On the Numbers of Variables to Represent Multi-Valued Incompletely Specified Functions

Tsutomu Sasao
Department of Computer Science and Electronics,
Kyushu Institute of Technology,
Iizuka 820-8502, Japan

Abstract—In an incompletely specified function \( f \), don’t care values can be chosen to minimize the number of variables to represent \( f \). We consider incompletely specified functions \( f : P^n \rightarrow Q \), where \( P = \{0,1,\ldots,p-1\}, Q = \{0,1,\ldots,q-1\} \), \( u \) combinations are mapped to \( i \) (\( i = 0,1,\ldots,q-1 \)), \( uq = k \), and other combinations are mapped to don’t cares. We show that most functions can be represented with \( 2 \lceil \log_q(k+1) \rceil \) variables or less. Experimental results are shown to support this.

I. INTRODUCTION

For completely specified logic functions, logic minimization is a process of reducing the number of products to represent the given function. However, for incompletely specified functions (i.e., functions with don’t cares), at least two problems exist [5]: The first is to reduce the number of the products to represent the function, and the second is to reduce the number of variables. The first problem is useful for sum-of-products expression (SOP)-based realizations [2], while the second problem is useful for memory-based realizations.

Example 1.1: Consider the four-variable function shown in Fig. 1.1, where the blank cells denote don’t cares. The SOP with the minimum number of products is \( F_1 = x_1x_4 \lor x_2x_3 \), while the SOP with the minimum number of variables is \( F_2 = x_1x_2 \lor x_1x_4 \lor x_2x_4 \). Note that \( F_1 \) has two products and depends on four variables, while \( F_2 \) has three products and depends on only three variables. \( x_3 \) is a non-essential variable, since \( F_2 \) does not include it.

In this paper, we consider the minimization of the number of variables. Especially, we are interested in the number of variables to represent logic functions whose values are specified for \( k \) combinations, where \( k \) is small. Due to the space limitation, all the proofs are omitted.

II. DEFINITIONS AND BASIC PROPERTIES

Definition 2.1: A multi-valued incompletely specified logic function \( f \) is a mapping \( D \rightarrow Q \), where \( D \subset P^n \), \( P = \{0,1,\ldots,p-1\}, Q = \{0,1,\ldots,q-1\} \).

Definition 2.2: \( f \) depends on \( x_i \) if there exists a pair of vectors

\[
\vec{a} = (a_1,a_2,\ldots,a_i,\ldots,a_n) \quad \text{and} \quad \vec{b} = (a_1,a_2,\ldots,b_i,\ldots,a_n),
\]

such that both \( f(\vec{a}) \) and \( f(\vec{b}) \) are specified, and \( f(\vec{a}) \neq f(\vec{b}) \). If \( f \) depends on \( x_i \), then \( x_i \) is essential in \( f \), and \( x_i \) must appear in every expression for \( f \).

Definition 2.3: Two functions \( f \) and \( g \) are compatible when the following condition holds: For any \( \vec{a} \in P^n \), if both \( f(\vec{a}) \) and \( g(\vec{a}) \) are specified, then \( f(\vec{a}) = g(\vec{a}) \).

Lemma 2.1: Let \( f_i = f([x = i]) \) for \( i = 0,1,\ldots,p-1 \). Then, \( x \) is non-essential in \( f \) if \( f_i \) and \( f_j \) are compatible for all the pair \( (i,j) \).

Example 2.1: Consider the function \( f \) in Fig. 1.1. It is easy to verify that \( x_1, x_2, \) and \( x_4 \) are essential. However, \( x_3 \) is non-essential. In fact, \( f \) is represented as

\[
f = x_1x_2 \lor x_2x_4 \lor x_1x_4.
\]

Essential variables must appear in every expression for \( f \), while non-essential variables may appear in some expressions and not in others. Algorithms to represent a given function by using the minimum number of variables have been considered [1], [3], [4], [5].

III. ANALYSIS FOR TWO-VALUED OUTPUT FUNCTIONS

In this section, we derive the number of variables to represent \( p \)-valued input two-valued output incompletely specified functions. In the analysis that follows, we consider a set of functions (e.g., all incompletely specified functions) restricted by conditions (e.g. the number of care values is \( k = 2u \)).

Definition 3.1: A set of functions is uniformly distributed, if the probability of occurrence of any function is the same as any other function.

For example, the set of two-valued input two-valued output 4-variable incompletely specified functions with 1 care value...
consists of 32 members, 16 having a single 1 and 16 having a single 0. If the functions are uniformly distributed, the probability of the occurrence of any one of them is \( \frac{1}{32} \).

**Theorem 3.1:** Consider a set of uniformly distributed \( p \)-valued \( n \)-variable input two-valued output incompletely specified function, where \( u \) combinations are mapped to 0, \( u \) combinations mapped to 1, and the other \( p^n - 2u \) combinations are mapped to don’t cares. Let \( \eta \) be the probability that \( f(x_1, x_2, \ldots, x_n) \) can be represented by using only \( x_1, x_2, \ldots, x_{t-1}, \) and \( x_t \), where \( t < n \). Then, \( \eta > (1 - \alpha)^{n-u} \), where \( \alpha = \frac{u}{p^n} \).

**Theorem 3.2:** Consider a set of uniformly distributed incompletely specified function, where \( u \) combinations are mapped to 0, \( u \) combinations mapped to 1, and the other \( p^n - 2u \) combinations are mapped to don’t cares. Let \( PR \) be the probability that \( f(x_1, x_2, \ldots, x_n) \) can be represented by using only \( t \) variables. Then, \( PR = 1 - \sigma(\frac{t}{n}) \), where \( \sigma = 1 - \eta \), and \( \eta \) is the probability that \( f(x_1, x_2, \ldots, x_n) \) can be represented by using only \( x_1, x_2, \ldots, x_{t-1}, \) and \( x_t \).

From this, we have the following:

**Conjecture 4.1:** Consider a set of uniformly distributed functions of \( n \) variables, where \( u \) combinations are mapped to 0, \( u \) combinations mapped to 1, and the other \( p^n - 2u \) combinations are mapped to don’t cares.

**Example 4.1:** Table 4.1 shows a registered vector table consisting of 6 vectors. When no entry matches the input vector, the function produces 0. Consider the decomposition chart shown in Table 4.2. In Table 4.2, \( x_1, x_2, \) and \( x_3 \) specify the columns, and \( x_4 \) and \( x_5 \) specify the rows, and blank elements denote don’t cares. Note that in Table 4.2, each column has at most one care element. Thus, the function can be represented by only the column variables: \( x_1, x_2, \) and \( x_3 \).

From here, we obtain the probability of such a condition by a statistical analysis.

**Theorem 4.1:** Consider a set of uniformly distributed \( p \)-valued input incompletely specified index generation functions \( f(x_1, x_2, \ldots, x_n) \) with weight \( k \), where \( p \leq k < p^{n-2} \). Let \( \eta(k) \) be the probability that \( f \) can be represented with \( x_1, x_2, \ldots, x_n, \) where \( t < n \). Then, \( \eta(k) \approx \exp(-\frac{k^2}{2p}) \).

The above theorem shows the case when the input variables are removed without considering the property of the function. In practice, we can remove the maximum number of non-essential variables by an optimization program.

**Theorem 4.2:** Consider a set of uniformly distributed incompletely specified index generation functions \( f(x_1, x_2, \ldots, x_n) \) with weight \( k \), where \( p \leq k < p^{n-2} \). Let \( PR \) be the probability that \( f \) can be represented with \( t \) variables, then \( PR = 1 - (1 - \eta(k))^t \), where \( \eta(k) \) is the probability that \( f \) can be represented with \( x_1, x_2, \ldots, x_t \).

From this, we have the following:

**Conjecture 4.1:** Consider a set of uniformly distributed incompletely specified \( p \)-valued input \( n \)-variable index generation functions with weight \( k \). If

\[
t \geq 2 \log_p k - \log_p 4.158,
\]
then more than 95% of the functions can be represented with \( t \) variables.
Note that there exist functions that require all the variables as shown below. However, the fraction of such functions approaches to zero as $n$ increase.

**Example 4.2:** Consider the $n$-variable incompletely specified index generation function $f$ with weight $k = n + 1$ and $p = 2$:

\[
\begin{align*}
    f(1, 0, 0, \ldots, 0, 0) &= 1 \\
    f(0, 1, 0, \ldots, 0, 0) &= 2 \\
    f(0, 0, 1, \ldots, 0, 0) &= 3 \\
    \vdots \\
    f(0, 0, 0, \ldots, 1, 0) &= n - 1 \\
    f(0, 0, 0, \ldots, 0, 1) &= n \\
    f(0, 0, 0, \ldots, 0, 0) &= n + 1 \\
    f(a_1, a_2, a_3, \ldots, a_{n-1}, a_n) &= d \quad \text{(for other combinations)}.
\end{align*}
\]

In this function, all the variables are essential, and no variable can be removed.

**Theorem 4.3:** To represent an incompletely specified index generation function with weight $k$, at least $\lceil \log_p (k + 1) \rceil$ variables are necessary.

V. EXPERIMENTAL RESULTS

**A. Random Single-Output Functions**

For different $n$ and $p$, we randomly generated 100 functions, where $u$ combinations are mapped to 0, $u$ combinations are mapped to 1, and the remaining $p^n - 2u$ combinations are mapped to don’t cares. We minimized the number of variables by an exact optimization algorithm [7]. In Table 5.1, the columns headed with $\text{Exp}$ show average numbers of variables to represent $p$-valued input two-valued output functions, where the set of variables are selected by the optimization algorithm. The values are the average of 100 randomly generated functions. The columns headed by $\text{Conj}$ show the numbers of variables to represent incompletely specified functions given by Conjecture 3.1. For example, when $p = 2$ and $n = 20$, functions whose 31 minterms are mapped to zeros, 31 minterms are mapped to ones, and the other minterms are mapped to don’t cares, require, on the average, 6.98 variables to represent the functions. On the other hand, Conjecture 3.1 shows that 8 variables are sufficient. We performed additional experiments for $p = 3$, 5, and 7 and confirmed that the Conjecture.

VI. APPLICATIONS

**A. Two-Valued Case**

A terminal access controller (TAC) for a local area network checks whether the requested terminal has permission to access Web, e-mail, FTP, Telnet, or not. Each terminal has its unique MAC address represented by 48 bits. Note that the table for the terminal access controller must be updated frequently.

**Example 6.1:** Let the number of terminals to be connected to a TAC be at most 255. To implement the TAC, we use an IGU and a memory. The number of inputs for the index generation unit (IGU) is 48 and the number of outputs is 8. Fig. 6.1 shows the IGU. When the number of registered vectors is 255, we need about 12 variables to distinguish these vectors. Let $t = 15$ be the number of inputs, and $m = 8$ be the number of outputs of main memory. The size of the main memory is $2^{15} \times 8 = 2^{18} = 256 \times 2^{10}$. The size of the AUX memory is $2^{8} \times (48 - 15) = 256 \times 33 = 8448$. Note that, in many cases, only 12 inputs are necessary to distinguish the MAC addresses, but 15 inputs are used to cover more addresses.

**B. Four-Valued Case**

Deoxyribonucleic acid (DNA) contains the genetic instructions used in the development and functioning of all known
functions can be represented by at most $2^\lceil \log_p(k + 1) \rceil$ variables, in most cases. These results show that reduction of the number of variables is quite effective for incompletely specified functions.

ACKNOWLEDGMENTS

This work was supported in part by the Regional Innovation Cluster Program (Global Type, Second Stage).

REFERENCES


VII. CONCLUSIONS

In this paper, we have derived the number of variables to represent incompletely specified $p$-valued two-valued output functions and index generation functions with weight $k$. Such