Algorithm of Superposition of Boolean Functions Given with Truth Vectors

Anatoly Plotnikov[1], Alexander Petrov[2], Anton Petrov[3]

[1] Department of Computer Systems and Networks, Volodymyr Dalh East-Ukrainian National University Luhansk, 91034, Ukraine

[2] Department Applied Informatics, AGH University of Science and Technology Krakow, Poland
Department of Computer Systems and Networks, the Dalh East-Ukrainian national university

[3] Department of Computer Systems and Networks, Volodymyr Dalh East-Ukrainian National University Luhansk, 91034, Ukraine

Abstract

In this paper are examined the practical problems of construction of arbitrary superposition of Boolean functions when all functions are given with truth vectors.

Keywords: *Boolean Function, True Vector, Truth Table, Superposition.*

1. Problem statement

Let there be a set E = {0;1}. Then the mapping of *f*:

$$E^n \to E$$

is called a Boolean function.

A Boolean function $f(x_1, x_2, ..., x_n) \in E$ ($x_i \in E$, i = *1,2,...,n*) can be completely defined with truth table:

Table 1: Truth table

x_1	x_2	…	x_{n-1}	x_n	$f(x_1, x_2, ..., x_{n-1}, x_n)$
0	*0*	…	*0*	*0*	*f(0,0,…,0,0)*
0	*0*	…	*0*	*1*	*f(0,0,…,0,1)*
0	*0*	…	*1*	*0*	*f(0,0,…,1,0)*
0	*0*	…	*1*	*1*	*f(0,0,…,1,1)*
…	…	…	…	…	…
1	*1*	…	*1*	*0*	*f(1,1,…,1,0)*
1	*1*	…	*1*	*1*	*f(1,1,…,1,1)*

The first columns of this table contain the lexicographically ordered value sets of Boolean variables and the last column of this table is the value of the given function of every set. This last column is called truth vector of the Boolean function $f(x_1, x_2, ..., x_n)$. It is obvious that it is not necessary to write all 2^n sets of values of Boolean variables for determining the Boolean function. It is enough to submit the truth row that corresponds with it.

Example 1. Let the Boolean function of three variables be represented with the truth vector: (10110010). Then we can write the appropriate truth table:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	*0*	*0*	*1*
0	*0*	*1*	*0*
0	*1*	*0*	*1*
0	*1*	*1*	*1*
1	*0*	*0*	*0*
1	*0*	*1*	*0*
1	*1*	*0*	*1*
1	*1*	*1*	*0*

Let there be a system of Boolean functions:

$$\begin{cases} f(x_1, x_2, ..., x_n) \\ f_1(x_1, x_2, ..., x_m) \\ f_2(x_1, x_2, ..., x_m) \\ ..... \\ f_n(x_1, x_2, ..., x_m) \end{cases} \quad (1)$$

Then the function

$$f(f_1(x_1, x_2, ..., x_m), f_2(x_1, x_2, ..., x_m), ..., f_n(x_1, x_2, ..., x_m)) = F(x_1, x_2, ..., x_m)$$

is called a superposition of functions of the system (1). In simple terms *superposition* is the construction of new function by means of replacement of some variables of the initial function by the appropriate functions.

The function *f* is called *basic function* and the function f_k is called *substitution function*.

Suppose we want to perform superposition of Boolean functions. Such problem arises, for example, while analyzing cryptosystems (see [1, 2, 5]) and Boolean multipoles [3, 4].

To consider the practical problems of construction of superposition of Boolean functions, we assume that every Boolean function of the system (1) is given with truth vector.

Thus, the purpose of this paper is to develop an algorithm of construction of truth vector of function, which is the result of superposition of functions, which are also given with truth vectors.

Let us consider reasonability of representation of a Boolean function as a truth vector.

There are different forms of representation of Boolean functions [3]. Thus, the tabular representation in form of *S*-blocks is used to define the system of Boolean functions in cryptosystems. This representation is absolutely inconvenient for analysis when it is necessary to retrace the changes of every bit in process of ciphering.

The other most popular form of representation of Boolean functions is representation of function in disjunctive normal form (DNF).

Let only one bit be used for representation of every literal (complemented or uncomplemented variable) of DNF. Then for representation of every elementary conjunction of DNF *n* bits are required. Assumed that the function *f* takes the unit value in half of sets, which occurs in cryptosystems, we come to conclusion that it is required to expend $n \cdot 2^{n-1}$ bits to represent the function. At the same time it is required 2^n bits to represent Boolean function with truth vector. It is obvious that if $n \geq 3$ then $n \cdot 2^{n-1} > 2^n = 2 \cdot 2^{n-1}$, thus the representation of function with truth vector is more efficient.

In general, other forms of representation of Boolean function are not so efficient as a truth vector.

2. Essential and fictitious variables

Let us consider the concepts of essential and fictitious variables of the function f.

Let there be the Boolean function $f(x_1, x_2, ..., x_n)$. Let us designate the set of values of variables $x_1, x_2, ..., x_n$ of the function f as $\tilde{\sigma} = (\sigma_1, \sigma_2, ..., \sigma_n)$.

It is said that the function f heavily depends upon the variable x_k (k = 1,2,..., n) if two sets
$\tilde{\sigma}_1 = (\sigma_1, ..., \sigma_{k-1}, 0, \sigma_{k+1}, ..., \sigma_n)$
$\tilde{\sigma}_2 = (\sigma_1, ..., \sigma_{k-1}, 1, \sigma_{k+1}, ..., \sigma_n)$
can exist and $f(\tilde{\sigma}_1) \neq f(\tilde{\sigma}_2)$. Such variable is called essential variable. Otherwise the variable x_k is called fictitious variable.

It is easy to see that if the variable x_k in function f is fictitious, the truth table of this function and, accordingly, its truth vector can be halved by deleting, for example, all sets, in which the variable $x_k = 1$. On the contrary, if it is necessary to enter the fictitious variable x_{n+1} in function $f(x_1, x_2, ..., x_n)$, then its truth table will double.

Let there be the Boolean function $\varphi(x_{i_1}, x_{i_2}, ..., x_{i_n})$ which is given with truth vector. Let us consider the procedure of entering of the fictitious variable x_k ($k \notin \{i_1, i_2, ..., i_n\}$) into the function φ. As a result, we obtain truth vector of the Boolean function $f(x_{i_1}, x_{i_2}, ..., x_{i_{n+1}})$ which is equivalent to the function φ.

We will consider the truth vector of the function φ as array $\vec{\Phi}$ with length 2^n and the truth vector of the function f as array $\vec{F}$ with length 2^{n+1}.

Theorem 1. *Let the Boolean function φ, which depends upon n variables, is given with the array $\vec{\Phi}$. If the fictitious variable x_k, which is entered into φ, takes k-place in the set of variables of the function $f(x_{i_1}, x_{i_2}, ..., x_{i_{n+1}})$ (if counted from left to right), then it is necessary to copy 2^{n-k+1} cells of the array $\vec{\Phi}$ in consecutive order to construct the truth vector of the function f (the array $\vec{F}$) and doubly place the same value in the cells of the array $\vec{F}$ in consecutive order.*

Proving. In fact, the procedure of construction of the array $\vec{F}$ depends upon the number k of the input fictitious variable x_k.

Let this number take k-place (if counted from left to right) in set of values of variables of the function $f(x_{i_1}, x_{i_2}, ..., x_{i_{n+1}})$. Assuming that the sets of values $n + 1$ of variables of the function f are ordered lexicographically, we can see that the variable x_k in the truth table f possess the same value in succession in 2^{n-k+1} sets ($k = 1,2,..., n + 1$).

This proves Theorem 1. Q.E.D.

Thus, the procedure of entering of one fictitious variable in the Boolean function φ, which is given with truth vector, is completely defined with the Theorem 1.

Example 2. Let it be necessary to enter the fictitious variable x_2 into the function $\varphi(x_1, x_3) = (1001) = \vec{\Phi}$. Thus, $n = 2$, $k = 2$. It is easy to see that the truth table of the function $f(x_1, x_2, x_3)$ will be the following:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

In this case it is obviously that it is necessary to copy $2^{n-k+1} = 2^{2-2+1} = 2$ cells of the array $\vec{\Phi}$ successively and doubly place them successively into the array $\vec{F}$.

It is clear, that if it is necessary to enter several fictitious variables into the function φ, it can be done by means of entering fictitious variables one by one into the initial function φ, and then in every function, which would obtained at the previous stage.

Theorem 2. Let there be the function $\varphi(x_1, x_2, ..., x_n)$, which is defined with truth vector, and p fictitious variables should be entered into it. Then the time of construction of the function $f(x_1, x_2, ..., x_{n+p})$ is equal to 2^{n+p}.

Proving. Let us define the time required for entering of fictitious variable into the function φ. In accordance with Theorem 1 it is necessary to copy every 2^{n-k+1} bits of truth vector φ and doubly place them into the truth vector of the function $f(x_1, x_2, ..., x_{n+1})$. The total number of the bits, placed into the truth vector of the function f, obviously is equal to 2^{n+1}, and the total number of copied bits is equal to 2^n. Consequently, the time of entering of the first fictitious variable is equal to $O(2^n)+O(2^{n+1}) = O(2^{n+1})$.

It is easy to see that the time of entering of the second fictitious variable is equal to $O(2^{n+2})$. Consequently, the time required for entering of the first two fictitious variables will be equal to $O(2^{n+1}) + O(2^{n+2}) = O(2^{n+2})$. Similarly we come to conclusion that it is necessary to expend 2^{n+p} time units for entering of p fictitious variables into the function φ. Q.E.D.

3. Algorithm of construction of superposition

Thus, let there be a system of Boolean functions (1). Let us say, that

$$X = \{x_1, ..., x_n\}$$

is the set of variables of the basic function f, and

$$X_i = \{x_1, ..., x_m\}$$

is the set of variables of substitution function

$$f_i \ (i = 1,2,..., n).$$

Further, we suppose that

$$\bigcup_{i=1}^{n} X_i \subseteq X. \qquad (2)$$

It is clear that, in general, the relation (2) may be not satisfied. Therefore we suppose that the necessary fictitious variables are beforehand entered into the basic function.

We shall carry out the construction of truth vector of the resulting function f_r bit by bit. The number of the bit, which is constructed, will determine counter content $C_r(n)$, which has n positions. The counter content is written as sequence $C_r(n) = (\sigma_1, \sigma_2, ..., \sigma_n)$. In fact, the counter content determines the set of variables of the Boolean function f.

Firstly let us consider the substitution of the function $f_1(x_1,...,x_m)$ into the basic function f instead of the variable x_k (k = 1,2,..., n), $m \le n$. We suppose that we have three arrays: $\vec{F}$, $\vec{F}_1$ and $\vec{F}_r$, which represent the truth vectors of the basic function, the substitution function and the resulting function respectively, and three counters C(n), C(m) and $C_r(n)$.

Let us write C(n) = 0, when the counter content is zero, and C(n) := C(n)+1, when the counter content is increased by 1. Finally, we will designate the content of the array $\vec{F}_r$ with number (address) $C_r(n)$ as $\vec{F}_r(C_r(n))$.

Algorithm A1

Step 1. Fix $C_r(n) := 0$.

Step 2. Copy to counter C(m) those positions of the counter $C_r(n)$ which correspond to variables of the function f_1. Also, copy all positions of the counter $C_r(n)$ to C(n).

Step 3. Find the value of $\vec{F}_1(C(m))$ in the array $\vec{F}_1$.

Step 4. Substitute the value $\vec{F}_1(C(m))$ for the k^{th} position of the counter C(n).

Step 5. Find the value of $\vec{F}(C(n))$ in the array $\vec{F}$.

Step 6. Write $\vec{F}_r(C_r(n)) = \vec{F}(C(n))$, that is, write the value of the bit with number $C_r(n)$ in the array $\vec{F}_r$.

Step 7. Fix $C_r(n) := C_r(n)+1$.

Step 8. If $C_r(n) < 2^n$ then go to Step 2.

Step 9. The array $\vec{F}_r$ is completely formed.

Example 3. Let it be necessary to substitute the Boolean function $f_1(x_1,x_2,x_3)$ = (10110010) for the variable x_1 in the function $\varphi(x_1,x_3,x_4)$ = (10100011).

To satisfy the relation (2) we enter the fictitious variable x_2 into the basic function. Using the procedure, determined by Theorem 1, we obtain new expression for the basic function: $f(x_1,x_2,x_3,x_4)$ = (1010101000110011).

Now we can use the algorithm A1 to construct the truth vector of the resulting function f_r:

Step 1. Fix $C_r(n) = C_r(4) := 0$.

Step 2. Have: $C(m)$ = C(3) := 0, C(4) := 0.

Step 3. Find $\vec{F}_1(C(m)) = f_1(0) := 1$.

Step 4. Obtain C(n) = C(4) := 8 (high-order position of the counter $C_r(n)$ changes by 1).

Step 5. Find $\vec{F}(C(n)) = \vec{F}(C(4)) := 0$.

Step 6. Write $\vec{F}_r(C_r(n)) = \vec{F}_r(0) := 0$.

Step 7. Fix $C_r(4) := 0 + 1 = 1$.

Step 8. As $C_r(4) = 1 < 2^4$ then go to Step 2.

Step 2. Have: C(3) := 0, C(4) := 1.

Step 3. Find $\vec{F}_1(0) := 0$.

Step 4. Obtain C(4) := 1.

Step 5. Find f(1) := 0.

Step 6. Write $f_r(1) := 0$.

Step 7. Fix $C_r(4) := 1 + 1 = 2$.

Step 8. As $C_r(4) = 1 < 2^4$ then go to Step 2.

Step 2. Have: C(3) := 1, C(4) := 2.

Step 3. Find $\vec{F}_1(1) := 0$.

Step 4. Obtain C(4) := 2.

Step 5. Find $\vec{F}(2) := 1$.

Step 6. Write $\vec{F}_r(2) := 1$

and so on. Consequently we obtain the truth vector of the resulting function $f_r(x_1,x_2,x_3,x_4)$ = (0010001100100010).

Theorem 3. The algorithm A1 finds the truth vector of the resulting function.

Proving. Let there be the basic function $f(x_1,x_2,...,x_n)$ and the substitution function $f_1(x_1,...,x_m)$ ($m \le n$). We can suppose without loss of generality that the first m variables of the functions f and f_1 coincide and the function f_1 is substituted for the variable x_1.

Suppose that it is necessary to find the value of the resulting Boolean function $f_r(f_1,x_2,...,x_n)$ in the set $\tilde{\sigma} = (\sigma_1,\sigma_2,...,\sigma_n)$. Let $\tilde{\sigma}_1 = (\sigma_1,\sigma_2,...,\sigma_m)$ be the subset of the first m values of variables.

According to the algorithm A1 the value of the substitution function $f_1(\tilde{\sigma}_1)$ should be found at Step 3. It is obvious that the value of the resulting function in the set $\tilde{\sigma}$ will depend upon which value the function f_1 takes in the set $\tilde{\sigma}_1 \subseteq \tilde{\sigma}$. As this function replaces the variable x_1, the resulting function f_r takes the same value in the set $\tilde{\sigma}$ as the function f in the set
$(f_1(\tilde{\sigma}_1),\sigma_2,...,x_n)$

which should be found at Step 5 of the algorithm A1. Q.E.D.

Theorem 4. The running time of algorithm A1 is equal to $O((m+n)\cdot 2^n)$.

Proving. In fact, carrying out of Step 1 takes *n* units of time. Carrying out of Step 2 takes, obviously, *m* units of time. Each of Steps 3 – 6 takes one unit of time. In the general case, Step 7, as Step 1, takes *n* units of time. Step 8 also takes one unit of time.

Thus, on-time execution of Steps 2 – 8 takes *m*+*n*+5 units of time, therefore, it is necessary to spend $(m + n + 5) \cdot 2^n = O((m+n)\cdot 2^n)$ units of time for construction of the truth vector of the resulting function. Q.E.D.

References

[1] A.D. Plotnikov, Logical cryptoanalysis on the example of the cryptosystem DES. http://eprint.iacr.org/2010/053, 2010.
[2] Data Encryption Standard (DES). Federal Information Processing Standards Publication (FIPS PUB 46-3). National Bureau of Standards, Gaithersburg, MD (1999).
[3] A.D. Zakrevskij, Solving large systems of Boolean equations. "Informatics", 4, 2004. pp. 42–53.
[4] S. Rudeanu, Boolean functions and equations. Amsterdam-London-New York: North- Holland and American Elsevier, 1974.
[5] J.A. Clark, , J. L. Jacob, S. Maitra, and P. Stanica, Almost Boolean Functions: the Design of Boolean Functions by Spectral Inversion. "Computational Intelligence", Volume 20, Number 3, 2004. pp. 447–458.

Anatoly Plotnikov has got his PhD in 1969, He is a professor of Volodymyr Dalh East-Ukrainian National University, Luhansk. He is also an associate member of AMS (USA) and a reviewer of the journal "Mathematical Reviews". His scientific interests include algorithms, combinatorial optimization, complexity theory.

Alexander Petrov is a professor of Department Applied Informatics, AGH University of Science and Technology, Krakow, Poland.

Anton Petrov is a professor of Volodymyr Dalh East-Ukrainian National University, Luhansk. His scientific interests include algorithms and Boolean functions.