МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2003
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2003
Proceedings of the Thirty Second Spring Conference of
the Union of Bulgarian Mathematicians
Sunny Beach, April 5–8, 2003

GENERATING AND IDENTIFICATION OF MONOTONE BOOLEAN FUNCTIONS

Valentin Bakoev

Three algorithms, based on a matrix structure, are described here. First of them generates all monotone Boolean functions of n variables in lexicographic order. The second one determines the first (resp. the last) lexicographically minimal true (resp. maximal false) vector of an unknown function. It serves the third algorithm, which identifies an unknown monotone Boolean function f of n variables by using membership queries only. For up to 6 variables it determines f with at most $m.n$ queries, where m is the combined size of the sets of minimal true and maximal false vectors of f.

1. Introduction. The problems in the area of monotone Boolean functions (MBFs) are important not only for the Boolean algebra. Many of them concern problems, arising in various fields, such as graph (hypergraph) theory, threshold logic, circuit theory, computation learning theory, artificial intelligence etc. [2, 3, 6]. Many of them still have open complexities and some scientists recommend new structures and tools to be searched and used [2, 6]. Three of the well known problems, concerning MBFs are: (1) Dedekind's problem, (2) Identification problem, (3) Searching the maximal upper zeros (resp. minimal lower ones) [7, 11].

In [1, 10] we introduced one matrix structure and summarized three algorithms, related to the solving of problems (1), (2) and (3). Here we present these algorithms and our recent results about them.

2. Basic notions, definitions and preliminary results. Let $\alpha = (a_1, \dots, a_n)$, $\beta = (b_1, \dots, b_n)$ be binary vectors from the $n-$dimensional Boolean cube $\{0,1\}^n$. An *ordinal number* of the vector α is the number $\#(\alpha) = a_1.2^{n-1} + a_2.2^{n-2} + \cdots + a_n.2^0$. The vector α *precedes lexicographically* vector β, if there exists an integer $k, 1 \le k \le n-1$ such that $a_i = b_i$ for $i = 1, 2, \dots, k$ and $a_{k+1} < b_{k+1}$. When the vectors from $\{0,1\}^n$ are in lexicographic order (as we consider further), their ordinal numbers form the sequence $0, 1, \dots, 2^n - 1$.

The relation "$\preceq$" is defined over $\{0,1\}^n \times \{0,1\}^n$ as follows: $\alpha \preceq \beta$ iff $a_i \le b_i$ for $i = 1, 2, \dots, n$. It is *reflexive, antisymmetric* and *transitive* and so $\{0,1\}^n$, towards the relation "$\preceq$", is a *partially ordered set* (POSet). When $\alpha \preceq \beta$ or $\beta \preceq \alpha$ we call α and β *comparable*, otherwise we call them *incomparable.*

A *Boolean function* (or simply a *function*) f of n variables is a mapping $f : \{0,1\}^n \to \{0,1\}$. If $\alpha, \beta \in \{0,1\}^n$ and $\alpha \preceq \beta$ always implies $f(\alpha) \le f(\beta)$, then the function f is called *monotone* (or *positive*). If $f(\alpha) = 0$ (resp. $= 1$) then α is called a *false* (resp. *true*) vector of f. The set of all false vectors (resp. all true vectors) of f is denoted by $F(f)$ (resp. $T(f)$). When the set $T(f) = \emptyset$ (resp. $F(f) = \emptyset$) the corresponding function is a *constant* $\tilde{0}$ (resp. *constant* $\tilde{1}$). The false vector α is called *maximal* if there is no other vector $\alpha' \in F(f)$, such that $\alpha \preceq \alpha'$ and $\alpha \neq \alpha'$. The set of all maximal false vectors is denoted by *max* $F(f)$. Symmetrically, the true vector β is called *minimal* if there is no other vector $\beta' \in T(f)$, such that $\beta' \preceq \beta$ and $\beta' \neq \beta$, also *min* $T(f)$ denotes the set of all minimal true vectors of f. If f is a monotone function, it has an unique *minimal disjunctive normal form* (MDNF), consisting of all prime implicants of f, where all literals are uncomplemented. There is a bijection between the set of prime implicants in MDNF of f and the set *min* $T(f)$, i. e. each prime implicant of the type $x_{i_1}x_{i_2}\ldots x_{i_k}$ corresponds to the vector having ones in positions $i_1, i_2, \ldots, i_k$ and zeros in all the rest positions. Let P_n denotes the set of all MBFs of n variables and C_n - the set of all possible prime implicants for P_n.

We define the matrix $M_n = ||m_{i,j}||$ with dimension $2^n \times 2^n$ as follows: for each pair of vectors $\alpha, \beta \in \{0,1\}^n$, such that $\#(\alpha) = i$ and $\#(\beta) = j$, we put $m_{i,j} = 1$ if $\alpha \preceq \beta$ and $m_{i,j} = 0$ otherwise. Since "$\preceq$" is reflexive, antisymmetric and also the vectors from $\{0,1\}^n$ are in lexicographic order, M_n is a triangular matrix, containing ones on its major diagonal and zeros under it. For $n = 1$ and $n = 2$ the corresponding matrices are:

$$M_1 = \begin{pmatrix} 1\,1 \\ 0\,1 \end{pmatrix} \qquad M_2 = \begin{pmatrix} 1\,1 & 1\,1 \\ 0\,1 & 0\,1 \\ 0\,0 & 1\,1 \\ 0\,0 & 0\,1 \end{pmatrix}$$

Theorem 1. *M_n is a block matrix of the type*

$$M_n = \begin{pmatrix} M_{n-1}M_{n-1} \\ O_{n-1}\ M_{n-1} \end{pmatrix},$$

where M_{n-1} denotes the same matrix with dimension $2^{n-1} \times 2^{n-1}$, and O_{n-1} denotes the zero matrix with dimension $2^{n-1} \times 2^{n-1}$.

Remark 1. The triangle of numbers, placed on and over the major diagonal of the matrix M_n, is related to other well known structures:

a) it is a discrete variant of the fractal structure of the Serpinsky's triangle (we due this note to Prof. Krassimir Manev, Sofia University);

b) it coincides with the Pascal's triangle with 2^n rows, where all its numbers are taken modulo 2.

By $R_n = \{r_0, \ldots, r_{2^n-1}\}$ we denote the set of all rows of M_n (as binary vectors).

Theorem 2. *Let $\alpha = (a_1, a_2, \ldots, \not{a}_n) \in \{0,1\}^n$, $\#(\alpha) = i$, $1 \leq i \leq 2^n - 1$ and α has ones in positions $(i_1, i_2, \ldots, i_r)$, $1 \leq r \leq n$. Then the $i-th$ row r_i of the matrix M_n is the vector with functional values of the prime implicant $c_i = x_{i_1}x_{i_2}\ldots x_{i_r}$, being a monotone function. When $\#(\alpha) = 0$, the zero row of M_n corresponds to the constant $\tilde{1}$.*

For $n = 3$, the assertion of Theorem 2 is illustrated by the following table:

$\alpha = (x_1, x_2, x_3)$	$i = \#(\alpha)$	M_3	c_i
(0 0 0)	0	1 1 1 1 1 1 1 1	$\tilde{1}$
(0 0 1)	1	0 1 0 1 0 1 0 1	x_3
(0 1 0)	2	0 0 1 1 0 0 1 1	x_2
(0 1 1)	3	0 0 0 1 0 0 0 1	x_2x_3
(1 0 0)	4	0 0 0 0 1 1 1 1	x_1
(1 0 1)	5	0 0 0 0 0 1 0 1	x_1x_3
(1 1 0)	6	0 0 0 0 0 0 1 1	x_1x_2
(1 1 1)	7	0 0 0 0 0 0 0 1	$x_1x_2x_3$

Remark 2. Theorem 2 states the bijection $\varphi : R_n \to C_n$, the bijection between the vector representation and the formula representation of the prime implicants. The disjunction (resp. conjunction) over the set R_n we understand as a bit-serial disjunction (resp. conjunction). For $r_i, r_j \in R_n$, $\varphi(r_i \vee r_j) = c_i \vee c_j$ and $\varphi(r_i.r_j) = c_i.c_j$, and so φ is an isomorphism. Any monotone function f can be expressed in terms of a linear combination $f(x_1, x_2, \ldots, x_n) = a_0r_0 \vee a_1r_1 \vee \cdots \vee a_{2^n-1}r_{2^n-1}$, where the coefficients $a_0, a_1, \ldots, a_{2^n-1} \in \{0, 1\}$ (the trivial combination corresponds to $\tilde{0}$). When $f(x_1, x_2, \ldots, x_n) = r_{i_1} \vee r_{i_2} \vee \cdots \vee r_{i_k}$ is a MDNF of f, then r_{i_j} and $r_{i_l}, 1 \leq j < l \leq k$, are pairwise incomparable.

3. Algorithms, based on the matrix structure. The Dedekind's problem is a problem for enumerating all MBFs of n variables (or all antichains of subsets of a given POSet) [7]. A general formula for $|P_n|$ is still not derived. Till 1999 this number was known only for $0 \leq n \leq 6$ variables. Few years we tried to find the cardinality of $|P_7|$, applying the principle "generating and counting" [10]. It was not enough powerful and in 1999 Vl. Jovovic etc. succeeded to derive a formulae for $|P_7|$ and $|P_8|$, using analytic techniques mostly [8].

An algorithm for generating all MBFs for a given n is outlined in [7], its C++ realization (destined to test sorting the networks) is given in [8]. It is based on the statement, that if $g, h \in P_{n-1}$ and $g \preceq h$, then their concatenation $f = gh$ is also a monotone function and $f \in P_n$. So the program generates and stores all MBFs of $n-1$ variables to obtain all these of n variables, for $1 \leq n \leq 7$. Some aspects of the generating problem are considered and investigated in [3].

Our first algorithm, called *"GEN"*, generates all MBFs, whose vectors are in lexicographic order, for a given n $(1 \leq n \leq 7)$. It is based on the matrix M_n as follows: if the $i-$th row r_i of M_n has zero in position j, $i < j < 2^n - 1$, then the rows r_i and r_j are incomparable and $f = r_i \vee r_j = c_i \vee c_j$ is a MBF. If f has zero in position k, $i < k < 2^n - 1$, then f and r_k are incomparable and $f = r_i \vee v_j \vee v_k = c_i \vee c_j \vee c_k$ is a MBF. And so on.

Algorithm *Gen*
Input: the number of the variables n
Output: the vectors of P_n in lexicographic order
Procedure:

```
1) Set $f = \tilde{0}$. Output $f$.
2) For each row $r_i$, $i = 2^n - 1, \ldots, 0$, set $f = r_i$ and:
  a) output $f$;
  b) for each position $j$, $j = 2^n - 2, \ldots, i$ check whether $f[j] = 0$. If "Yes" then set
(recursively) $f = f \vee r_j$. Go to a).
```

The Pascal-code of Gen, given below, is very simple.

```
Procedure Gen (G: BoolFun; i : byte);
  var  j : byte;
  begin
    for j:= i to dim do     { disjunction between the i-th row }
      if M[i,j]=1 then  G[j]:= 1;   { and the current function }
    Output (G);
    for j:= dim-1 downto i+1 do   { searching zero for the next }
      if G[j]=0 then  Gen (G, j); { disjunction (function)      }
  end;   { Gen }

Begin   {Main}
  ...
  readln (n);            { number of variables }
  dim:= 1 shl n - 1;     { dim:= 2^n-1 }
  Fill_Array;            { filling in the matrix M }
  FillChar (F, dim, 0);  { the constant zero - separately }
  Output (F);
  for k:= dim downto 0 do Gen (F, k);
End.
```

Comments.

a) The Procedure Fill_Array generates and stores the matrix $M-$ by using either Remark 1 b), or Theorem 1). For $n > 7$ the matrix becomes large – so bit-representation of the matrix or other program environment must be used. A similar algorithm, which does not store the matrix in the memory, was invented about five years ago by Prof. Stoyan Kapralov (Technical University in Gabrovo).

b) GEN works correctly because the cycles have decreasing step. So, the rightmost zero of the serial function is selected for a disjunction on each step of the recursion.

c) GEN can generate all functions, which are lexicographically after given function f – it must be assigned to F (instead of $\tilde{0}$) initially. Also the cycle in the main program must begin from i – the position of the leftmost one in f.

d) GEN works in *incremental polynomial time* [5], i. e. the serial MBF is generated in time polynomial towards the combined size of: the input, the last configuration and a certain row from R_n.

e) GEN can be modified easily to generate all antichains of a given POSet in a specific order.

The next algorithm, called *"SEARCH"*, determines a minimal (resp. maximal) true (resp. false) vector (problem (3)) of an unknown MBF $f \in P_n$, by using *membership queries* only. In the terms of computational learning theory, this serves to the exact learning by queries to an oracle: it answers "Yes" or "No" to the asking $f(\alpha) = 0$ (or $f(\alpha) = 1$), for some unknown vectors $\alpha \in \{0,1\}^n$ [2, 6]. A similar and most popular algorithm is the algorithm of Gainanov [2, 6], which finds a new vector for $minT(f)$ or $maxF(f)$ and enlarges one of them. It uses at most $n+1$ membership queries and needs $O(n)$ total time to perform certain operations on the vectors.

SEARCH is based on Theorem 1 and it is simply a *binary search.* When it searches the first (resp. the last) lexicographically vector of $minT(f)$ (resp. $maxF(f)$), it tests only the corresponding odd (resp. even) positions of the unknown function $f \in P_n$. Each time it starts from position $2^{n-1} - 1$ (resp. 2^{n-1}) and chooses the half of the function (subfunction) for the next step. So SEARCH uses exactly n queries without any operations on the vectors.

The problem (2) (precisely its restricted version) is studied extensively by Makino, Ibaraki etc. [6, 2]. They propose some new algorithms, which decide whether an unknown function $f \in P_n$ is $2-$monotonic or not, and if f is $2-$monotonic output both $minT(f)$ and $maxF(f)$. These algorithms work in polynomial total time towards the combined size of the input n and of the output $m = |minT(f)| + |maxF(f)|$, and use polynomial (towards n and m) number of queries. Applying these results Shmulevich etc. proved, that almost all MBFs are polynomially learnable using membership queries [9].

The third algorithm is called *"IDENTIFY"* and it is an example of exact learning. It identifies an unknown $f \in P_n$, without any restrictions, by using membership queries only and determines its sets $minT(f)$ and $maxF(f)$. IDENTIFY works recursively. On each step it determines the leftmost one and the rightmost zero in f (by using SEARCH), and it splits f into two subfunctions $g, h \in P_{n-1}$, whose concatenation is f. After that it identifies recursively each of the subfunctions g and h. Splitting of the subfuction continues until the position of the rightmost zero in the serial subfunction becomes smaller than the position of the leftmost one in it. Then a separation of its prime implicants starts. IDENTIFY submits to the dynamic-programming strategy. Its main idea is realized by the procedure "ID", given below (some details are omitted).

```
Procedure Id (l, r, l1, r0  : word);
  { l is the left, r is the right boundary of the subfunction; }
  { l1 and r0 are the positions of the leftmost 1 and          }
  { of the rightmost 0 in it, respectively                     }
  var m  : word;   { position of the median }
      p0,          { positions of the leftmost 1 and     }
      p1 : word;   { the rightmost 0 in the subfunctions }
  begin
    if l1 > r0 then Registrate_Prime_Implicants (l1, r)
    else
      begin
        m:= (l+r) div 2;
        p0:= Search_Zero (l, m);
        Id (l, m, l1, p0);
        p1:= Search_One (m+1, r);
        Id (m+1, r, p1, r0);
      end;
  end;   {Id}
```

IDENTIFY is still in development and investigation. The recent experimental results show, that it identifies an unknown function $f \in P_n$ $(1 \leq n \leq 6)$ by using at most $n.m$ membership queries. Other results are given in the following table, where q_n denotes the corresponding number of queries for n variables.

n	Maximal q_n	Average q_n		n	Maximal q_n	Average q_n
1	2	1.66		4	12	8.95
2	3	2.66		5	22	16.94
3	6	4.70		6	41	31.23

Our future goals concern IDENTIFY and they are:

1) to prove that it uses at most $n.m$ queries for $n > 6$;

2) to reduce the recent exponential time complexity of the algorithm to a polynomial one (towards n and m).

REFERENCES

[1] V. Bakoev. Some properties of one matrix structure at monotone Boolean functions. Proc. of the EWM Intern. Workshop on Groups and Graphs. Varna, Bulgaria, 2002, 5–8.

[2] E. Boros, P. L. Hammer, T. Ibaraki, K. Kawakami. Polynomial time recognition of 2-monotonic positive Boolean functions given by an oracle. *SIAM J. Comput.* **26**, No 1 (Feb. 1997) 93–109.

[3] V. Gurvich, L. Khachiyan. On generating the irredundant conjunctive and disjunctive normal forms of monotone Boolean functions. *Discr. Appl. Mathematics*, **96–97** (1999), 363–373.

[4] `http://home.san.rr.com/ronz/Articles/998Dedekind.cpp.html`

[5] D. S. Johnson, M. Yannakakis. On generating all maximal independent sets. *Inform. Process. Letters*, **27**, (1988), 119–123.

[6] K. Makino, T. Ibaraki. A fast and simple algorithm for identifying 2-monotonic positive Boolean functions. *J. Algorithms*, **26** (1998), 291–305.

[7] `http://mathpages.com/home/kmath030.htm`, Dedekind's Problem

[8] `http://mathpages.com/home/kmath094.htm`, Generating the M. B. Functions

[9] I. Shmulevich, A. Korshunov, J. Astola. Almost all monotone Boolean functions are polynomially learnable using membership queries. *Inform. Process. Letters*, **79** (2001), 211–213.

[10] В. Бакоев. Генериране на подмножествата на частично наредено множество със запазване на сянката. Докл. от IV конф. "Приложения на математиката в икономиката и проблеми на обучението по математика и информатика", 1995, ВФСИ "Д.А. Ценов" и СМБ-Свищов, 186–192.

[11] М. М. Ковалев, П. Миланов. Монотонные функции многозначной логики и суперматроиды, *Журнал вычисл. математики и математ. физики*, **24**, є 5 (1984), 786–789.

Valentin P. Bakoev
Faculty of Pedagogy
University of Veliko Tarnovo
V. Tarnovo, Bulgaria

ГЕНЕРИРАНЕ И ИДЕНТИФИЦИРАНЕ НА МОНОТОННИТЕ БУЛЕВИ ФУНКЦИИ

Валентин П. Бакоев

В работата са представени три алгоритъма, основаващи се на една матрична структура. Първият от тях генерира лексикографски монотонните булеви функции на n променливи. Вторият определя лексикографски първия минимален истинен (респ. последния максимален неистинен) вектор на непозната функция. Той обслужва третия алгоритъм, който идентифицира непозната монотонна функция f на n променливи чрез използване само на т. нар. „въпроси за членство“ – дали стойността на функцията върху даден вектор е равна на нула. При n променливи, $1 \leq n \leq 6$, алгоритъмът разпознава f, използвайки не повече от $n.m$ такива въпроси, като m е сумарната големина на множествата от минималните истинни и максималните неистинни вектори на f.