FINDING AN APPROPRIATE PARTITION ON THE SET OF ARGUMENTS OF A PARTIAL BOOLEAN FUNCTION TO BE DECOMPOSED

Arkadij Zakrevskij

Abstract: A hard combinatorial problem is investigated – the two-block decomposition, generally non-disjunctive, of partial Boolean functions. The key task is regarded: finding such a weak partition on the set of arguments, at which the considered function can be decomposed. Solving that task is essentially speeded up by the way of preliminary discovering traces of the sought-for partition. Boolean and ternary vectors and matrices are used, with efficient parallel combinatorial operations over them.

Keywords: Partial Boolean function, non-disjunctive decomposition, appropriate partition, combinatorial search, traces of the solution.

Introduction

A task is considered concerning the problem of functional decomposition of Boolean functions, which was set originally in papers [Povarov, 1954], [Ashenhurst, 1959], [Curtis,1962]. Let's formulate it as follows.

Suppose a Boolean function $f(x) = f(x_1, x_2, \ldots, x_n)$ is given. It is required to decompose it, having presented as the following composition of two functions g and h of smaller number of variables:

$$f(x) = g(h(u, w), w, v).$$

The sets of arguments given in the vector form are connected by that with the relations $x = u \cup w \cup v$, $u \cup w = u \cup v = w \cup v = \varnothing$, and the couple of sets u and v determines a weak partition on set x, designated as u/v. It is accepted to name such composition sequential two-block. It is illustrated by Figure 1 for the case, when $u = (x_1, x_2, x_3)$, $w = (x_4, x_5)$ and $v = (x_6, x_7)$.

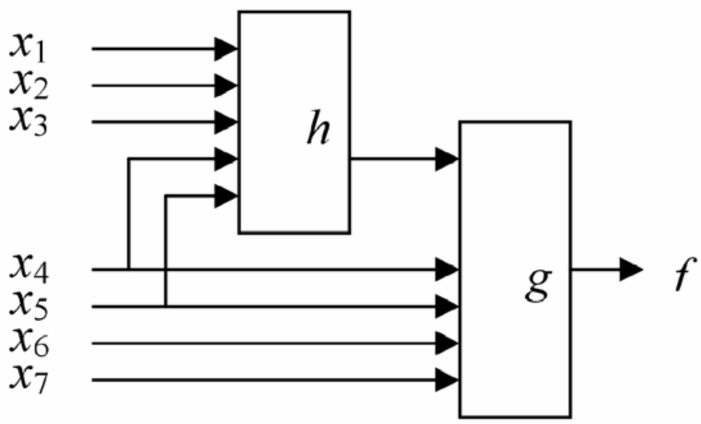

Fig. 1. Example of composition

To solve the formulated task it is necessary, first of all, to find such a weak partition u/v, at which the variables of set u enter in number of arguments of function h only, and variables of v – only in number of arguments of function g. The conditions $|u| > 1$ and $|v| > 0$ should be fulfilled also - otherwise the composition will appear trivial (exists always). Let's name this partition *appropriate*, and the function $f(x)$ – separable, or decomposable at the given partition.

The finding of appropriate partition is a difficult task, for which solution an effective combinatorial algorithm was offered [Zakrevskij, 2006], in application to completely specified Boolean functions. This task becomes even more complicated, when the function $f(x)$ appears to be partial, being defined not on all sets of values of variables from set x. Just this case is considered below.

It] was shown [Zakrevskij, 2006], that the probability of decomposability of a random completely defined Boolean function fast tends to zero with growth of number of variables n, so already at $n > 9$ such a function, most likely, is not decomposable. In case of partial functions this probability arises with growth of uncertainty, however even in this case it remains small enough.

Taking into account the given remark, let's assume, that it is known beforehand, that the considered function $f(x)$ is separable, being obtained as a result of composition $g(h(\mathbf{u}, \mathbf{w}), \mathbf{w}, \mathbf{v})$ of some two Boolean functions g and h on a weak partition $\mathbf{u}/\mathbf{v}$ on the set of arguments $\mathbf{x}$. It is required to detect (to recognize) this partition, after which the obtaining of functions g and h is not a difficult task.

A method of checking a partial Boolean function for decomposability at some given weak partition was offered in [Zakrevskij, 2007]. In the case, when this appropriate partition is not known a priori, it is possible to organize its search, sorting out different weak partitions and checking the function on decomposability at them. However, such way is rather labor-consuming, as the number of different weak partitions on the set of variables is approximated from above by value 3^n, fast growing with increase of number of variables n.

In the present paper the method of search for appropriate partition $\mathbf{u}/\mathbf{v}$ by its traces is suggested, which sufficiently cuts down the number of analyzed partitions. Originally it was designed for completely specified Boolean functions [Zakrevskij, 2006], but here it is extended on the case of partial Boolean functions.

Basic operations in Boolean space

The parallelism of component-wise operations above long Boolean vectors is used in the offered method of search for appropriate partitions, and that essentially accelerates the fulfilled calculations.

It is possible to consider Boolean functions $f(x) = f(x_1, x_2, \ldots, x_n)$ and $g(x) = g(x_1, x_2, \ldots, x_n)$ of n variables as appropriate subsets of units of Boolean space $\{0, 1\}^n$. Let's represent them by Boolean 2^n-vectors $\mathbf{f}$ and $\mathbf{g}$, displaying such vectors below in the more convenient for visual perception matrix form. In such representation the two-place Boolean operations $f \vee g$, $f \wedge g$, $f \oplus g$, $f \to g$ are easily implemented as parallel component-wise operations over appropriate Boolean vectors designated $\mathbf{f} \vee \mathbf{g}$, $\mathbf{f} \wedge \mathbf{g}$ (or, simpler, $\mathbf{f}\,\mathbf{g}$), $\mathbf{f} \oplus \mathbf{g}$, $\mathbf{f} \to \mathbf{g}$. Including these operations in the formed basis, we shall supplement them by some operations of interaction between different components within the framework of one Boolean vector.

Let's remind, that the function $f(x)$ can be represented as Shannon disjunctive decomposition by an arbitrary variable x_i, which coefficients $f_{i\,0}$ and $f_{i\,1}$ are Boolean functions obtained as a result of substitution of variable x_i by values 0 or 1 :

$$f(x) = \bar{x}_i f_{i\,0} \vee x_i f_{i\,1} ,$$

Using vector representation of the function, we shall designate these operations accordingly through $\mathbf{f} - i$ and $\mathbf{f} + i$. They are easily implemented in the Boolean space on couples of elements adjacent by the variable x_i. When executing the operation $\mathbf{f} - i$ both elements of the couple gain the value of the element defined by the condition $x_i = 0$, at execution of the operation $\mathbf{f} + i$ – the value of the other element corresponding to value 1 of variable x_i.

Let's show examples of such operations, and also of their compositions.

```
          f                   f-2                 f-2-5

            --------            --------            --------     4
        ----    ----        ----    ----        ----    ----    3
      --  --  --  --      --  --  --  --      --  --  --  --   2
     - - - - - - - -     - - - - - - - -     - - - - - - - -  1
    0110110101011110    0101111101011111    0101111101011111
  | 0010010000010110    0000010100000101    0101111101011111
 |  1100101001110001    1111101001010000    1111101001010000
 || 0100010111010011    0101010111110000    1111101001010000

    1101110111101110    1010101000010001    0000000000110011
  | 0100010001100110    0101010100110011    0000000000110011
 |  1010101000010001    1010101000010001    1100110011111111
 || 0101010100110011    0101010100110011    1100110011111111
 6 5       f+3                f+3+6             f+1-3+5
```

By interaction of adjacent units there are implemented also operation Inv_i *f* of inverting the function *f* at the variable x_i (adjacent elements interchange their values), and so-called operations of symmetrization $S_i{}^{*} f$, in which both elements get value defined by the two-place operation * above their initial values [Zakrevskij, 1963]. As a result of these operations the function

$$f(x) = \bar{x}_i f_{i0} \vee x_i f_{i1}$$

is transformed correspondingly into functions

$$\bar{x}_i f_{i1} \vee x_i f_{i0}\ ,\quad \bar{x}_i (f_{i0} * f_{i1}) \vee x_i (f_{i0} * f_{i1}).$$

Examples of these operations are shown below.

```
          f                 Inv4 f               S4∨ f

            --------            --------            --------     4
        ----    ----        ----    ----        ----    ----    3
      --  --  --  --      --  --  --  --      --  --  --  --   2
     - - - - - - - -     - - - - - - - -     - - - - - - - -  1
    0110110101011110    0101111001101101    0111111101111111
  | 0010010000010110    0001011000100100    0011011000110110
 |  1100101001110001    0111000111001010    1111101111111011
 || 0100010111010011    1101001101000101    1101011111010111

    0000110000001100    0011001100110011    0111111101111111
  | 0000000000000000    0011001000110010    0111111101111111
 |  1100000000110000    1011101110111011    1111111111111111
 || 0000000011000011    1001011010010110    1111111111111111
 6 5
```

$S_1^{\wedge}\,\mathbf{f}$ $\qquad\qquad$ $S_4^{\oplus}\,\mathbf{f}$ $\qquad\qquad$ $S_{4,5}^{\vee}\,\mathbf{f}$

Here $S_{4,5}^{\vee} f$ means composition $S_4^{\vee}(S_5^{\vee} f)$.

Search by traces. Triads and fragments

The method of decomposition suggested below is based on the following reasons which key moments are given in the form of assertions. They were formulated before for the case of completely specified Boolean functions [Zakrevskij, 2006], but became valid when partial Boolean functions are considered.

Suppose two partitions u/v and u^*/v^* are given, such that $u^* \subseteq u$ and $v^* \subseteq v$. Let's speak, that partition u^*/v^* submits to partition u/v, and call it a *trace* of u/v,

Assertion 1. If a partial Boolean function $f(x)$ is decomposable at partition u/v, it is decomposable as well at partition u^*/v^*.

Corollary. If the function $f(x)$ is not decomposable at partition u^*/v^*, it is not decomposable also at partition u/v.

Let's assume $|u| = k$ and $|v| = m$. Partition with $k = 2$ and $m = 1$ we shall term as a *triad*. It is the simplest of partitions, at which some nontrivial decomposition can take place.

Assertion 2. A partial Boolean function is decomposable, if and only if it is decomposable if only at one of triads.

Therefore in the offered method the search for partition u/v starts with the search of its traces on the set of triads, i.e. with looking for an appropriate triad. The needed checking of triads can be fulfilled fast enough, as their number is not large, being significantly less than the number of all weak partitions.

Assertion 3. The number of triads is equal to $C_n^2\,(n-2) = \dfrac{n(n-1)(n-2)}{2}$.

Suppose, that some appropriate triad $(x_p, x_q)/x_r$. is detected. If it submits to the required partition u/v, the latter can be found, having put for the beginning $u = (x_p, x_q)$ and $v = (x_r)$, and then sequentially expanding these two sets, sorting out remaining variables and testing them on possibility of inclusion into set u or v.

By reviewing some concrete triad u/v the Boolean space $M = \{0, 1\}^n$, where the partial Boolean function $f(x)$ is presented, is divided into $2^{|w|}$ intervals corresponding to different values of vector w. On each of them the corresponding coefficient f_i of disjunctive decomposition of the function on variables of set w is given. It represents some partial Boolean function of variables x_p, x_q, x_r. As a matter of convenience of subsequent reasoning we shall present each of these coefficients by a ternary matrix size 4×2, which rows correspond to values of the two-component vector u, and columns – to values of the one-component vector v. Let's designate this matrix T_i and name it a *fragment*. Thus, the 2^n-element ternary matrix representing the function $f(x)$, is decomposed into 2^{n-3} eight-element fragments specifying functions $f_i(x_p, x_q, x_r)$.

A concrete example of such splitting into eight fragments for a partial Boolean function $f(a, b, c, d, e, f)$ and triad $(a, b)/c$ is shown below.

```
                ----------    f
           -----      -----  e
        --    --    --    -- d
       -  -  -  -  -  -  -  - c
          10 0- 1- 11 0- 10 -1 10
      |   -1 00 -1 1- 00 -1 1- -0
     |    0- 1- -0 10 10 0- 11 -1
     ||   1- 01 11 -1 00 10 -0 11
     b a
```

Assertion 4. The function $f(x)$ can be decomposed at triad $(x_p, x_q)/x_r$, if and only if each of the coefficients $f_i(x_p, x_q, x_r)$ also is decomposable at the same triad.

It follows from here, that the probability of decomposability of function $f(x)$ at a concrete triad is equal to γ^k, where k is the number of coefficients equal 2^{n-3} and γ – the probability of decomposability of one coefficient. In the case of a completely specified Boolean function the last probability is approximated by the value 1/3, and with growth of uncertainty decreases. Nevertheless, the probability of decomposability of the function $f(x)$ quickly decreases with growth of the number of variables in it.

Checking triads for fitness

So, a triad is appropriate, if each of fragments of the corresponding splitting of the ternary matrix is suitable. That means, the partial function $f(x)$ can be completely defined in such a way, that each fragment will contain no more than two types of Boolean rows (each having equal rows). In other words, a fragment is suitable, if it contains no more than two classes of compatible rows. Remind, that two ternary rows are compatible, i.e. they could become equal by changing "–"-values of some components for 1 or 0, if they are not orthogonal. It follows from here, that the fragment is suitable, if the graph of orthogonality of its rows is bichromatic [Harary, 1969].

Let's offer the following way of checking fragments with the purpose of detection of suitable ones among them. Any fragment contains four rows, therefore the graph of orthogonality has four vertices. It is bichromatic, if it has no cycle of length three. Let's select arbitrary two different vertices. If such a cycle exists, then one of the selected vertices will belong to it. Therefore, it is enough to test each of these two vertices on belonging to a cycle of length three. If such belonging will not be revealed, graph is bichromatic, and the triad is suitable.

Necessary and sufficient condition of entering a vertex, i.e. corresponding row, in a cycle of length three could be formulated as follows: among rows orthogonal to the given one, there exist mutually orthogonal rows.

For example, the first of the shown below fragments appears to be suitable, and the second – no, as there is a cycle of length three, composed by three last rows: each of them is orthogonal to the other two.

10	**-1**	1
-1	**00**	2
0-	**1-**	3
01	**01**	4

The offered way is implemented by the following algorithm, which is remarkable by that it checks on fitness simultaneously all 2^{n-3} fragments generated by the given triad, and finds out by that if the triad is appropriate. The partial function $f(x)$ is represented by a couple of Boolean vectors $\boldsymbol{f}^0$ and $\boldsymbol{f}^1$, in first of which by 1s are marked the values 0 of the function and in the second - values 1. The splitting of space into fragments is fulfilled by the triad $(x_p, x_q) / x_r$.

To begin with, first rows of fragments are checked, which form the initial coefficient f^- of decomposition of the function $f(x)$ on variables x_p and x_q. The rows orthogonal to the checked up row, are marked by value 1 in the computed vector $\boldsymbol{g}$, and their values are fixed by the couple of vectors $\boldsymbol{h}^0$ and $\boldsymbol{h}^1$, checked up further for orthogonality. Alike the initial vectors $\boldsymbol{f}^0$ и $\boldsymbol{f}^1$, they are Boolean vectors with 2^n components.

$\boldsymbol{h}^0 := (\boldsymbol{f}^0 – p) – q$	Getting the initial coefficient f^-
$\boldsymbol{h}^1 := (\boldsymbol{f}^1 – p) – q$	
$\boldsymbol{g} := S_r^{\vee}(\boldsymbol{h}^0\boldsymbol{f}^1 \vee \boldsymbol{h}^1\boldsymbol{f}^0)$	Finding coefficients orthogonal to f^-
$\boldsymbol{h}^0 := S_u^{\vee}(\boldsymbol{f}^0\boldsymbol{g})$	Computing their intersection

$h^1 := S_u^{\vee}(f^1 g)$

If it turns out that $h^0 h^1 \neq \mathbf{0}$, the triad is accepted as not appropriate. In case if $h^0 h^1 = \mathbf{0}$ the final rows of fragments are checked, which constitute the final coefficient f^+.

$h^0 := (f^0 + p) + q$	Getting the final coefficient f^+
$h^1 := (f^1 + p) + q$	
$g := S_r^{\vee}(h^0 f^1 \vee h^1 f^0)$	Finding coefficients orthogonal to f^+
$h^0 := S_u^{\vee}(f^0 g)$	Computing their intersection
$h^1 := S_u^{\vee}(f^1 g)$	

If $h^0 h^1 \neq \mathbf{0}$, then the triad is not appropriate. On the other hand, if $h^0 h^1 = \mathbf{0}$, the triad is accepted as appropriate.

Example. Let's return to regarding the partial Boolean function $f(a, b, c, d, e, f)$, representing it by a couple of Boolean vectors (rolled up into matrices) f^0 and f^1:

```
                      -----------    f
                -----       -----   e
             --    --    --    --  d
            -  -  -  -  -  -  -  - c
           10 0- 0- 11 0- 10 -1 10
       |   -1 10 -1 1- 00 -1 0- -0
      |    0- -1 -0 00 10 0- 11 -1
      ||   1- -0 11 -1 -0 10 -0 11
     b a
           01 10 10 00 10 01 00 01   f0
           00 01 00 00 10 00 10 01
           10 00 01 11 00 10 00 00
           00 01 00 00 00 01 01 00

           10 00 00 11 00 10 01 00   f1
           01 10 01 10 00 01 00 00
           00 01 00 00 10 00 11 01
           10 00 11 01 00 10 00 01
```

The check of the function for decomposability at triad (a, b) / c is reduced to testing in parallel all of fragments for fitness. In the given example all of eight fragments are suitable, therefore the function can be decomposed at that triad

We shall illustrate that operation by the case of testing one of the fragments, third at the left, demonstrating initial values of appropriate components of the ternary vector ***f*** and of vectors obtained sequentially by the algorithm:

$\mathbf{f}^0$, $\mathbf{f}^1$, $\mathbf{h}^0$, $\mathbf{h}^1$, $\mathbf{h}^0\mathbf{f}^1$, $\mathbf{h}^1\mathbf{f}^0$, $\mathbf{g}$, $\mathbf{f}^0\mathbf{g}$, $\mathbf{f}^1\mathbf{g}$, $\mathbf{h}^0$ and $\mathbf{h}^1$.

The check is carried out both on initial coefficient f^-, and on finite coefficient f^+.

f	f^0	f^1	h^0	h^1	h^0f^1	h^1f^0	g	f^0g	f^1g	h^0	h^1	
0-	10	00	10	00	00	00	00	00	00	00	11	Initial
-1	00	01	10	00	00	00	00	00	00	00	11	coefficient
-0	01	00	10	00	00	00	00	00	00	00	11	f^-
11	00	11	10	00	10	00	11	00	11	00	11	
												$h^0h^1 = 0$
			00	11	00	10	11	10	00	11	00	Final
			00	11	00	00	00	00	00	11	00	coefficient
			00	11	00	01	11	01	00	11	00	f^+
			00	11	00	00	00	00	00	11	00	
												$h^0h^1 = 0$

The triad is appropriate

Search for appropriate partition

If the considered triad $(p, q / r$ had appeared suitable, it is possible to assume, that it is a trace of the sought-for appropriate partition. In this case the latter can be found by moving along the track generating from the found trace. By that the value of vector g obtained at the previous stage is used, and sets u and v are sequentially expanding, beginning with initial values $u = (p, q)$ and $v = (r)$.

Expanding set v. Let's begin from set v. Sorting out sequentially all elements s from set $x \setminus (u \cup v)$, we shall discover among them such ones, at which inclusion in set v the partition u/v remains appropriate. With this purpose three operations are fulfilled for each element s:

$$e := S_s^{\vee} g$$

$$h^0 := S_u^{\vee} (f^0 e)$$

$$h^1 := S_u^{\vee} (f^1 e)$$

and if $h^0 h^1 = \mathbf{0}$, then s is included into v by implementing operations

$$v := v \cup \{s\},\ g := e.$$

So the final value of set v is found.

Expanding set u. The maximum expansion of set u is found similarly. If it is known, that the required partition is strict, it is possible to put $u = x / v$ and, probably, to test the function for decomposability, as the algorithm used is heuristic. Let's remark, however, that the probability of obtaining by this algorithm erroneous solution fast tends to zero with growth of the number of variables n.

If the required partition could be not strict, it is necessary to test all elements from initial value of set $x \setminus (u \cup v)$ for the possibility of including them into set u.

Check of the immediate element s can be fulfilled by the following heuristic algorithm, which partly implements the procedure circumscribed in [Zakrevskij, 2007]. The algorithm considers the initial coefficient f^- of the function f decomposition by the current value of set u, finds orthogonal to it coefficients, checks them for compatibility and, in case of compatibility, includes element s in set u without further check.

$e := u \cup \{s\}$

$h^0 := f^0 - e$

$h^1 := f^1 - e$

$g := S_v^{\vee} (h^0 f^1 \vee h^1 f^0)$

$h^0 := S_u^{\vee} (f^0 g)$

$h^1 := S_u^{\vee} (f^1 g)$

If $h^0 h^1 = \mathbf{0}$, then s is included into u by operation $u := e$.

In such a way the set u is found and, therefore, the whole partition u/v .

Note, that the operation of looking for coefficient f^- is presented in this algorithm in abbreviated form, by expressions $h^0 := f^0 - e$ and $h^1 := f^1 - e$, instead of more detailed

$h^0 := (\ldots((f^0 - e_1) - e_2) - \ldots) - e_t$,

$h^1 := (\ldots((f^1 - e_1) - e_2) - \ldots) - e_t$,

where $e = (e_1, e_2, \ldots, e_t)$.

Conclusion

In this paper, the heuristic algorithm is offered for finding such weak two-block partition on the set of variables of a partial Boolean function, on which the function can be decomposed. The algorithm is effective, if there exists a good solution "«hidden" in vector representation of the function of many variables. In this case the search of the partition is reduced to recognition of the latter.

Acknowledgement

This work was supported by Belarusian Republican Fond of Fundamental Researches (Project **Ф07МС-034**).

Bibliography

[Ashenhurst, 1959] Ashenhurst R.L. The decomposition of switching functions. – Proc. International Symposium on the Theory of Switching, Part 1. – Harvard University Press, Cambridge, 1959, pp. 75-116.

[Curtis,1962] Curtis H.A. Design of switching circuits. – Van Nostrand, Princeton, N. J., 1962.

[Povarov, 1954] Povarov G.N. About functional decomposition of Boolean functions. – Reports of the AS of USSR, 1954. – V. 4, No 5 (in Russian).

[Harary, 1969] Frank Harary. Graph theory. – Addison-Wesley Publishing Company : Reading, Massachusetts; Menlo Park, California; London; Don Mills, Ontario. 1969.

[Zakrevskij, 1963] Zakrevskij A.D. Universal system for solving problems the type relay system synthesis. – Annals of Siberian Physical-Technical Institute, 1963. – Issue 42, pp. 9-37 (in Russian).

[Zakrevskij, 2006] Arkadij Zakrevskij. Decomposition of Boolean functions by looking for tracks of a good solution. – Proceedings of XII-th International Conference Knowledge-Dialogue-Solution (KDS–2006), June 20-30, 2006, Varna, Bulgaria. pp. 211-216.

[Zakrevskij, 2007] Zakrevskij A.D. Decomposition of partial Boolean functions – checking for decomposability at a given partition. – Informatics, 2007, 1(13), (in Russian).

Author's Information

Arkadij Zakrevskij - United Institute of Informatics Problems of the NAS of Belarus, Surganov Str. 6, 220012 Minsk, Belarus; e-mail: zakr@newman.bas-net.by