On the Minimization of Variables to Represent Partially Defined Classification Functions

Tsutomu Sasao
Department of Computer Science
Meiji University, Kawasaki 214-8571, Japan

***Abstract*—A partially defined classification function is a mapping from the set of k distinct vectors of n bits to m elements, where $k << 2^n$. Such a function can often be represented with fewer variables than n, by appropriately assigning values to don't cares. The number of variables can be further reduced by a linear transformation of the input variables. This paper shows an efficient method to find a linear transformation that reduces the number of variables. The method is illustrated with examples.**

***Keywords*-linear decomposition, logic design, partially defined function, support minimization, classification.**

I. Introduction

Given disjoint sets of samples, to find a simple rule to distinguish these sets is a major problem in machine learning and data mining. Occam's razor [2] recommends to use as simple rule as possible [1]. In this paper, we consider the problem to minimize the number of variables to represent such a rule. In particular, we use a linear decomposition to reduce the number of variables.

The rest of the paper is organized as follows: Section II introduces partially defined functions; Section III shows a method to minimize primitive variables for a function; Section IV introduces compound variables and linear transformations; Section V shows a method to reduce compound variables; Section VI considers the number of variables to represent a symmetric function; Section VII illustrates the reduction method for compound variables; Section VIII shows an exact minimization method for compound variables; and finally, Section IX concludes the paper.

II. Partially Defined Functions

Definition 2.1: Consider a set of k distinct vectors of n bits. These vectors are **registered vectors**. For each registered vector, assign an integer from 1 to m, where $2 \leq m \leq k$. A **registered vector table** shows **function value** for each registered vector. A **partially defined classification function** shows a corresponding integer value when the input vector is equal to a registered vector. Otherwise, the value of the function is undefined (*don't care*). A partially defined classification function is a mapping $f : D \rightarrow \{1, 2, \ldots, m\}$, where $D \subset B^n$ shows the set of k registered vectors, and $B = \{0, 1\}$. k is the **weight** of the function. Let F_i be the set of vectors $\vec{a} \in D$ such that $f(\vec{a}) = i$. We assume that $F_i \neq \phi$ for $i = 1, 2, \ldots, m$. Note that $D = \bigcup_{i=1}^{m} F_i$. When $m = k$, the function f is an **index generation function** [12]. When $m = 2$, the function f is a **decision function**.

TABLE 2.1
Registered Vector Table

x_1	x_2	x_3	x_4	f
1	0	0	0	1
0	1	0	0	1
0	1	1	0	2
1	1	0	1	2

TABLE 2.2
Illustration of support set.

		x_1	x_2	x_3	x_4	f
F_1	$\vec{a}_1$	1	0	0	0	1
	$\vec{a}_2$	0	1	0	0	1
F_2	$\vec{b}_1$	0	1	1	0	2
	$\vec{b}_2$	1	1	0	1	2

Example 2.1: The registered vector table in Table 2.1 shows a decision function with weight $k = 4$. ■

Definition 2.2: An m-tuple $(F_1, F_2, \ldots, F_m)$ shows a **partially defined classification function**, when $B^n \supset \bigcup_{i=1}^{m} F_i$, and a **completely specified classification function** when $B^n = \bigcup_{i=1}^{m} F_i$.

Definition 2.3: [6] Let $F_i \subseteq B^n$, for $i = 1, 2, \ldots, m$, and $F_i \cap F_j = \phi$ for $i \neq j$. For a partially defined function $(F_1, F_2, \ldots, F_m)$, the function $(E_1, E_2, \ldots, E_m)$ that satisfies

$$E_i \supseteq F_i \ (i = 1, 2, \ldots, m),$$

$E_i \cap E_j = \phi$,$(i \neq j)$, and $E_i, E_j \subseteq B^n$, is an **extension** of $(F_1, F_2, \ldots, F_m)$.

For a given partially defined function, many extensions exist. In this paper, we seek the extension that depends on the least number of variables.

Definition 2.4: Let $F_i \subseteq B^n$ $(i = 1, 2, \ldots, m)$. Given a partially defined function $(F_1, F_2, \ldots, F_m)$ and a subset $S \subseteq \{1, 2, \ldots n\}$, when $F_i|_S \cap F_j|_S = \phi$, holds for $i \neq j$, then S is a **support set**. In such a case, $(F_1|_S, F_2|_S, \ldots, F_m|_S)$ is independent of the variable x_j, where $j \in \overline{S}$. Such variable is **redundant**.

Example 2.2: Consider the decision function (F_1, F_2) shown in Table 2.2. In this case, $S = \{3, 4\}$ is the support set, since $F_1|_S \cap F_2|_S = \phi$ holds, where

$$\begin{aligned} F_1|_S &= \{(*, *, 0, 0), (*, *, 0, 0)\}, \quad \text{and} \\ F_2|_S &= \{(*, *, 1, 0), (*, *, 0, 1)\}. \end{aligned}$$

978-1-7281-5406-0/20/$31.00 ©2020 IEEE
DOI 10.1109/ISMVL49045.2020.00-19

Also, this function can be represented with two variables:

$$f = \bar{x}_3\bar{x}_4 \vee 2(x_3\bar{x}_4 \vee \bar{x}_3 x_4),$$

where $\vee$ denotes the max operation. Note that

$$E_1 = \{(*, *, 0, 0)\} \supseteq F_1,$$
$$E_2 = \{(*, *, 1, 0), (*, *, 0, 1)\} \supseteq F_2,$$

and * denotes either 0 or 1. ■

The next section shows that $\{3, 4\}$ is the minimum support set.

III. Minimization of Primitive Variables

This section shows a method to minimize the number of variables for a partially defined function.

Definition 3.1: Consider two vectors $\vec{a} \in F_i$ and $\vec{b} \in F_j$, where $i \neq j$:

$$\begin{aligned} \vec{a} &= (a_1, a_2, \ldots, \quad a_i, \quad \ldots, a_n), \\ \vec{b} &= (b_1, b_2, \ldots, \quad b_i, \quad \ldots, b_n). \end{aligned}$$

For some i and arbitrary $j \in \{1, 2, \ldots n\}$, if there exists a pair $(\vec{a}, \vec{b})$ satisfying the relation

$$\begin{aligned} a_j &= b_j \quad (\text{When } j \neq i) \quad \text{and} \\ a_j &\neq b_j \quad (\text{When } j = i), \end{aligned}$$

then the function $(F_1, F_2, \ldots, F_m)$ **depends** on x_i.

Example 3.1: Consider the function (F_1, F_2) shown in Table 2.2. This function depends on x_3. This can be verified by the fact

$$\begin{aligned} &\vec{a}_2 = (0, 1, 0, 0) \in F_1 \quad \text{and} \\ &\vec{b}_1 = (0, 1, 1, 0) \in F_2. \end{aligned}$$

■

Definition 3.2: In a partially defined function $(F_1, F_2, \ldots, F_m)$, let $\vec{a} \in F_i$ and $\vec{b} \in F_j$, $(i \neq j)$. Then, $\vec{d} = \vec{a} \oplus \vec{b}$ is a **difference vector**. The **set of difference vectors** [20] is denoted by D_f. Let $\vec{a}$ and $\vec{b}$ be two vectors. If $a_i \leq b_i$ for all i, and $a_j < b_j$ for some j, then $\vec{a} < \vec{b}$. In the set of difference vectors, the set after removing the vectors $\vec{b}$ satisfying $\vec{a} < \vec{b}$ is called the **set of minimal difference vectors**, and denoted by MD_f.

The following is an extension of the algorithm for two-valued output functions [7], [11] into multi-valued output functions.

Algorithm 3.1: (Minimization of primitive variables)

1) Obtain the set of minimal difference vectors MD_f for the partially defined function $(F_1, F_2, \ldots, F_m)$. For each $\vec{d} = (d_1, d_2, \ldots, d_n) \in MD_f$, make a clause

$$C(\vec{d}) = z_1 \vee z_2 \vee \cdots \vee z_n,$$

where

$$z_j = \begin{cases} y_i & (\text{When } d_j = 1) \\ 0 & (\text{When } d_j = 0) \end{cases}$$

2) Construct the product-of-sums expression

$$R = \bigwedge_{\vec{d} \in MD_f} C(\vec{d}).$$

TABLE 3.1
The set of difference vectors for the function in Table 2.2

x_1	x_2	x_3	x_4	TAG
1	1	1	0	$(\vec{a}_1, \vec{b}_1)$
0	1	0	1	$(\vec{a}_1, \vec{b}_2)$
0	0	1	0	$(\vec{a}_2, \vec{b}_1)$
1	0	0	1	$(\vec{a}_2, \vec{b}_2)$

3) Convert R into a sum-of-products expression [10], and simplify it. Then, the product term with the fewest literals shows the **minimum support set**.

Example 3.2: Consider the decision function (F_1, F_2) shown in Table 2.2. To represent the function, vectors in F_1 and F_2 must be distinguished. From the Table 2.2, we have the following four conditions:

1) To distinguish $\vec{a}_1$ and $\vec{b}_1$, either x_1, x_2 or x_3 is necessary.
2) To distinguish $\vec{a}_1$ and $\vec{b}_2$, either x_2 or x_4 is necessary.
3) To distinguish $\vec{a}_2$ and $\vec{b}_1$, x_3 is necessary.
4) To distinguish $\vec{a}_2$ and $\vec{b}_2$, either x_1 or x_4 is necessary.

Since the condition 1) is dominated by the condition 3), we can remove the condition 1). ■

Usually, the number of the registered vectors is very large, and the number of conditions tends to be huge. Experimental results show that many dominated conditions are generated. Also, the same conditions are generated many times. So, we represent the conditions by binary vectors to remove dominated conditions efficiently.

Example 3.3: Consider the decision function (F_1, F_2) shown in Table 2.2.

1) Table 3.1 shows the set of difference vectors. A TAG shows the pair of vectors that generates the difference vector. In the set D_f, since $(1, 1, 1, 0) > (0, 0, 1, 0)$, $(1, 1, 1, 0)$ is deleted from D_f. The set of minimal difference vectors is $MD_f = \{(0, 1, 0, 1), (0, 0, 1, 0), (1, 0, 0, 1)\}$.
Corresponding to the set of minimal difference vectors, we have the set of clauses:

$$\begin{aligned} C(\vec{a}_1, \vec{b}_2) &= y_2 \vee y_4, \\ C(\vec{a}_2, \vec{b}_1) &= y_3, \qquad \text{and} \\ C(\vec{a}_2, \vec{b}_2) &= y_1 \vee y_4. \end{aligned}$$

2) The product of all the clauses is $R = (y_2 \vee y_4) y_3 (y_1 \vee y_4)$.
3) By converting R into a sum-of-products expression, and by simplifying it, we have: $R = y_1 y_2 y_3 \vee y_3 y_4$.
The minimum support set is $\{3, 4\}$. Thus, this function can be represented by x_3 and x_4. ■

Theorem 3.1: If the set of difference vectors for the function f contains the unit vector $\vec{e}_i$ (only the i-th element is 1, and other elements are 0), then the function f depends on x_i.

IV. Compound Variables and Linear Transformation

For partially defined functions, the number of variables often can be reduced further by a **linear decomposition** [8], [9], [13], [14], [16].

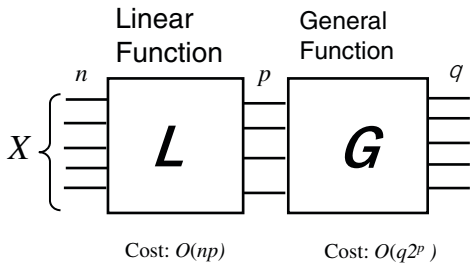

Fig. 4.1. Linear Decomposition

In the linear decomposition shown in Fig. 4.1, L realizes linear functions, while G realizes general functions (in most cases, non-linear functions). The cost of the linear part is $O(np)$, while the cost of the general part is $O(q2^p)$.

Definition 4.1: A **compound variable** has a form $y = c_1x_1 \oplus c_2x_2 \oplus \cdots \oplus c_nx_n$, where $c_i \in \{0,1\}$. The **compound degree** of a variable y is $\sum_{i=1}^{n} c_i$, where $\sum$ denotes an ordinary integer addition, and c_i is treated as an integer. A **primitive variable** is a compound variable with the degree one.

In this paper, we use the following linear transformation;

$$\begin{aligned} z_1 &= h_1(y_1, y_2, \ldots, y_p) \\ z_2 &= h_2(y_1, y_2, \ldots, y_p) \\ &\cdots \\ z_q &= h_q(y_1.y_2.\ldots, y_p), \end{aligned}$$

where

$$\begin{aligned} y_1 &= a_{1,1}x_1 \oplus a_{1,2}x_2 \oplus \cdots \oplus a_{1,n}x_n \\ y_2 &= a_{2,1}x_1 \oplus a_{2,2}x_2 \oplus \cdots \oplus a_{2,n}x_n \\ &\cdots \\ y_p &= a_{p,1}x_1 \oplus a_{p,2}x_2 \oplus \cdots \oplus a_{p,n}x_n \end{aligned}$$

Note that $z_1, z_2, \ldots, z_q$ are output functions of block G in Fig. 4.1.

Theorem 4.1: To represent a partially defined function $(F_1, F_2, \ldots, F_m)$, at least $q = \lceil \log_2 m \rceil$ compound variables are necessary.

Definition 4.2: Given an incompletely specified function f, the linear transformation that minimizes the number of compound variables p is the **optimal transformation**.

When the number of the compound variables can be reduced to $\lceil \log_2 m \rceil$ by a linear transformation, the transformation is optimum by Theorem 4.1.

V. Reduction of Compound Variables

This part shows that the number of the variables can be reduced by a linear transformation, when a partially defined function satisfies a certain condition [17].

Definition 5.1: A classification function is **reducible** if it can be represented with fewer variables than the original ones when a linear decomposition is used. Otherwise, the function is **irreducible**.

Lemma 5.1: An n-variable partially defined function $f = (F_1, F_2, \ldots, F_m)$ is reducible if and only if there exists a non-zero vector $\vec{u}$ that satisfies the condition:

$$\vec{u} \in B^n - D_f,$$

where D_f is the set of difference vectors of f.

(Proof $\Leftarrow$) Without loss of generality, we can assume that there exist a vector $\vec{u}$ such that $x_1 = 1$. Let $S(F_i, x_1 = a)$, $(i = 1, 2, \ldots, m)$ be the sets of vectors in F_i such that $x_1 = a$, where $a \in \{0, 1\}$. Next, consider the following $2m$ sets of vectors:

$$\begin{aligned} A_i &= S(F_i, x_1 = 0) \\ B_i' &= \{\vec{b} \oplus \vec{u} \mid \vec{b} \in S(F_i, x_1 = 1)\} \end{aligned}$$

In this case, f can be represented with $n-1$ compound variables. The first bits of A_i, $(i = 1, \ldots, m)$ are all 0's. Also, the first bits of B_i', $(i = 1, \ldots, m)$ are all 0's. Thus, the first bit of each vector is 0 in these $2m$ sets of vectors.

From the property of registered vectors, all the vectors in A_i, $(i = 1, \ldots, m)$ are distinct. Similarly, all the vectors in B_i', $(i = 1, \ldots, m)$ are distinct.

Next, we will show that all the vectors in A_i and B_j', $(i \neq j)$ are distinct.

On the contrary, assume that a vector $\vec{a}$ in A_i, and a vector $\vec{b'}$ in B_j' are equal. Then, we have the relation $\vec{a} = \vec{b'}$. Since $\vec{b'} = \vec{b} \oplus \vec{u}$, we have a relation $\vec{a} = \vec{b} \oplus \vec{u}$, and the relation $\vec{u} = \vec{a} \oplus \vec{b}$. However, this contradicts the condition of $\vec{u}$. Thus, any vector in A_i and any vector in B_j', $(i \neq j)$ are different. To summarize, the first bits of the vectors in $A_1, A_2, \ldots, A_m, B_1', B_2', \ldots B_m'$ are all 0's, and all the vectors in the sets $A_1 \cup B_1', A_2 \cup B_2', \ldots A_m \cup B_m'$ are distinct.

Adding the vector $\vec{u}$ to an element in the set $S(F_i, x_1 = 1)$ corresponds to perform a linear transformation to $S(F_i, x_1 = 1)$. Thus, to represent f, the first bit x_1 is not necessary, and the remaining $n-1$ compound variables are sufficient to represent the function.

(Proof $\Rightarrow$) We prove by contraposition. Since $B^n = D_f \cup \vec{0}$, for any non-zero $\vec{u}$ that we chose in the operation of the proof $\Leftarrow$ of Lemma 5.1, we have that $\vec{u} = \vec{a} \oplus \vec{b}$, where $\vec{a} \in F_i$ and $\vec{b} \in F_j$, and $i \neq j$. Since $\vec{u}$ is a non-zero vector, without loss of generality, we can assume that $x_1 = 1$ in $\vec{u}$. Hence, in $\vec{a}$, $x_1 = 0$ and in $\vec{b}$, $x_1 = 1$ (or vice-versa), which means that $\vec{a} \in A_i$ and $\vec{a} = \vec{b'} = \vec{u} \oplus \vec{b} \in B_j'$, implying that $\vec{u}$ cannot be used to reduce variable x_1. Thus, we cannot reduce any variable. □

Theorem 5.1: An n-variable function $(F_1, F_2, \ldots, F_m)$ is reducible if and only if the set of difference vectors for f contains fewer than $2^n - 1$ elements.

(Proof) This is a direct consequence of Lemma 5.1. □

Theorem 5.2: A classification function $(F_1, F_2, \ldots, F_m)$ can be represented with $r = \lceil \log_2(S+1) \rceil$ compound variables, where

$$S = \sum_{(i<j)} k_i k_j,$$

$i, j \in \{1, 2, \ldots, m\}$. and $k_i = |F_i|$.

(Proof) Note that S denotes an upper bound on the number of difference vectors. By Theorem 5.1, if $S < 2^n - 1$, then we can reduce one variable. By reducing variables repeatedly, we have $2^r > S + 1$, and $r > \log_2(S+1)$. When $S + 1 \neq 2^p$, we can reduce the number of variables up to $r = \lceil \log_2(S+1) \rceil$. When $S + 1 = 2^p$, we can show that the function can be represented with p variables. □

Theorem 5.3: In an n-variable decision function $f : D \to \{1, 2\}$, where $D \subseteq B^n$, let h_a be the number of registered vectors in D whose j-th bit is $a \in \{0, 1\}$. If $h_0 h_1 < 2^{n-1}$, then f can be represented with $n - 1$ variables.

(Proof) The number of difference vectors whose j-th bit is 1, is at most $h_0 h_1$. Since $h_0 h_1 < 2^{n-1}$, at least one non-zero vector exists that is missing in the set of difference vectors D_f. Thus, $|D_f| < 2^r - 1$, and f is reducible by Theorem 5.1. □

VI. Symmetric Decision Function

Lemma 6.1: Consider the n-variable decision function:

$$\begin{aligned} f(\vec{e}_i) &= 1 \ (i = 1, 2, \ldots, n) \\ f(\vec{0}) &= 2, \end{aligned}$$

where $\vec{e}_i$ denotes the unit vector where only the i-th component is 1. To represent f, we need n primitive variables if we can use only the primitive variables. If we can use a compound variable: $y = x_1 \oplus x_2 \oplus \cdots \oplus x_n$, then the function can be represented by only one variable.

(Proof) By Definition 3.1, f depends on all the variables. Thus, all the variables are necessary. □

As shown in the previous lemma, the number of variables can be reduced drastically when a linear transformation can be used to represent a symmetric classification function.

Lemma 6.2: Consider the $n = 2s$-variable symmetric decision function:

$$\begin{aligned} f(\vec{x}) &= 1, \ (\text{when } |\vec{x}| = \lfloor n/2 \rfloor - 1), \\ f(\vec{x}) &= 2, \ (\text{when } |\vec{x}| = \lfloor n/2 \rfloor + 1). \end{aligned}$$

This function can be represented with $n-1$ primitive variables. However, even if compound variables can be used, the number of variables cannot be reduced to $n - 2$ or less.

Consider the $n = 2s + 1$-variable symmetric decision function:

$$\begin{aligned} f(\vec{x}) &= 1, \ (\text{when } |\vec{x}| = \lfloor n/2 \rfloor - 1), \\ f(\vec{x}) &= 2, \ (\text{when } |\vec{x}| = \lfloor n/2 \rfloor + 2). \end{aligned}$$

This function can be represented with $n-2$ primitive variables. However, even if compound variables can be used, the number of variables cannot be reduced to $n - 3$ or less.

Example 6.1: Consider the case of $n = 8$. We have

$$\begin{aligned} f(\vec{x}) &= 1, \ (\text{when } |\vec{x}| = 3), \\ f(\vec{x}) &= 2, \ (\text{when } |\vec{x}| = 5). \end{aligned}$$

The number of vectors $\vec{x}$ such that $f(\vec{x}) = 1$ is $\binom{8}{3} = 56$, the number of vectors $\vec{x}$ such that $f(\vec{x}) = 2$ is $\binom{8}{5} = 56$. When we omit the the first bit, the set of difference vectors contains $2^7 - 1 = 127$ elements. Thus, to represent the function f, we need 7 compound variables, by Theorem 5.1.

TABLE 7.1
4-VARIABLE PARTIALLY DEFINED DECISION FUNCTION.

input s				f	TAG
x_1	x_2	x_3	x_4		
1	1	1	1	1	1
1	0	1	1	1	2
1	0	0	0	1	3
0	0	1	0	1	4
1	0	1	0	2	5
0	1	1	1	2	6
0	1	1	0	2	7
0	0	0	1	2	8
0	0	1	1	2	9

Consider the case of $n = 7$. We have

$$\begin{aligned} f(\vec{x}) &= 1, \ (\text{when } |\vec{x}| = 2), \\ f(\vec{x}) &= 2, \ (\text{when } |\vec{x}| = 5). \end{aligned}$$

The number of vectors $\vec{x}$ such that $f(\vec{x}) = 1$ is $\binom{7}{2} = 21$, the number of vectors $\vec{x}$ such that $f(\vec{x}) = 2$ is $\binom{7}{5} = 21$. When we omit the first two bits, the set of difference vectors contains $2^5 - 1 = 31$ elements. Thus, to represent the function f, by Theorem 5.1, we need 5 compound variables. ■

VII. Examples of Reduction Method

Example 7.1: Table 7.1 shows a 4-variable partially defined decision function f. Among $2^4 = 16$ possible combinations, the value 1 is assigned to 4 combinations, while the value 2 is assigned to 5 combinations. Table 7.2 shows the set of difference vectors D_f of f. The TAG part of the table shows the pair of vectors that produced the difference vector. The set of difference vectors D_f contains all the unit vectors: $(1,0,0,0), (0,1,0,0), (0,0,1,0)$ and $(0,0,0,1)$. Thus, all the variables are essential. Hence, if no compound variable are used, to represent the function f all the primitive variables are necessary.

TABLE 7.2
SET OF DIFFERENCE VECTORS DF.

x_1	x_2	x_3	x_4	TAG
0	0	0	1	(2, 5)
0	0	1	0	(3, 5)
0	0	1	1	(4, 8)
0	1	0	0	(4, 7)
0	1	0	1	(1, 5)
1	0	0	0	(1, 6)
1	0	0	1	(1, 7)
1	0	1	0	(2, 8)
1	0	1	1	(3, 9)
1	1	0	0	(1, 9)
1	1	0	1	(2, 7)
1	1	1	0	(1, 8)
1	1	1	1	(3, 6)

Next, check if the number of variables can be reduced by using a linear transformation. Note that the vector $\vec{u} = (0, 1, 1, 0)$ is missing in the set of difference vectors. With this vector, we can reduce the number of variables. Here, we try to reduce the variable x_2. Consider the linear transformation $y_3 = x_3 \oplus x_2$.

TABLE 7.3
MODIFIED REGISTERED VECTOR TABLE.

x_1	y_3	x_4	g	TAG
1	0	1	1	1
1	1	1	1	2
1	0	0	1	3
0	1	0	1	4
1	1	0	2	5
0	0	1	2	6
0	0	0	2	7
0	0	1	2	8
0	1	1	2	9

Table 7.3 shows the modified registered vector table. Note that vectors with TAGs 1, 6, and 7 are modified by $\vec{u} = (0, 1, 1, 0)$. In Table 7.3, consider the set H_1 that consists of vectors for $g = 1$, and the set H_2 that consists of vectors for $g = 2$. Note that H_1 and H_2 are disjoint. Thus, the function f can be represented with the variables x_1, y_3 and x_4. In Table 7.3, the vectors that correspond to TAG6 and TAG8 are the same. Also, as for the registered vectors, the following relation holds: $f(x_1, x_2, x_3, x_4) = g(x_1, y_3, x_4),\ y_3 = x_2 \oplus x_3$. ■

To minimize primitive variables, we can use the set of minimal difference vectors. However, to minimize the compound variables, we have to use the set of all the difference vectors.

Example 7.2: In the world, there are 197 independent countries [3]. The continent of these countries can be classified into $m = 6$ areas: Europe, Asia, Africa, North America, South America, and Oceania.

Now, consider the classification function that produces the area for each country. The names of the countries are described by alphabets (26 characters), blank and hyphen(-). The country with the longest name is "Saint Vincent and the Grenadines," having 32 characters. However, all the countries can be distinguished by the first 12 characters.

Thus, to show the countries, we used the first 12 characters. All the upper case letters are converted into lower case letters, and all the characters are coded by 5 bits. For the countries whose name have less than 12 characters, blank characters were appended to make the lengths of all the names 12.

In the original classification function, the number of inputs is $n = 5 \times 12 = 60$, the number of outputs is $m = 6$ (one-hot code), and the number of registered vectors is $k = 197$. Among these, Armenia, Azerbaijan, Cyprus, Georgia, Kazakhstan, Turkey, Russia, are classified to Europe.

1) When we minimized the number of primitive variables by Algorithm 3.1, we had a 12-variable solution.
2) When we applied the iterative minimization using Lemma 5.1 to the result of 1), we had a 10-variable solution. In this case, the maximum compound degree was three.
3) When we directly applied the iterative minimization using Lemma 5.1, we had a 9-variable solution. In this case, the maximum compound degree was 11.

■

Table 7.4 summarizes the experimental results. The column headed with $AL3.1$ denotes the number of the primitive variables obtained by Algorithm 3.1. The column headed with $AL3.1 + TH5.1$ denotes the number of the compound variables obtained by Algorithm 3.1 and by the iterative algorithm using Lemma 5.1. The column headed with $TH5.1$ denotes the number of the compound variables obtained by the iterative algorithm using Lemma 5.1. The number in the parenthesis shows the maximum compound degree.

TABLE 7.4
MINIMIZATION RESULTS FOR EXAMPLES 7.2-7.5.

Example	n	m	k	AL3.1	$AL3.1 + TH5.1$	$TH5.1$
7.2	60	6	197	12 (1)	10 (3)	9 (11)
7.3	60	7	118	10 (1)	9 (2)	8 (12)
7.4	30	9	3700	20 (1)	18 (3)	18 (4)
7.5	30	4	4000	20 (1)	18 (3)	18 (3)

The number in the parenthesis denotes the maximum compound degree.

Example 7.3: Up to now, 118 chemical elements are known [5]. These chemical elements can be classified into $m = 7$ periods according to their number of protons. For example, Hydrogen is in the first period; Lithium is in the second period; Sodium is in the third period; Potassium is in the fourth period; Rubidium is in the fifth period; Caesium is in the sixth period; and Francium is in the seventh period.

Consider the classification function that gives the corresponding period for a given name of a chemical element. The name of the chemical elements is represented by 13 alphabet characters. However, all the elements can be distinguished by the first 12 characters. Thus, in the classification functions, we consider only the first 12 characters. Similarly to Example 7.2, each character is coded by 5 bits. In the original classification function, the number of inputs is $n = 5 \times 12 = 60$, the number of outputs is $m = 7$ (one-hot code), and the number of registered vectors is $k = 118$. We minimized the variables in the same way as in Example 7.2. The second row of Table 7.4 shows the experimental results. ■

Example 7.4: Consider the classification function, where the inputs are the telephone numbers of 3700 Japanese companies [19], and the outputs show the names of the stock exchange. There are $m = 9$ different stock exchange. 1) Tokyo 1st; 2) Tokyo 2nd; 3) Tokyo Mothers; 4) Sapporo; 5) Nagoya; 6) Fukuoka; 7) JASDAQ; 8) REIT; and 9) Foreign.

The telephone numbers are represented by 9-digit decimal numbers. We converted them to binary numbers of 30-bits. Thus, the number of inputs is $n = 30$. Also, the number of outputs is 9 (one-hot code), and the number of registered vectors is $k = 3700$. We minimized the variables in the same way as in Example 7.2. The third row of Table 7.4 shows the experimental results. ■

Example 7.5: We generated 4000 distinct random vectors of 30 bits, and partitioned them into $m = 4$ sets, each consists of 1000 vectors. From this, we made a classification function with $n = 30$ inputs, four outputs, and 4000 registered vectors. We minimized the variables in the same way as in Example 7.2. The last row of Table 7.4 shows the experimental results. ■

TABLE 8.1
REGISTERED VECTOR

x_1	x_2	x_3	f
0	1	1	1
0	0	0	1
0	0	1	1
1	1	0	2
1	0	0	2
0	1	0	2

Example 7.5 can be used as a packet filter of the internet.

VIII. EXACT MINIMIZATION OF COMPOUND VARIABLES

When we apply Lemma 5.1 repeatedly, the number p of the compound variables can be reduced. However, with this method, we may fail to find the solution with the exact minimum number of compound variables. To obtain an exact minimum solution, we have to consider all possible linear transformations. Here, we show a method using a SAT (satisfiability decision) solver [4], [15].

Theorem 8.1: A partially defined classification function can be represented with p compound variables:

$$\begin{aligned} y_1 &= a_{1,1}x_1 \oplus a_{1,2}x_2 \oplus \cdots \oplus a_{1,n}x_n \\ y_2 &= a_{2,1}x_1 \oplus a_{2,2}x_2 \oplus \cdots \oplus a_{2,n}x_n \\ &\cdots \\ y_p &= a_{p,1}x_1 \oplus a_{p,2}x_2 \oplus \cdots \oplus a_{p,n}x_n \end{aligned}$$

if and only if the value of $(y_1, y_2, \ldots, y_p)$ is non-zero for all the difference vectors.

(Proof) The given function can be represented with $(y_1, y_2, \ldots, y_p)$ if and only if the values of $(y_1, y_2, \ldots, y_p)$ are distinct for all the registered vectors that produce distinct functional values.

Suppose that the value $(y_1, y_2, \ldots, y_p)$ is non-zero for any difference vector $\vec{d} \in D_f$. Since $\vec{d} = \vec{a}_i \oplus \vec{a}_j$ is a difference vectors, $f(\vec{a}_i) \neq f(\vec{a}_j)$. Thus, for any pair of registered vectors $\vec{a}_i$ and $\vec{a}_j$ $(i \neq j)$ such that $\vec{d} = \vec{a}_i \oplus \vec{a}_j$, the function h_j $(j = 1, 2, \ldots q)$ take distinct values. This means that the functions $(y_1, y_2, \ldots, y_m)$ must take distinct values for these registered vectors. Thus, the function can be represented by $(y_1, y_2, \ldots, y_p)$.

Suppose that the value $(y_1, y_2, \ldots, y_p)$ is zero for some difference vector $\vec{d} \in D_f$. Let $\vec{d} = \vec{a}_i \oplus a_j$, where $\vec{a}_i$ and $\vec{a}_j$ are registered vectors that produce distinct functional values. Then, we have

$$y_t(\vec{a}_i) = y_t(\vec{a}_j), \ (t = 1, 2, \ldots, p).$$

Thus, $\vec{a}_i$ and $\vec{a}_j$ produces the same function values. This means that the function cannot be represented by $(y_1, y_2, \ldots, y_p)$. □

Example 8.1: The three-variable decision function shown in Table 8.1 requires three variables even if any linear transformations is used. To prove this, assume that this function could be represented with two compound variables:

$$\begin{aligned} y_1 &= a_{1,1}x_1 \oplus a_{1,2}x_2 \oplus a_{1,3}x_3 \\ y_2 &= a_{2,1}x_1 \oplus a_{2,2}x_2 \oplus a_{2,3}x_3 \end{aligned}$$

TABLE 8.2
THE SET OF DIFFERENCE VECTORS DF

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

Table 8.2 shows the set of difference vectors. The values of the vector (y_1, y_2) for the difference vectors are:

1) $(a_{1,1}, a_{2,1})$
2) $(a_{1,2}, a_{2,2})$
3) $(a_{1,3}, a_{2,3})$
4) $(a_{1,1} \oplus a_{1,2}, a_{2,1} \oplus a_{2,2})$
5) $(a_{1,1} \oplus a_{1,3}, a_{2,1} \oplus a_{2,3})$
6) $(a_{1,2} \oplus a_{1,3}, a_{2,2} \oplus a_{2,3})$
7) $(a_{1,1} \oplus a_{1,2} \oplus a_{1,3}, a_{2,1} \oplus a_{2,2} \oplus a_{2,3})$

If there exists an assignment of $a_{i,j}$ that makes these 7 vectors all non-zero, then the function can be represented with two compound variables. Otherwise, the function cannot be represented with two compound variables. We can use a SAT solver to prove that there is no such assignment. That is, to represent this function, at least three compound variables are necessary. ■

Algorithm 8.1: (Exact Minimization of Compound Variables)

1) By applying Lemma 5.1 repeatedly, reduce the number of compound variables to p. Save this solution.
2) By using Theorem 8.1, check if the current partially defined function can be represented by $p-1$ compound variables.
3) If no solution exist, then output the solution with p compound variables, and stop. If the solution exist, then save this solution, $p \leftarrow p-1$, and go to Step 2.

Example 8.2: The 3-variable decision function f shown in Table 8.1 requires three compound variables. As shown in Table 8.2, the set of difference vectors of f contains $2^3 - 1$ elements. From Theorem 5.1, at least three variables are necessary. ■

IX. CONCLUSION

In this paper, we showed methods to reduce the number of variables for partially defined classification functions. The major contributions are:

- It shows a minimization method of primitive variables.
- It shows a reduction method for compound variables using a linear transformation.
- It shows an exact minimization method for compound variables using a linear transformation.

In [18], circuits for handwritten digit recognition is designed using classification functions. In these circuits, the number of variables is $n = 196$, the number of classes is $m = 10$, and the total number of registered vectors is $k = 58191$.

ACKNOWLEDGMENTS

This research is partly supported by Grant in Aid for Scientific Research of the Japan Society for the Promotion of Science (JSPS). The author thanks the reviewer who improved the proof of Lemma 5.1.

REFERENCES

[1] D. Angluin and C. H. Smith, "Inductive inference: Theory and methods," *Computing Surveys,* Vol. 15, No. 3, Sept. 1983, pp. 327-369.

[2] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Occam's razor," *Information Processing Letters*, Vol. 24, Issue 6, 1987, pp. 377-380.

[3] https://www.countries-ofthe-world.com/all-countries.html

[4] N. Een and N. Sorensson,"An extensible SAT-solver," *Proc. 6th Int. Conf. on Theory and Applications of Satisfiability Testing* (SAT-2003), pp.502-518, 2003.

[5] https://en.wikipedia.org/wiki/Periodic_table

[6] T. Ibaraki,"Partially defined Boolean functions," Chapter 8 in: Y. Crama and P. L. Hammer, *Boolean Functions - Theory, Algorithms and Applications*, Cambridge University Press, New York, 2011.

[7] J. Kuntzmann, *Algèbre de Boole*, Dunod, Paris, 1965. English translation: *Fundamental Boolean Algebra*, Blackie and Son Limited, London and Glasgow, 1967.

[8] R. J. Lechner, "Harmonic analysis of switching functions,"in A. Mukhopadhyay (ed.), *Recent Developments in Switching Theory*, Academic Press, New York, 1971.

[9] E. I. Nechiporuk, "On the synthesis of networks using linear transformations of variables,"*Dokl. AN SSSR*, vol. 123, no. 4, Dec. 1958, pp. 610-612 (in Russian).

[10] T. Sasao, *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, 1999.

[11] T. Sasao, "On the number of variables to represent sparse logic functions,"*ICCAD-2008*, San Jose, California, USA, Nov. 10-13, 2008, pp. 45-51.

[12] T. Sasao, *Memory-Based Logic Synthesis*, Springer, 2011.

[13] T. Sasao, "Linear decomposition of index generation functions," *17th Asia and South Pacific Design Automation Conference* (ASPDAC-2012), 2012, Sydney, Australia, pp. 781-788.

[14] T. Sasao, "Index generation functions: Tutorial,"*Journal of Multiple-Valued Logic and Soft Computing*, Vol. 23, No. 3-4, pp. 235-263, 2014.

[15] T. Sasao, I. Fumishi, and Y. Iguchi, "A method to minimize variables for incompletely specified index generation functions using a SAT solver," *International Workshop on Logic and Synthesis*, Mountain View, June 12-13, 2015.

[16] T. Sasao, "Index generation functions: Minimization methods," *International Symposium on Multiple-Valued Logic* (ISMVL-2017) (invited), Novi Sad, Serbia, May 2017, pp. 197-206.

[17] T. Sasao, *Index Generation Functions*, Morgan & Claypool, Oct. 2019.

[18] T. Sasao, Y. Horikawa, and Y. Iguchi, "Handwritten digit recognition based on classification functions," *International Symposium on Multiple-Valued Logic*, (ISMVL-2020), May 20-22, 2020, Miyazaki, Japan

[19] https://str.toyokeizai.net/magazine/shikiho_cd/

[20] D. A. Simovici, M. Zimand, and D. Pletea,"Several remarks on index generation functions,"*International Symposium on Multiple-Valued Logic* (ISMVL-2012), Victoria, Canada, May 2012, pp. 179-184.