leq \beta_y$, (c) $\sigma_x \leq \sigma_y$.*

5 Finding Separations Using Blankets

Consider function F of Table 1. Let $V = \{x_2, x_3, x_4\}$, and let G be the function of Table 3. Row 1 of F has the vectors 000 and 010 for x_2, x_3, x_4. From Table 3 we see that $g(000) = g(010) = 0$. Thus Row 1 can only produce a 0 for w. In fact, Rows 1, 2, 3, and 7 can only produce 0, Rows 4 and 5 can only produce 1, and Row 6 can have both 0 and 1. The set of rows that can produce a 0 for w is therefore $F_{w=0} = \{1, 2, 3, 6, 7\}$, and the set of rows that can produce a 1 is $F_{w=1} = \{4, 5, 6\}$. In this way we define a blanket which we call γ_g. Here $\gamma_g = \{\overline{1, 2, 3, 6, 7}; \overline{4, 5, 6}\}$.

In general, suppose we are given a consistent function matrix $F(x)$ with n inputs, a set $V \subseteq X$ of cardinality s, and a consistent function matrix $G(v)$ with s inputs and p outputs. Furthermore, assume that $g(b^v) \in \Sigma^p$ for all minterms $b \in \Sigma^n$ that are relevant to F. Let $e \in \Sigma^p$, and let

$$F_{w=e} = \{t \in F \mid \exists d \in \Sigma^s : t^v \supseteq d \text{ and } g(d) = e\} \tag{10}$$

be the set of all rows of F that can produce the value e for w. Note that $F_{w=e}$ is always a union of blocks of the form $F_{v \supseteq d}$. Blanket γ_g is defined by

$$\gamma_g = ne\{F_{w=e}\}. \tag{11}$$

Theorem 1 *Let $F(x)$ be a consistent function matrix and let (U, V) be a pair of subsets of X satisfying $U \cup V = X$. For every blanket β_g satisfying the conditions*

1. *$\beta_v \leq \beta_g$, and*

2. *$\beta_u * \beta_g \leq \beta_y$,*

there is a separation (G, H) of F with respect to (U, V) such that

- *$\gamma_g \leq \beta_g$.*

Proof: The proof is constructive, and is given in the appendix. □

This theorem is a generalization of a result by Łuba and Selvaraj [16], who limited their attention to decompositions obtainable with partitions. Also, the model used in [16] is rather informal and incomplete, and the proof of the theorem is only sketched. Moreover, it is claimed incorrectly that the converse of the result also holds. We show in Example 4 that the converse is false in general, although it does hold in the case of disjoint decompositions, as we prove in Theorem 2.

Intuitively, Condition 1 states that the information present in the inputs of vector v should be sufficient to compute the outputs of function g. Condition 2 requires that the information contained in input vector u together with the information contained in the outputs of g should be sufficient to reproduce output y. The condition $\gamma_g \leq \beta_g$ can be interpreted as stating that function g is derived from blanket β_g. The theorem would not make sense if this condition were omitted, because one might be able to find a decomposition that has nothing to do with the given β_g. Since we are going to use β_g to generate g, this condition is needed. We are not aware of any special properties of decompositions in which $\gamma_g = \beta_g$.

Example 3 In this example we illustrate the construction in the proof of the theorem. We will find a separation of the function of Table 1 with $U = \{x_1, x_4\}$, and $V = \{x_2, x_3, x_4\}$, using a given blanket $\beta_g = \{\overline{1,2,3,6,7}; \overline{4,5,6}\}$. We verify that β_g satisfies the two conditions of the theorem. We now show the steps of the construction.

Table 4: Function g for Example 3.

x_2	x_3	x_4	Block of β_v	Block of β_g	w
0	0	0	$\{1,2,3\}$	$\{1,2,3,6,7\}$	0
0	0	1	$\{3,7\}$	$\{1,2,3,6,7\}$	0
0	1	0	$\{1,2\}$	$\{1,2,3,6,7\}$	0
0	1	1	$\{4\}$	$\{4,5,6\}$	1
1	0	1	$\{6,7\}$	$\{1,2,3,6,7\}$	0
1	1	0	$\{5\}$	$\{4,5,6\}$	1
1	1	1	$\{4,6\}$	$\{4,5,6\}$	1

1. Suppose the given blanket β_g has q blocks. We encode the blocks of β_g by a set of $p \geq \lceil \log_2 q \rceil$ binary variables in such a way that each block corresponds to a unique code word. Here $q = 2$; we choose $p = 1$ and the following encoding:

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

2. In Table 4, we show in the first three columns all the minterms from V that are relevant to the function G being constructed. In the fourth column, we assign to each minterm d the block of β_v corresponding to that minterm, that is the block $F_{v \supseteq d}$. For example, $F_{v \supseteq 000} = \{1, 2, 3\}$.

3. In the fifth column we show a block of β_g containing each block of β_v. The existence of such a block is guaranteed by the first condition of the theorem. If there is more than one such block, we select one arbitrarily. In this case there is no choice. For example, block $\{3, 7\}$ of β_v is contained in block $\{1, 2, 3, 6, 7\}$ of β_g.

4. In the last column we show the output value corresponding to each block of β_g as chosen in Step 1.

The first three columns of Table 4 together with the last column define g. The reader may verify that this is the same function as that defined by Table 3. Note that $\gamma_g = \beta_g$; hence $\gamma_g \leq \beta_g$, as required.

Next, we construct function h. The steps of the algorithm are shown in Table 5.

1. For each minterm b relevant to F, construct the corresponding minterm $(b^u, g(b^v))$ of h. For example, $b = 1101$ is relevant; so $b^u = 11$, and $g(b^v) = g(101) = 0$ from Table 4. Altogether, $(b^u, g(b^v)) = 110$ is a minterm relevant to H. This procedure produces the first three columns of Table 5.

2. Each minterm of h found in Step 1 is of the form $(x_1, x_4, w) = (c_1, c_2, d)$. For each such minterm, list the corresponding block $F_{u \supseteq c}$ of β_u, where $c = (c_1, c_2)$, in Column 4 of the table. For example, in the fourth row, 01 corresponds to $\{3, 4, 6, 7\}$.

3. For each value of w in Column 3, list the corresponding block of β_g in Column 5. For example, $w = 1$ corresponds to $\{4, 5, 6\}$.

4. Intersect each block from Column 4 with the block from Column 5, and list the result in Column 6. For example, in Row 1, $\{1, 3, 5\} \cap \{1, 2, 3, 6, 7\} = \{1, 3\}$. Thus, Column 6 has blocks of $\beta_u * \beta_g$. By the second condition of the theorem, each such block is contained in some block of β_y. Because of this and because F is consistent, the

intersections of the output vectors corresponding to the blocks in Column 6 are nonempty. Assign these intersections as outputs of h in Columns 7 and 8. For example, for the last row, we have the intersection $0\Phi \cap \Phi 1 = 01$ in Columns 7 and 8.

The table consisting of the three input columns and two output columns defines h. The reader can verify that this h is the same as the function defined by Table 3. □

Table 5: Defining function h.

x_1	x_4	w	β_u	β_g	$\beta_u * \beta_g$	z_1	z_2
0	0	0	$\{1,3,5\}$	$\{1,2,3,6,7\}$	$\{1,3\}$	1	1
0	0	1	$\{1,3,5\}$	$\{4,5,6\}$	$\{5\}$	0	0
0	1	0	$\{3,4,6,7\}$	$\{1,2,3,6,7\}$	$\{3,6,7\}$	1	1
0	1	1	$\{3,4,6,7\}$	$\{4,5,6\}$	$\{4,6\}$	0	1
1	0	0	$\{2,3,5\}$	$\{1,2,3,6,7\}$	$\{2,3\}$	1	0
1	0	1	$\{2,3,5\}$	$\{4,5,6\}$	$\{5\}$	0	0
1	1	0	$\{3,4,6\}$	$\{1,2,3,6,7\}$	$\{3,6\}$	1	1
1	1	1	$\{3,4,6\}$	$\{4,5,6\}$	$\{4,6\}$	0	1

We have seen from Theorem 1 that, if there is a blanket satisfying the conditions of the theorem, then there is a separation. In general, the converse of this theorem is false, as is demonstrated by Example 4 below. We show in the next section, however, that the result is true in the special case of *disjoint* separations, for which $U \cap V = \emptyset$.

Example 4 Consider matrix F of Table 6 and matrices G and H of Ta-

Table 6: Matrix F for Example 4.

	x_1	x_2	x_3	x_4	y
1	0	Φ	0	1	0
2	0	Φ	1	1	1

Table 7: Matrices G and H for Example 4.

Matrix G

x_2	x_3	x_4	w
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

Matrix H

x_1	x_2	w	z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0

ble 7. Let $U = \{x_1, x_2\}$ and $V = \{x_2, x_3, x_4\}$. One verifies that (G, H) is a separation of F with respect to (U, V). Note, however, that $\beta_u = \{\overline{1,2}\}$, $\beta_v = \{\overline{1};\ \overline{2}\}$, and $\beta_y = \{\overline{1};\ \overline{2}\}$. The blanket constructed from G is $\gamma_g = \{\overline{1,2}\}$. For $\gamma_g \leq \beta_g$ we need to have $\beta_g = \{\overline{1,2}\}$. But this blanket fails to satisfy the condition $\beta_u * \beta_g \leq \beta_y$. Consequently, the converse of the theorem does not hold here.

Note that blanket $\beta_g = \{\overline{1};\ \overline{2}\}$, satisfies both conditions of the theorem; however, it fails the condition $\gamma_g \leq \beta_g$. $\square$

Example 5 This example illustrates the choices that might exist in the decomposition process. We decompose the matrix of Table 8 with $U = \{x_1\}$ and $V = \{x_2, x_3, x_4\}$. The blankets for this decomposition are: $\beta_u = \{\overset{0}{\overline{1,3,4,5,6}};\ \overset{1}{\overline{2,3,4,5}}\}$, $\beta_v = \{\overset{000}{\overline{1,2,3,6}};\ \overset{001}{\overline{3,6}};\ \overset{010}{\overline{1,2}};\ \overset{011}{\overline{4}};\ \overset{100}{\overline{6}};\ \overset{101}{\overline{5,6}};\ \overset{111}{\overline{4,5}}\}$, and $\beta_y = \{\overset{00}{\overline{4}};\ \overset{01}{\overline{4,5}};\ \overset{10}{\overline{1,2,3,6}};\ \overset{11}{\overline{1,3,5,6}}\}$. Suppose $\beta_g = \{\overset{00}{\overline{1,2,3,6}};\ \overset{01}{\overline{5,6}};\ \overset{10}{\overline{4,5}};\ \overset{11}{\overline{3,6}}\}$. One verifies that the conditions of Theorem 1 are satisfied.

In Table 9 we show the minterms relevant to g and the corresponding blocks of β_v. There are several choices for covering blocks of β_v with blocks of β_g. For example, block $\{6\}$ of β_v can be covered by the first, second, or fourth block of β_g. This results in three choices for the output assigned to 100. The last three columns of Table 9 show three different assignments, resulting in three different functions g, g' and g''. Calculating blanket γ_g corresponding to each of these functions, we find: $\gamma_g = \{\overset{00}{\overline{1,2,3,6}};\ \overset{01}{\overline{5,6}};\ \overset{10}{\overline{4,5}};\ \overset{11}{\overline{6}}\}$, $\gamma_{g'} = \{\overset{00}{\overline{1,2,3,6}};\ \overset{01}{\overline{5,6}};\ \overset{10}{\overline{4,5}}\}$, $\gamma_{g''} = \{\overset{00}{\overline{1,2,3,6}};\ \overset{01}{\overline{5,6}};\ \overset{10}{\overline{4,5}};\ \overset{11}{\overline{3,6}}\}$. In each case, the condition $\gamma_g \leq \beta_g$ is satisfied, and in one case we have equality. The

Table 8: Matrix F for Example 5.

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

assignment involving g' shows that we could have used a set system, namely $\gamma_{g'}$, to find this decomposition. This has only 3 blocks, as compared to the 4 blocks of β_g; however, the number of outputs remains two. □

Table 9: Choices for g in Example 5.

x_2	x_3	x_4	Block of β_v	g	g'	g''
0	0	0	$\{1,2,3,6\}$	00	00	00
0	0	1	$\{3,6\}$	00	00	11
0	1	0	$\{1,2\}$	00	00	00
0	1	1	$\{4\}$	10	10	10
1	0	0	$\{6\}$	11	00	11
1	0	1	$\{5,6\}$	01	01	01
1	1	1	$\{4,5\}$	10	10	10

6 Blankets from Disjoint Separations

A separation of F with respect to (U, V) is *disjoint* if $U \cap V = \emptyset$.

Theorem 2 *Let $F(x)$ be a consistent function matrix and let (U, V) be a pair of disjoint subsets of X satisfying $U \cup V = X$. For every separation (G, H) of F with respect to (U, V) blanket γ_g satisfies the conditions*

- $\beta_v \leq \gamma_g$, *and*

- $\beta_u * \gamma_g \leq \beta_y$.

Proof: The proof is constructive, and it is given in the appendix. $\square$

Example 6 Matrix F of Table 1 is decomposed with respect to $U = \{x_1\}$ and $V = \{x_2, x_3, x_4\}$. Here

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

and, as before,

$$\beta_v = \{\overset{000}{\overline{1,2,3}};\ \overset{001}{\overline{3,7}};\ \overset{010}{\overline{1,2}};\ \overset{011}{\overline{4}};\ \overset{101}{\overline{6,7}};\ \overset{110}{\overline{5}};\ \overset{111}{\overline{4,6}}\},$$

and

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

Table 10: Matrices G and H for Example 6.

Matrix G

x_2	x_3	x_4	w_1	w_1
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	1

Matrix H

x_1	x_2	w	z_1	z_2
0	0	0	1	1
0	0	1	0	0
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	Φ	1
1	1	1	0	1

Table 10 shows a matrix G, such that $g(b^v)$ is specified whenever b is relevant to F, and a matrix H. One verifies that (G, H) is a separation of F. Blanket γ_g for this case is $\{\overline{1,2,3,7};\ \overline{5};\ \overline{6,7};\ \overline{4,6}\}$. It is easily verified that this blanket satisfies the two conditions of the theorem. $\square$

7 Finding Separations Using Set Systems

The main results about the existence of separations and blankets remain true if we use set systems instead of blankets.

Theorem 3 *Let $F(x)$ be a consistent function matrix and let (U, V) be a pair of subsets of X satisfying $U \cup V = X$. For every set system σ_g satisfying*

1. $\sigma_v \leq \sigma_g$, *and*
2. $\sigma_u \circ \sigma_g \leq \sigma_y$,

there is a separation (G, H) of F with respect to (U, V) such that

- $max\,\gamma_g \leq \sigma_g$,

where $\sigma_u = max\,\beta_u$, $\sigma_v = max\,\beta_v$, and $\sigma_y = max\,\beta_y$.

Proof: The proof is given in the appendix, where we show that every solution constructed using set systems throughout is among the solutions constructed using blankets. □

Example 7 Consider how the use of set systems would change Example 5. Blocks $\{3, 6\}$, $\{1, 2\}$, and $\{6\}$ of β_v would be replaced by the maximal block $\{1, 2, 3, 6\}$, and block $\{4\}$ would be replaced by $\{4, 5\}$. Then only function g' would be found.

For function h, consider Example 3. Here, we would use $\{3, 4, 6, 7\}$ as the block of β_u in Rows 7 and 8. In Row 7 the block of $\beta_u * \beta_g$ would be $\{3, 6, 7\}$; the output for that row would not change in this case. □

The use of set systems restricts the solution space. In general, in constructing g we have more choice with blankets. Recall that we are looking for a blanket β_g satisfying $\beta_v \leq \beta_g$. A given block of β_v may be contained in more blocks of β_g, if β_g is a blanket rather that a set system. Also, in constructing h, the larger the block we use, the more constraints we impose on the value of h. Thus, blankets are less restrictive. On the other hand, if one wishes to reduce the size of the solution space, set systems are preferred. The decision is left to the implementer.

Theorem 2 also has its counterpart in terms of set systems.

Theorem 4 *Let $F(x)$ be a consistent function matrix and let (U, V) be a pair of disjoint subsets of X satisfying $U \cup V = X$. For every separation (G, H) of F with respect to (U, V), the set system $\sigma_g = max\gamma_g$, satisfies the conditions*

- $\sigma_v \leq \sigma_g$, *and*
- $\sigma_u \circ \sigma_g \leq \sigma_y$.

Proof: The proof is given in the appendix. □

The following summarizes our reasons for using blankets instead of only set systems:

- Blankets arise naturally, since each minterm defines a block of a blanket. To find the corresponding set system, we must first find the blanket and then remove subsets contained in other subsets.
- Blankets are more general, and lead to more separations. Theorem 3 assures us that, if there is a separation based on a blanket, then there is one based on a set system, but there are many separations based on blankets corresponding to the same set system.
- The theory is no more complicated with blankets than it is with set systems. Thus, generality does not cost us anything at this stage.
- The decomposition method described in Section 8 works with blankets, but not with set systems, and there are other examples [13] of techniques where only blankets could be used.
- Given the general theory of blankets, the implementer of a software package based on this theory may add the constraint that only set systems should be used, or that some blankets and some set systems be used, for example, β_u, β_v, σ_g and σ_y.

8 Finding Separations Using Cubes

Finding a separation of a function f consists of finding functions g and h, specified in some way. For convenience and simplicity, in the proof of Theorem 1 we have used (incomplete) truth tables for g and h. Note, however, that F is specified by cubes, and it would be desirable to have a method that handles all functions uniformly as cubes. We now present such a method.

Example 8 Table 11 shows a matrix F to be decomposed with respect to (U, V), where $U = \{x_1, x_2\}$ and $V = \{x_3, x_4, x_5, x_6\}$. Blanket β_v is

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

Assume that we have found blanket β_g below, in this case a set system, that satisfies the conditions of Theorem 1. Assume that the blocks of β_g are encoded as shown.

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

□

Table 11: Matrix F.

	x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3
1	Φ	0	1	Φ	0	0	1	Φ	Φ
2	0	Φ	1	1	0	0	1	Φ	Φ
3	1	Φ	Φ	Φ	0	1	Φ	1	Φ
4	0	Φ	0	Φ	Φ	1	Φ	1	Φ
5	1	Φ	1	Φ	0	Φ	Φ	Φ	1
6	Φ	Φ	1	0	1	Φ	0	0	0
7	Φ	Φ	0	Φ	0	0	Φ	Φ	0
8	1	1	Φ	Φ	Φ	1	0	Φ	Φ

A number of new concepts are now introduced with the aid of Example 8.

To define a function g by a matrix G of cubes, we must represent each minterm relevant to g by some cube. All minterms in $\{0,1\}^4$ are relevant except 0010, 0110, and 1110. Instead of minterms we will use blocks of β_v. As we have seen, each such block has the form $B = F_{v \supseteq d}$ for some $d \in \Sigma^s$. A minterm d of g that so defines B will be called a *characteristic minterm* of B. Note that every minterm relevant to g is a characteristic minterm of exactly one block of β_v.

Table 12: Characteristic minterms for g.

Block of β_v	Rows of F	Characteristic minterms
B_1	$\{7\}$	$\{0000, 0100\}$
B_2	$\{3, 4, 8\}$	$\{0001, 0101\}$
B_3	$\{4, 8\}$	$\{0011, 0111\}$
B_4	$\{1, 5\}$	$\{1000\}$
B_5	$\{3, 5, 8\}$	$\{1001, 1101\}$
B_6	$\{6\}$	$\{1010\}$
B_7	$\{6, 8\}$	$\{1011\}$
B_8	$\{1, 2, 5\}$	$\{1100\}$
B_9	$\{8\}$	$\{1111\}$

Blocks of β_v and their characteristic minterms are shown in Table 12. For example, minterm 0011 is a characteristic minterm of block B_3 consisting of rows 4 and 8 of F, since the set of all rows that contain 0011 in the v positions is precisely $\{4, 8\}$. Another characteristic minterm of B_3 is 0111. We will ensure that all the characteristic minterms of a block will be accounted for by the cube corresponding to that block.

The *v-intersection* of a block B of rows of F is defined as the intersection of the v-projections of the rows in B: $\bigcap_{t \in B} t^v$.

For example, for B_3 the v-intersection is $0\Phi\Phi1$, which contains all the minterms common to Row 4 and Row 8. Note that this v-intersection contains not only the characteristic minterms of B_4, but also two other minterms, namely 0001 and 0101. These other minterms will be accounted for by block B_2, since they are characteristic minterms of that block. The v-intersections will be used as the input cubes for g.

Computing the v-intersections for each block of β_v, we obtain Columns 2–5 of Table 13. (For now, ignore the last line and the columns labeled 'Final cubes.') In effect, we have now represented all the relevant minterms of g by the v-intersections in Table 13. These v-intersections are now the input cubes for G.

Recall that each block B of β_v is contained in some block B' of blanket β_g used in the decomposition. We assign the output values associated with B' to the cube of B. In our example, the outputs will be assigned as in the sixth and seventh columns of Table 13. Block $B_1 = \{7\}$ is contained only

Table 13: Cubes for G .

Block of β_v	First cubes				Outputs	Final cubes			
B_1	0	Φ	0	0	00	0	Φ	0	0
B_2	0	Φ	0	1	00	0	Φ	0	1
B_3	0	Φ	Φ	1	00	0	Φ	Φ	1
B_4	1	Φ	0	0	11	1	Φ	0	0
B_5	1	Φ	0	1	01	1	Φ	0	1
B_6	1	0	1	Φ	10	1	0	1	Φ
B_7	1	0	1	1	10	1	0	1	1
B_8	1	1	0	0	11	1	1	0	0
B_9	Φ	Φ	Φ	1	00	0	Φ	Φ	1
					00	Φ	1	1	1

in block $\{3, 4, 7, 8\}$ of β_g; hence, its cube should be assigned output 00. The same type of argument applies to all the other blocks, except B_9, which is contained in all four blocks of β_g. We arbitrarily assign the value 00 to this block.

Matrix G constructed in this fashion is not necessarily consistent. Roughly speaking, we must ensure that two cubes that have a nonempty intersection are assigned the same output value. For example, cube $\Phi\Phi\Phi 1$ of block B_9 has a nonempty intersection with cube $0\Phi 01$ of B_2. This does not matter, since the same output has been assigned to both cubes. But $\Phi\Phi\Phi 1$ also has a nonempty intersection with cube $1\Phi 01$ of B_5, which has been assigned a different output. This conflict must be resolved.

In the following, the cube of block B_i is called c_i. One verifies that the following pairs of cubes have nonempty intersections and are assigned different outputs: (c_5, c_9), (c_6, c_9), and (c_7, c_9). We avoid these conflicts by subtracting cubes $c_5 = 1\Phi 01$, $c_6 = 101\Phi$, and $c_7 = 1011$ from cube $c_9 = \Phi\Phi\Phi 1$[3]. Then, only those minterms in $\Phi\Phi\Phi 1$ which do not conflict with any of the cubes c_5, c_6, and c_7 will be assigned the value 00. One verifies that this subtraction[4] results in two cubes: $0\Phi\Phi 1$ and $\Phi 111$, as shown in Table 13.

Before we describe the general method, we need to point out some prop-

[3]We show later that c_6 needs not be subtracted.

[4]An efficient method for carrying out such cube operations can be found in [2].

erties of cubes and blankets.

Proposition 3 *Let B and B' be distinct blocks of β_v, and let c and c' be the corresponding cubes (v-intersections). Then*

$$B \subseteq B' \text{ iff } c \supseteq c'.$$

Proof: The proof is given in the appendix. $\Box$

To illustrate this proposition consider the blocks of β_v in Table 13. They are partially ordered by set inclusion. For example, $B_2 \supseteq B_3 \supseteq B_9$, since $\{3,4,8\} \supseteq \{4,8\} \supseteq \{8\}$. The corresponding cubes are in reverse order: $c_2 \subseteq c_3 \subseteq c_9$, since $0\Phi01 \subseteq 0\Phi\Phi1 \subseteq \Phi\Phi\Phi1$.

Proposition 4 *Let B and B' be distinct blocks of β_v, and let c and c' be the corresponding cubes. Cube c contains a characteristic minterm of B' iff $c \supseteq c'$. Also, if c contains a characteristic minterm of B', then c' contains no characteristic minterms of B.*

Proof: The proof is given in the appendix. $\Box$

In the example of Table 13, there are conflicts (c_5, c_9), (c_6, c_9), and (c_7, c_9). Note that c_9 contains the characteristic minterms 1001 and 1101 of B_5, and the characteristic minterm 1011 of B_7. However, it does not contain the characteristic minterm 1010 of B_6. One verifies that $c_9 \supseteq c_5$, and $c_9 \supseteq c_7$, but $c_9 \not\supseteq c_6$. It will be shown below that these containments mean that only c_5 and c_7 need to be subtracted.

The propositions above permit us to develop a general algorithm for resolving conflicts. To keep the ideas simple, we do not attempt to make the algorithm efficient, but just to convince the reader that method works. To construct G we proceed as follows:

1. For each block B of β_v find the v-intersection c of B; this becomes a *candidate input cube* for G.

2. Find any block B' of β_g that contains B, and assign the output vector d of B' to b.

3. Let $i = 1$, and pick cube c_i. If there is a cube $c_j, j \neq i$, such that

   - c_j has been assigned a different output than c_i, and
   - c_j contains a characteristic minterm of c_i,

   then replace c_j by $c_j - c_i$.

4. Repeat Step 2 for each c_j conflicting with c_i as above.

5. Repeat Steps 2 and 3 for $i = 2, \ldots, k$, where k is the number of blocks in β_v.

In our example, the first conflict is (c_5, c_9), so we replace c_9 by $c_9' = c_9 - c_5$. The second conflict is (c_7, c_9'), so we replace c_9' by $c_9'' = c_9' - c_7$. There are no other conflicts, so the final cubes are as shown in Table 13. Note that (c_6, c_9') is only an apparent conflict, since c_9 does not contain any characteristic minterms of c_6, even though it shares with c_6 the non-characteristic minterm 1011.

Next, we obtain a specification of function H. Here, we deal with blocks of the product $\beta_u * \beta_g$ to derive the input cubes of H, and with blocks of β_y to derive the output cubes, where

$$\beta_u = \{\overline{1,2,4,6,7};\ \overline{2,4,6,7};\ \overline{1,3,5,6,7};\ \overline{3,5,6,7,8}\},$$

and

$$\beta_y = \{\overline{1,2,3,4,5}^{111};\ \overline{1,2,3,4,7}^{110};\ \overline{1,2,5}^{101};\ \overline{1,2,7}^{100};\ \overline{3,4,5,8}^{011};\ \overline{3,4,7,8}^{010};\ \overline{5,8}^{001};\ \overline{6,7,8}^{000}\}.$$

We find

$$\beta_u * \beta_g = \{\overline{4,7}^{B_1'};\ \overline{6}^{B_2'};\ \overline{1,2}^{B_3'};\ \overline{2}^{B_4'};\ \overline{3,7}^{B_5'};\ \overline{3,5}^{B_6'};\ \overline{1,5}^{B_7'};\ \overline{3,7,8}^{B_8'};\ \overline{3,5,8}^{B_9'};\ \overline{6,8}^{B_{10}'};\ \overline{5,8}^{B_{11}'}\}.$$

To compute the table for H we consider each block of $\beta_u*\beta_g$ in turn, and generate Table 14 as follows. For $B_1' = \{4,7\}$, the u-intersections from Table 11 are $0\Phi \cap \Phi\Phi = 0\Phi$, so 0Φ becomes the first part of an input cube of H. Since block B_1' is contained in the block of β_g that has output 00, then 00 becomes the second part of the input cube. The output value assigned to this cube is $\Phi 1 \Phi \cap \Phi\Phi 0 = \Phi 10$; this is the y-intersection of Rows 4 and 7. The remaining rows are completed in the same fashion. Note that B_{11}' appears in blocks 01 and 11 of β_g; we arbitrarily pick 01 as its output.

In general, to find H, we carry out the following steps:

Table 14: Cubes for H .

Block	β_u	β_g	Output
B'_1	0Φ	00	$\Phi 10$
B'_2	$\Phi\Phi$	10	000
B'_3	00	11	$1\Phi\Phi$
B'_4	0Φ	11	$1\Phi\Phi$
B'_5	1Φ	00	$\Phi 10$
B'_6	1Φ	01	$\Phi 11$
B'_7	10	11	$1\Phi 1$
B'_8	11	00	010
B'_9	11	01	011
B'_{10}	11	10	000
B'_{11}	11	01	$0\Phi 1$

1. For each block B of blanket $\beta_u * \beta_g$ find the u-intersection b of B; this becomes the first part of an input cube for H.

2. Find a block C of β_g containing B; the output c assigned to C becomes the second part of the input cube for H.

3. Assign to (b, c) the y-intersection d of B. Now (b, c, d) is a row of H.

Theorem 5 *Let $F(x)$ be a consistent function matrix and let (U, V) be a pair of subsets of X satisfying $U \cup V = X$. Let β_g be a blanket satisfying the conditions of Theorem 1. Then matrices G and H constructed by the algorithm given above constitute a separation of F.*

Proof: The proof is given in the appendix. □

We close this section by pointing out that this method does not work if we use set systems throughout, rather than blankets. If we try to use set system $\sigma_v = max \beta_v$ instead of β_v, we have no way of representing the characteristic minterms of B_3, since it has been absorbed in B_2, etc. Thus function g obtained in this way would not be defined for all relevant minterms.

9 Finding Blanket β_g

So far we have assumed that the blanket β_g is given. In this section we show how β_g can be calculated. We return to Example 3, where the function of Table 1 is decomposed, with $U = \{x_1, x_4\}$ and $V = \{x_2, x_3, x_4\}$. For this function we have

$$\beta_u = \{\overline{1,3,5};\ \overline{3,4,6,7};\ \overline{2,3,5};\ \overline{3,4,6}\},$$

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

and

$$\beta_y = \{\overline{4,5};\ \overline{4,6};\ \overline{2,3,7};\ \overline{1,3,6,7}\}.$$

We are looking for a blanket β_g satisfying

- $\beta_v \leq \beta_g$, and
- $\beta_u * \beta_g \leq \beta_y$.

We verify that β_v itself satisfies these conditions. The first condition holds trivially. For the second condition, we use a proposition of [13]:

Proposition 5 *Let U and V be two subsets of the set X of input variables. Then*

- *if $V \supseteq U$, then $\beta_v \leq \beta_u$;*
- *if $W = U \cup V$, then $\beta_u * \beta_v \leq \beta_w$.*

Proof: The proof is given in the appendix. □

To complete the proof that $\beta_u * \beta_v \leq \beta_y$, note that from the second property of Proposition 5 with $W = X$, it follows that $\beta_u * \beta_v \leq \beta_x$. Because F is consistent, we always have $\beta_x \leq \beta_y$. Hence, $\beta_u * \beta_v \leq \beta_y$.

Of course, using β_v as β_g leads to a trivial separation. Usually, we try to minimize the number of blocks of β_g, to minimize the number of outputs of g. We can try merging two blocks of β_v to obtain β'_v. Clearly, $\beta_v \leq \beta'_v$; however, β'_v may not satisfy the second condition required by Theorem 1.

For example, we can merge B_1 and B_2, obtaining new block $\{1, 2, 3, 7\}$. Intersecting this block with blocks of β_u, we get $\{1, 3\}, \{3, 7\}, \{2, 3\}, \{3\}$.

Since all these blocks are contained in blocks of β_y, blocks B_1 and B_2 are mergeable. On the other hand, if we merge B_1 and B_4, we get new block $\{1,2,3,4\}$, and intersections $\{1,3\}, \{3,4\}, \{2,3\}, \{3,4\}$, and $\{3,4\}$ is not contained in any block of β_y. Hence, B_1 and B_4 are not mergeable.

We can repeat the merging procedure, until we get a set system whose blocks are no longer mergeable. In our example, one can verify that blocks B_1, B_2, B_3, and B_5 can all be merged into a single block, as can blocks B_4, B_6, and B_7. This way we obtain $\beta_g = \{\overline{1,2,3,6,7}; \overline{4,5,6}\}$.

We now outline a procedure for finding a blanket β_g satisfying the conditions of Theorem 1. The procedure actually produces a set system σ_g, which guarantees the smallest possible number of blocks, and hence the smallest number of outputs in g.

Two blocks B_i and B_j of blanket β_v are *compatible* if blanket β_{ij} obtained from β_v by merging B_i and B_j into a single block satisfies the second condition of Theorem 1. Otherwise, the blocks are *incompatible*. A subset δ of blocks of β_v is a *compatibility class* of blocks, if the blocks in δ are pairwise compatible. A compatibility class is *maximal* if it is not contained in any other compatibility class.

From the computational point of view, finding maximal compatiblity classes is equivalent to finding maximal cliques in a graph $\Gamma = (N, E)$, where the set N of nodes is the set of blocks of β_v, and the set E of edges corresponds to compatible pairs.

The next step in the calculation of σ_g is the selection of a minimum-cardinality set of maximal classes that covers all the blocks of β_v. The minimum cardinality results in a set system.

In certain heuristic strategies, both procedures (for finding maximal compatibility classes, and for finding the minimal cover) can be reduced to the graph coloring problem. Thus the task of finding σ_g can be reduced to the graph coloring problem in the graph $\Gamma = (N, E)$ defined above. The number of blocks of σ_g corresponds to the minimal number k of colors for Γ. Although the problem of finding the minimal chromatic number for a given graph is NP-complete, a number of fast heuristics have been developed for it [25].

In the example above, the incompatible pairs are: (B_1, B_4), (B_1, B_6), (B_1, B_7), (B_2, B_4), (B_2, B_6), (B_2, B_7), (B_3, B_6), (B_4, B_5), and (B_5, B_7). Therefore two colors are needed: one for nodes B_1, B_2, B_3, B_5, and the other, for nodes B_4, B_6, B_7. The union of all the sets of rows associated with nodes assigned the same color forms a block of σ_g. For example, for nodes B_4, B_6, B_7 we have $\{4\} \cup \{5\} \cup \{4,6\} = \{4,5,6\}$. Thus we find $\sigma_g = \{\overline{1,2,3,6,7}; \overline{4,5,6}\}$.

The following example[5] shows that it is not possible to always replace blankets by set systems when using the block merging method. Consider matrix F of Table 15, and its decomposition (G, H) with respect to $U = \{x_1, x_2\}$ and $V = \{x_3, x_4\}$, as shown in Table 16.

Table 15: Matrix F.

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

Table 16: Matrices G and H.

Matrix G

x_3	x_4	w
0	0	0
0	1	1
1	0	1
1	1	0

Matrix H

x_1	x_2	w	z_1	z_2
0	Φ	Φ	0	1
1	0	Φ	0	0
1	1	0	1	0
1	1	1	0	0

We have $\beta_u = \{\overline{1};\ \overline{2};\ \overline{3,4,5,6}\}$, $\beta_v = \{\overset{B_1}{\overline{1,2,3}};\ \overset{B_2}{\overline{1,4}};\ \overset{B_3}{\overline{2,5,6}};\ \overset{B_4}{\overline{6}}\}$, and $\beta_y = \{\overline{2,4,5,6};\ \overline{1};\ \overline{3,6}\}$.

For this decomposition, blanket γ_g is

$$\gamma_g = \{\overline{1,2,3,6};\ \overline{1,2,4,5,6}\}.$$

One blanket β_g satisfying $\gamma_g \leq \beta_g$ is $\beta_g = \gamma_g$. This blanket can be obtained by merging blocks B_1 with B_4, and B_2 with B_3.

[5]This is a modification of an unpublished example by J. J. Lou

If we use set systems throughout instead of blankets, we have $\sigma_u = \beta_u$, $\sigma_y = \beta_y$ and $\sigma_v = \{\overline{1,2,3};\ \overline{1,4};\ \overline{2,5,6}\}$.

Since 00 and 11 are both assigned output 0 by g, Rows 3 and 6 of F must be in the same block of σ_g. We cannot obtain such a σ_g by merging blocks $\overline{1,2,3}$ and $\overline{2,5,6}$ of σ_v, since $\{1,2,3,5,6\}$ has intersection $\{3,5,6\}$ with σ_u, and this block is not contained in any block of σ_y.

10 Decompositions

Suppose we have a separation (G, H) of a matrix F of n variables, where G has s inputs and p outputs, and H has $r + p$ inputs and m outputs. Such a separation is not quite a decomposition, because in a decomposition we want G and H to have strictly fewer inputs than F. Thus we should insist that $s < n$ and $r + p < n$.

Definition 2 A *decomposition* of F *with respect to* (U, V) is a separation (G, H), in which $s < n$, and $r + p < n$.

Proposition 6 *If a separation* (G, H) *based on blanket* β_g *is a decomposition, then* $s < n$ *and the number* q *of blocks of* β_g *satisfies*

- $r + \lceil \log_2 q \rceil < n$.

We now develop a useful necessary condition for the existence of a decomposition. Let β be a blanket on a set, and $\beta' = \{B'_i\}$, a blanket on the same set. The *quotient* of a block B of β by β' is a set system on the set B defined by

$$B/\beta' = max\{B \cap B'_i\}.$$

The *block-cover cost* $c(B_1, \beta')$ is the minimum number of blocks of B/β' required to cover B, in the sense that the union of these blocks is B. For example, let

$$\beta = \{\overbrace{\overline{1,2,3}}^{B_1};\ \overbrace{\overline{3,6,7}}^{B_2};\ \overbrace{\overline{1,2,5}}^{B_3};\ \overbrace{\overline{4,6}}^{B_4}\},$$

and

$$\beta' = \{\overline{1,3,6,7}; \overline{2,3,7}; \overline{4,6}; \overline{4,5}\}.$$

To calculate $c(B_1, \beta')$, we first intersect B_1 with all the blocks of β', obtaining $\overline{1,3};\ \overline{2,3};\ \emptyset;\ \emptyset$. Applying the *max* operation, we get $B_1/\beta' = \{\overline{1,3};\ \overline{2,3}\}$. Since both blocks are needed to cover B_1, we have $c(B_1, \beta') = 2$. Similarly, we find $B_2/\beta' = \{\overline{3,6,7}\}$, $B_3/\beta' = \{\overline{1};\ \overline{2};\ \overline{5}\}$, and $B_4/\beta' = \{\overline{4,6}\}$. Thus $c(B_2, \beta') = 1$, $c(B_3, \beta') = 3$, and $c(B_4, \beta') = 1$.

The *cover cost*, $c(\beta, \beta')$, is now defined as the largest block-cover cost, that is,

$$c(\beta, \beta') = max\{c(B, \beta') \mid B \in \beta\},$$

where *max* here denotes the largest integer in a set. In the example above, $c(\beta, \beta') = 3$.

The following result was previously stated without proof by Łuba [15]. A proof was provided by Brzozowski and Łuba [5] and improved by Lou [13].

Theorem 6 *Let F be a consistent function matrix and let U and V be subsets of X, $U \cup V = X$, where X has n elements, U has $r < n$ elements, and V has $s < n$ elements. If a decomposition of F based on blanket β_g exists, then*

$$n - r > \lceil \log_2 c(\beta_u, \beta_y) \rceil .$$

Proof: The proof is given in the appendix. □

Intuitively, the condition in the theorem can be interpreted as follows. If (g, h) is to be a decomposition, then h must have fewer inputs than f, i.e., $r + p < n$, or, equivalently, $p < n - r$. Hence, the number p of outputs from g must be at most $n - r - 1$. Let $q = 2^{n-r-1}$; then q is the maximum number of possible output combinations from g. Now $c(\beta_u, \beta_y)$ is the maximum cost of covering a block of β_u by blocks of β_y. The maximum number of blocks of β_y needed to cover a block of β_u can be interpreted as the amount of information we still require to determine the outputs y, if we only know the inputs u. At least this amount of information must come from the outputs of g, which provide at most q combinations. Hence, q must be greater than or equal to $c(\beta_u, \beta_y)$. See the proof of the theorem for more details.

Example 9 Let us find all the sets U for which a disjoint decomposition of the matrix F of Table 1 is not ruled out by the condition of Theorem 6. For convenience, we will use c_0 as a shorthand for $\lceil \log_2 c(\beta_u, \beta_y) \rceil$. Recall that

$$\beta_y = \{\overline{4,5};\ \overline{4,6};\ \overline{2,3,7};\ \overline{1,3,6,7}\}.$$

For one-variable sets, that is, for $U = \{x_i\}$, we have $n - r = 3$. The condition of the theorem is satisfied, because the number of blocks in β_y is 4;

consequently, the cost $c(B, \beta_y)$ cannot be greater than 4 for any block B of β_u. Therefore, $c_0 \leq 2 < n - r = 3$.

For larger sets U, some computation is required. For $U = \{x_1, x_2\}$, we have:

$$\beta_u = \{\overbrace{\overline{1,3,4,7}}^{B_1};\ \overbrace{\overline{4,5,6,7}}^{B_2};\ \overbrace{\overline{2,3,4}}^{B_3};\ \overbrace{\overline{4,5,6}}^{B_4}\},$$

$$B_1/\beta_y = max\{\overline{4};\ \overline{4};\ \overline{3,7};\ \overline{1,3,7}\} = \{\overline{4};\ \overline{1,3,7}\},$$

$$B_2/\beta_y = max\{\overline{4,5};\ \overline{4,6};\ \overline{7};\ \overline{6,7}\} = \{\overline{4,5};\ \overline{4,6};\ \overline{6,7}\},$$

$$B_3/\beta_y = max\{\overline{4};\ \overline{4};\ \overline{2,3};\ \overline{3}\} = \{\overline{4};\ \overline{2,3}\},$$

and

$$B_4/\beta_y = max\{\overline{4,5};\ \overline{4,6};\ \overline{\emptyset};\ \overline{6}\} = \{\overline{4,5};\ \overline{4,6}\}.$$

The corresponding costs are $c(B_1, \beta_y) = 2$, $c(B_2, \beta_y) = 2$, $c(B_3, \beta_y) = 2$, $c(B_4, \beta_y) = 2$. Hence $c_0 = 1 < n - r = 2$, and the condition is satisfied.

For $U = \{x_1, x_3\}$,

$$\beta_u = \{\overbrace{\overline{1,3,6,7}}^{B_1};\ \overbrace{\overline{1,4,5,6}}^{B_2};\ \overbrace{\overline{2,3,6}}^{B_3};\ \overbrace{\overline{2,4,5,6}}^{B_4}\}.$$

Here U fails to satisfy the condition, because

$$B_4/\beta_y = max\{\overline{4,5};\ \overline{4,6};\ \overline{2};\ \overline{6}\} = \{\overline{4,5};\ \overline{4,6};\ \overline{2}\}.$$

Hence, $c_0 = 2 \not< 2 = n - r$.

Altogether, the condition is satisfied for the following subsets of X:

$$\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_1, x_2\}, \{x_1, x_4\}, \text{ and } \{x_2, x_4\}.$$

After the verification of the necessary conditions from Theorem 1, we find that only three of these sets, namely $U_1 = \{x_1\}, U_2 = \{x_2\}, U_3 = \{x_3\}$, lead to disjoint decompositions. $\square$

11 Conclusions

We have presented a comprehensive theory of serial decompositions of multiple-output, incompletely specified, Boolean functions represented in the compact consistent-cube notation. The task of finding a decomposition of a function

f has been divided into the problem of finding a separation of a given function, which is concerned only with functional properties, and the problem of finding functions g and h having fewer inputs than f. We have discussed the use of both blankets and set systems, and have illustrated various options in the decomposition process.

Many ideas presented here have been implemented and applied to practical decomposition problems. In particular, a program, called DEMAIN, uses some of these ideas. Serial decomposition constitutes a part of the techniques used in DEMAIN. A discussion of DEMAIN is outside the scope of this paper, and we refer the interested reader to [17, 18, 19, 20] for additional details.

Appendix

Proof of Theorem 1

Suppose that a blanket β_g satisfying the conditions of the theorem exists, and has q blocks. We define a function g with s inputs and $p \geq \lceil \log_2 q \rceil$ outputs as follows. Encode the q blocks of β_g by p variables in such a way that each block B is assigned a distinct p-tuple $\alpha(B)$; otherwise, the coding is arbitrary. For each $b \in \Sigma^n$ relevant to F, block $F_{v \supseteq b^v}$ is a nonempty block of β_v. By the first condition of the theorem, this block is contained in some block B of β_g. Then assign $g(b^v) = \alpha(B)$.

We claim that $\gamma_g \leq \beta_g$, by construction of g. For suppose that t is in the block of γ_g that produces value e for the output w of g. Then there is a minterm d of g such that $t^v \supseteq d$, and $g(d) = e$. Consider the row containing d in our construction of g. Minterm d defines block $F_{v \supseteq d}$ of β_v. Since we know that $g(d) = e$, we know that the block of β_g containing $F_{v \supseteq d}$ must be the block that has been assigned the code e. Thus, if t is in the block of γ_g that produces e, then t is in the block of β_g that corresponds to e. In other words, every block of γ_g is contained in a block of β_g, i.e., $\gamma_g \leq \beta_g$.

Next we construct a function h with $r + p$ inputs and m outputs. Let $b \in \Sigma^n$ be a minterm relevant to F. Since b is relevant, $F_{x \supseteq b}$ is not empty. By construction of g, $g(b^v)$ is in Σ^p. Hence the input vector $(b^u, g(b^v))$ to function h is completely specified.

Define

$$h(b^u, g(b^v)) = \bigcap_{t \in B_1 \cap B_2} t^y,$$

where $B_1 = F_{u \supseteq b^u}$, and $B_2 = F_{w=g(b^v)}$. The intersection $B_1 \cap B_2$ is not empty since $F_{x \supseteq b} \subseteq F_{u \supseteq b^u} = B_1$, and $F_{x \supseteq b} \subseteq F_{w=g(b^v)} = B_2$. By the second condition of the theorem, $B_1 \cap B_2$ is contained in some block of β_y. Since F is consistent, the intersection defining h is nonempty, and h is indeed well defined.

Since $F_{x \supseteq b} \subseteq B_1 \cap B_2$,

$$h(b^u, g(b^v)) = \bigcap_{t \in B_1 \cap B_2} t^y \subseteq \bigcap_{t \in F_{x \supseteq b}} t^y = f(v).$$

In words, the larger set $B_1 \cap B_2$ can only add more constraints to the intersection; hence the intersection can only be the same or smaller. Thus (g, h) is indeed a separation of f. □

Proof of Theorem 2

For convenience, and without loss of generality, we assume that the variables $x_1, \ldots, x_n$ have been relabeled in such a way that $u = (x_1, \ldots, x_r)$ and $v = (x_{r+1}, \ldots, x_n)$. Consequently, for an n-tuple x, the first r components are x^u, and the last $s = n - r$ components, x^v.

We first claim that $\beta_v \leq \gamma_g$. Let b be relevant to F; then $F_{v \supseteq b^v}$ is a nonempty block of β_v, and every block of β_v has this form. Each such block is contained in the block $F_{w=g(b^v)}$ of γ_g. Hence our claim holds.

Now we check that $\beta_u * \gamma_g \leq \beta_y$. Let $B_1 = F_{u \supseteq a}$ be a block of β_u. Then $t \in B_1$ iff $t \supseteq aa'$ for some $a \in \Sigma^r$ and $a' \in \Sigma^{n-r}$. Let $B_2 = F_{w=e}$ be a block of γ_g. Then $t \in B_2$ iff $t \supseteq b'b$ for some $b' \in \Sigma^r$ and $b \in \Sigma^{n-r}$, such that $g(b) = e$. Thus $t \in B_1 \cap B_2$ implies $t \supseteq ab$. Define blanket β_h over F as

$$\beta_h = ne\{F_{z \supseteq c}\}, \tag{12}$$

where $c \in \Sigma^m$ and

$$F_{z \supseteq c} = \{t \in F \mid \exists d \in \Sigma^n : t^x \supseteq d \text{ and } h(d^u, g(d^v)) \supseteq c\}. \tag{13}$$

Suppose $h(a, g(b)) \supseteq k$. Clearly, $t \in B_1 \cap B_2$ implies $t \in F_{z \supseteq k}$. Hence $B_1 \cap B_2 \subseteq F_{z \supseteq k}$, and $\beta_u * \gamma_g \leq \beta_h$.

Since (g, h) is a separation of f, we have $f(ab) \supseteq h(a, g(b)) \supseteq k$. Hence $t \in F_{y \supseteq k}$, showing that $\beta_h \leq \beta_y$. Consequently, $\beta_u * \gamma_g \leq \beta_y$, as required. □

Proof of Theorem 3

One easily verifies the following properties of blankets:

$$\beta \leq \beta' \text{ and } \beta' \leq \beta'' \text{ imply } \beta \leq \beta'',$$

$$\beta \leq \beta' \text{ implies } \beta * \beta'' \leq \beta' * \beta'',$$

$$\text{if } \sigma = max \beta \text{ then } \beta \leq \sigma \text{ and } \sigma \leq \beta.$$

Suppose now that we have a set system σ_g satisfying the conditions of Theorem 3. Since $\beta_v \leq \sigma_v$ and $\sigma_v \leq \sigma_g$, we have

$$\beta_v \leq \sigma_g.$$

Also, since

$$\beta_u * \sigma_g \leq max(\beta_u * \sigma_g) = max(\sigma_u * \sigma_g) = \sigma_u \circ \sigma_g \leq \sigma_y \leq \beta_y,$$

we have

$$\beta_u * \sigma_g \leq \beta_y.$$

Hence σ_g satisfies the conditions of Theorem 1, and one can construct a separation (G, H) of F as in the proof of Theorem 1, where $\gamma_g \leq \sigma_g$. In this construction, we can also use set systems throughout for the following reasons.

In constructing g, we take a relevant minterm d of g, find the associated block B of β_v, and find a block C of β_g containing it. If there is choice, we pick an arbitrary block of β_g. Instead, we can pick a maximal block B' of β_v containing B, i.e., a block of σ_v. This block must be contained in some block C' of σ_g, which also contains the original block B. Hence the function g that is constructed using set systems throughout is one of the solutions we could have found using blankets.

Now consider the construction of h. We find a minterm d relevant to h and then blocks B of β_u and C of σ_g. Next we find the intersection $B \cap C$, which is guaranteed to be contained in a block D of β_y. Instead, we could have found a maximal block B' of β_u, containing B, i.e., a block of σ_u. Now $B' \cap C$ is contained in some maximal block D' of β_y, i.e., in a block of σ_y. Hence, the function h that is constructed using set systems throughout is one of the solutions we could have found using blankets. $\square$

Proof of Theorem 4

Suppose F has separation (G, H). Let $\sigma_g = max\gamma_g$, where γ_g is defined in (11). It is straightforward to verify that σ_g satisfies the conditions of the theorem. $\square$

Proof of Proposition 3

If B is a block of β_v, it is defined by some characteristic minterm d, i.e., $B = F_{v \supseteq d} = \{t \in F \mid t^v \supseteq d\}$. Cube c of B has the form $\bigcap_{t \in B} t^v$. Suppose $B \subseteq B'$; then cube c' of B' is the v-intersection of all the blocks of B and the blocks of $B' - B$. Hence, $c' \subseteq c$.

Conversely, suppose $B \not\subseteq B'$. Then there exists $t \in B - B'$. Since $t \notin B'$, t^v does not contain the characteristic minterm d' of B'. Hence c, the v-intersection of all the t in B, cannot contain d', whereas c' does contain it. Therefore, $c' \not\subseteq c$. $\square$

Proof of Proposition 4

Let d' be a characteristic minterm of B', and suppose $c \supseteq d'$. Then every $t \in B$ satisfies $t^v \supseteq d'$, i.e., $B \subseteq B'$. By Proposition 3, $c' \subseteq c$. Now, if c' contained a characteristic minterm of c, then we would have $c \subseteq c'$, and $c = c'$, which is a contradiction. $\square$

Proof of Theorem 5

First we verify that G is consistent. Consider a minterm b relevant to F and let $d = b^v$. Minterm d of G is a characteristic minterm of precisely one block $B_d = \{t \mid t^v \supseteq d\}$ of β_v, and d is contained in the cube c_d of B_d.

We claim that d cannot be removed from c_d by the process of subtraction. Suppose another original cube c also contains d. In our construction, we subtract c_d from c; thus, we never subtract from c_d any cube containing d. At the end of the algorithm, d is contained only in the final cube derived from the original cube c_d. Hence, $g(d)$ is the output associated with that final cube, and G is consistent. By our construction, $g(b^v) \in \Sigma^p$ for every b relevant to F.

Now we show that $\gamma_g \leq \beta_g$. Suppose $t \in F_{w=e}$. Then there exists a minterm $d \in \Sigma^s$, such that $t^v \supseteq d$, and $g(d) = e$. Let B_e be the block of β_g assigned the value e. By our construction, B_e must contain block $F_{v \supseteq d}$ of β_v. Thus $t \in B_e$, and $F_{w=e} \subseteq B_e$, showing that $\gamma_g \leq \beta_g$.

Next, we check the consistency of H. Let $e = (e^u, e^w) \in \Sigma^{r+p}$ be a minterm relevant to H. Then e^u must appear in at least one row of F, and it must be a characteristic minterm of exactly one block, call it B_u, of β_u. Suppose the block of β_g that has been assigned the value e^w is B_g. Then e is involved in that row of H that corresponds to block B of $\beta_u * \beta_g$, where $B = B_u \cap B_g$.

Suppose e is also involved in a row $s = (s^u, s^w, s^z)$ of H that corresponds to block B' of $\beta_u * \beta_g$; thus $s^u \supseteq e^u$. Also, $t^u \supseteq s^u$ for all $t \in B'$, since s^u is the intersection of all the u-projections of rows of F that belong to B'. Therefore, $t^u \supseteq e^u$, for all $t \in B'$. Since $B_u = F_{u \supseteq e^u}$, we have $B' \subseteq B_u$. By construction, $B' \subseteq B_g$, because $s^w = e^w$. Altogether, $B' \subseteq B_u \cap B_g = B$. By the condition $\beta_u * \beta_g \leq \beta_y$, we know that B is contained in some block of β_y. Since s^z is defined as $\bigcap_{t \in B'} t^y$, we have $s^z \supseteq \bigcap_{t \in B_u} t^y$. Since this holds for all rows s of H in which e is involved, all these rows are compatible, since they have $\bigcap_{t \in B_u} t^y$ in common. Therefore, H is consistent.

It remains to verify that $f(b) \supseteq g(b^u, g(b^v))$ for every minterm b relevant to F. The argument is essentially the same as that in the proof of Theorem 1. $\square$

Proof of Proposition 5

Let B be the block of β_v with characteristic minterm d; then $t \in B$ iff $t^v \supseteq d$. Let $B' = F_{U \supseteq d^u}$. If $t \in B$, then $t^u \supseteq d^u$, since $V \supseteq U$. Hence, $t \in B'$, i.e., $B \supseteq B'$ and $\beta_v \leq \beta_u$.

For the second claim, if $W = U \cup V$, let $V' = V - U$; then $V \supseteq V'$ and $W = U \cup V'$. By the first part, $\beta_v \leq \beta_{v'}$. To prove the second part, it suffices to prove that $\beta_u * \beta_{v'} \leq \beta_w$. Suppose $B \in \beta_u * \beta_{v'}$; then $B = B_u \cap B_{v'}$, for some blocks B_u of β_u and $B_{v'}$ of $\beta_{v'}$. Suppose $B_u = F_{u \supseteq a}$ and $B_{v'} = F_{v' \supseteq b}$. Now, for any row t in B, we have $t^u \supseteq a$, and $t^{v'} \supseteq b$. Since U and V' are disjoint, we also have $t^w \supseteq ab$. Hence, $F_{w \supseteq ab}$ is nonempty, and is a block of β_w. Therefore, $t \in B$ implies t is in block $F_{w \supseteq ab}$ of β_w, and our claim follows. $\square$

Proof of Theorem 6

Here we follow the proof in [13]. To prove that

$$n - r > \lceil \log_2 c(\beta_u, \beta_y) \rceil ,$$

it is sufficient to show that

$$n - r - 1 \geq \log_2 c(\beta_u, \beta_y),$$

which is equivalent to showing that $q \geq c(\beta_u, \beta_y)$, where $q = 2^{n-r-1}$. Since (g, h) is a decomposition, g has at most $n - r - 1$ outputs, and, hence, at most q distinct output combinations. Therefore, there are at most q blocks of the form $F_{y \supseteq h(a,c)}$, for a fixed $a \in \{0,1\}^r$, and $c = g(d)$, as d varies over $\{0,1\}^s$. We claim that this set of blocks covers block $F_{u \supseteq a}$ of β_u.

Let $s \in F_{u \supseteq a}$; then there is a minterm b such that $s^x \supseteq b$ and $b^u = a$. Also, $g(b^v)$ is defined, since b is relevant to F; say $g(b^v) = c$. By definition of f, we have $f(b) = \bigcap_{t \in F_{x \supseteq b}} t^y$. Since $s^x \supseteq b$, we have $s \in \{t \mid t \in F_{x \supseteq b}\}$, and $s \supseteq f(b)$. By definition of separation, $f(b) \supseteq h(a, c)$. Thus, $s \in F_{y \supseteq h(a,c)}$. This shows that any block of β_u is covered by at most q blocks of β_y, and the theorem follows. □

References

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

[2] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis,* Kluwer Academic Publishers, Boston, MA, 1984.

[3] D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field Programmable Gate Arrays,* Kluwer Academic Publishers, Dordrecht, 1992.

[4] J. A. Brzozowski and J. J. Lou, Blanket Algebra for Multiple-Valued Function Decomposition, pp. 262–276 in *Algebraic Engineering*, C. L. Nehaniv and M. Ito, eds., World Scientific Publishing Co. Pte. Ltd., Singapore, 1999.

[5] 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, ON, Canada, 36 pp., January 1997.

[6] S.-C. Chang, M. Marek-Sadowska, and T. Hwang, Technology Mapping for TLU FPGAs Based on Decomposition of Binary Decision Diagrams, *IEEE Trans. on CAD*, Vol. 15, No. 10, pp. 1226–1236, 1996.

[7] M. J. Ciesielski and S. Yang, PLADE: A Two-Stage PLA Decomposition, *IEEE Trans. on CAD,* Vol. 11, No. 8, August 1992.

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

[9] J. Hartmanis and R. E. Stearns, *Algebraic Structure Theory of Sequential Machines,* Prentice-Hall, Englewood Cliffs, NJ, 1966.

[10] L. Józwiak, General Decomposition and its Use in Digital Circuit Synthesis, *VLSI Design,* Vol. 3, No. 3–4, pp. 225–248, 1995.

[11] L. Józwiak, Information Relationships and Measures in Application to Logic Design, Proc. *29th IEEE International Symposium on Multiple-Valued Logic,* Freiburg, Germany, IEEE Computer Society, Los Alamitos, pp. 228–235, 1999.

[12] Y.-T. Lai, K.-R. Pan, and M. Pedram, OBDD-Based Function Decomposition: Algorithms and Implementation, *IEEE Trans. on CAD,* Vol. 15, No. 8, pp. 977–990, 1996.

[13] J. J. Lou, *Decomposition of Multi-Valued Functions,* MMath Thesis, Department of Computer Science, University of Waterloo, Waterloo, ON, Canada, 88 pp., July 1998.

[14] J. J. Lou and J. A. Brzozowski, A Formalization of Shestakov Decomposition, Proc. *29th IEEE International Symposium on Multiple-Valued Logic,* Freiburg, Germany, IEEE Computer Society, Los Alamitos, pp. 66–71, 1999.

[15] T. Łuba, Multi-Level Logic Synthesis Based on Decomposition, *Microprocessors and Microsystems,* Vol. 18, No. 8, pp. 429–437, October 1994.

[16] T. Łuba and H. Selvaraj, A General Approach to Boolean Function Decomposition and its Application in FPGA-Based Synthesis, *VLSI Design*, Vol. 3, Nos. 3–4, pp. 289–300, 1995.

[17] T. Łuba, H. Selvaraj, M. Nowicka, and A. Krasniewski, Balanced Multilevel Decomposition and its Application in FPGA-Based Synthesis, *Logic and Architecture Synthesis,* G. Saucier and A. Mignotte, eds., Chapman & Hall, 1995.

[18] M. Nowicka, Balanced Decomposition for Technology Mapping of FPGAs, (in Polish), Ph.D. Thesis, Warsaw University of Technology, Warsaw, 1999.

[19] M. Nowicka, T. Łuba, and H. Selvaraj, Multilevel Decomposition Strategies in Decomposition-Based Algorithms and Tools, Proc. *International Workshop on Logic and Architecture Synthesis,* Grenoble, Institut National Polytechnique de Grenoble, pp. 129–136, 1997.

[20] M. Nowicka, T. Łuba, and M. Rawski, FPGA-Based Decomposition of Boolean Functions. Algorithms and Implementation, *Proc. Sixth international Conference on Advanced Computer Systems,* Szczecin,pp. 502–509, 1999.

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

[22] A. Sangiovanni-Vincentelli, A. Gamal, and J. Rose, Synthesis Methods for Field Programmable Gate Arrays, *Proc. IEEE,* Vol. 81, No. 7, pp. 1057–1083, 1993.

[23] 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.

[24] 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.

[25] M. Sysło, N. Deo, and J. Kowalik, *Discrete Optimization Algorithms,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1993.

[26] F. A. M. Volf, L. Józwiak, and M. P. J. Stevens, Division-Based versus General Decomposition-Based Multiple-Level Logic Synthesis, *VLSI Design*, Vol. 3, No. 3–4, pp. 267–287, 1995.