ELSEVIER

Information Processing Letters 79 (2001) 203–209

Information Processing Letters

www.elsevier.com/locate/ipl

Algorithms counting monotone Boolean functions

Robert Fidytek, Andrzej W. Mostowski, Rafał Somla, Andrzej Szepietowski *

Institute of Mathematics, University of Gdańsk, ul. Wita Stwosza 57, 80-952 Gdańsk, Poland

Received 17 August 2000; received in revised form 6 December 2000
Communicated by A. Tarlecki

Abstract

We give a new algorithm counting monotone Boolean functions of n variables (or equivalently the elements of free distributive lattices of n generators). We computed the number of monotone functions of 8 variables which is 56 130 437 228 687 557 907 788.

Keywords: Combinatorial problems; Algorithms; Monotone Boolean function; Free distributive lattice; Dedekind problem

1. Introduction

Let M_n be a free distributive lattice of n generators, with 0 and 1 included, and m_n the number of its elements. Finding a general formula for m_n is known as the Dedekind problem. The numbers $m_0 = 2$, $m_1 = 3$, $m_2 = 6$, $m_3 = 20$, and $m_4 = 168$ were computed by Dedekind. Church [4] computed $m_5 = 7581$, Ward [10] $m_6 = 7\,828\,354$, and again Church [5] $m_7 = 2\,414\,628\,040\,998$. Kleitman [8] proved that $\log_2(m_n)$ can be estimated as $\binom{n}{n/2}$.

It is commonly known that M_n is isomorphic with the lattice of monotone Boolean functions of n variables, see, e.g., Birkhoff [3, Ch. IX, par. 10, Th. 13]. This fact was commonly used [2,5,6,8,9] in counting m_n, and we do the same.

The impulse for writing this paper, rose from the fact, that there are two conflicting results concerning m_7. Church [5] and Berman et al. [1] gave $m_7 = 2\,414\,628\,040\,998$ while Lunnon [9] gave $m_7 = 2\,208\,061\,288\,138$. The authors give no details of the algorithms they used.

We give three algorithms for computing m_n. The first algorithm was used in [6]. We suppose that the second algorithm is in some sense similar to that used in [1,5]. The third algorithm is a new one. We tested our algorithms up to $n = 7$ and used the third algorithm to compute $m_8 = 56\,130\,437\,228\,687\,557\,907\,788$.

2. Preliminaries

Let B denote the set $\{0, 1\}$ and B^n the set of n-element sequences of B. A Boolean function with n variables is any function from B^n into B. There are 2^n elements in B^n and 2^{2^n} Boolean functions with n variables.

There is the order relation in B:

$$0 \leqslant 0, \qquad 0 \leqslant 1, \qquad 1 \leqslant 1$$

and the partial order in B^n: for any two elements: $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n)$ in B^n

$$x \leqslant y \quad \text{if and only if} \quad x_i \leqslant y_i \quad \text{for } 1 \leqslant i \leqslant n.$$

* Corresponding author.
E-mail address: matszp@math.univ.gda.pl (A. Szepietowski).

0020-0190/01/$ – see front matter
PII: S0020-0190(00)00230-1

The function $h : B^n \to B$ is monotone if

$$x \leqslant y \Rightarrow h(x) \leqslant h(y).$$

For example, the conjunction $C(a, b) = a \cdot b$ and disjunction $D(a, b) = a + b$ are monotone. The composition of monotone functions is monotone.

Let M_n denote the set of monotone functions with n variables. We have the partial order in M_n defined by:

$$g \leqslant h \quad \text{if and only if} \quad g(x) \leqslant h(x) \quad \text{for } x \in B^n.$$

Lemma 1 [6]. *Every monotone function $g \in M_n$ can be represented in the unique form*

$$g(x_1, x_2, \ldots, x_n) = g_0(x_2, \ldots, x_n) + g_1(x_2, \ldots, x_n) \cdot x_1$$

with $g_0, g_1 \in M_{n-1}$ and $g_0 \leqslant g_1$. Moreover $g_0(x_2, \ldots, x_n) = g(0, x_2, \ldots, x_n)$ and $g_1(x_2, \ldots, x_n) = g(1, x_2, \ldots, x_n)$.

Proof. We shall only prove that the representation is unique. Suppose that $g_0, g_1 \in M_{n-1}$ and $g_0 \leqslant g_1$ then the function

$$g(x_1, x_2, \ldots, x_n) = g_0(x_2, \ldots, x_n) + g_1(x_2, \ldots, x_n) \cdot x_1$$

is monotone (as a composition of monotone functions). Moreover $g(0, x_2, \ldots, x_n) = g_0(x_2, \ldots, x_n)$ and

$$g(1, x_2, \ldots, x_n) = g_0(x_2, \ldots, x_n) + g_1(x_2, \ldots, x_n) = g_1(x_2, \ldots, x_n).$$

The last equation follows from the fact that $g_0 \leqslant g_1$. □

Thus every function $g \in M_n$ gives a pair of functions $g_0, g_1 \in M_{n-1}$ satisfying $g_0 \leqslant g_1$ and vice versa every such pair defines one function from M_n. The pair g_0, g_1 can be treated as a monotone function from B into M_{n-1}, namely the function g described by $g(0) = g_0$ and $g(1) = g_1$.

Corollary 2. *There is one to one correspondence between M_n and the set of monotone functions from B into M_{n-1}.*

We shall represent the elements of M_n as strings of bits of length 2^n. Two elements of M_0 will be represented as 0 and 1, and any element $g \in M_n$, for $n > 0$, will be represented as the concatenation $g(0) * g(1)$.

For example, any element of $g \in M_3$ will be represented as:

$$\begin{aligned} g &= g(0) * g(1) = g(00) * g(01) * g(10) * g(11) \\ &= g(000) * g(001) * g(010) * g(011) \\ &\quad * g(100) * g(101) * g(110) * g(111). \end{aligned}$$

Elements of M_1 are: 00, 01, 11. Elements of M_2 are: 0000, 0001, 0011, 0101, 0111, 1111, or \$0, \$1, \$3, \$5, \$7, \$F in hexadecimal system. 20 elements of M_3 are: \$0, \$1, \$3, \$5, \$7, \$F, \$11, \$13, \$15, \$17, \$1F, \$33, \$37, \$3F, \$55, \$57, \$5F, \$77, \$7F, \$FF.

We have the operations of disjunction $+$ and conjunction $\cdot$ defined in M_n by

$$(\alpha_1, \ldots, \alpha_{2^n}) + (\beta_1, \ldots, \beta_{2^n}) = (\alpha_1 + \beta_1, \ldots, \alpha_{2^n} + \beta_{2^n})$$

and

$$(\alpha_1, \ldots, \alpha_{2^n}) \cdot (\beta_1, \ldots, \beta_{2^n}) = (\alpha_1 \cdot \beta_1, \ldots, \alpha_{2^n} \cdot \beta_{2^n}).$$

Lemma 3. *For every three elements $f, g, h \in M_n$, we have*

$$h \geqslant f \text{ and } h \geqslant g \quad \text{if and only if} \quad h \geqslant f + g$$

and

$$h \leqslant f \text{ and } h \leqslant g \quad \text{if and only if} \quad h \leqslant f \cdot g.$$

Applying Lemma 1 twice we can represent each function $g \in M_n$ in the form

$$g = g_{00} + g_{10} \cdot x_1 + g_{01} \cdot x_2 + g_{11} \cdot x_1 \cdot x_2,$$

where $g_{00}, g_{10}, g_{01}, g_{11} \in M_{n-2}$, $g_{00} \leqslant g_{10} \leqslant g_{11}$, and $g_{00} \leqslant g_{01} \leqslant g_{11}$.

Again this representation is unique and there is one to one correspondence between M_n and the set of monotone functions from B^2 into M_{n-2}.

Further on we shall show that for every k, $1 \leqslant k \leqslant n$, there is one to one correspondence between M_n and the set of monotone functions from B^k into M_{n-k}. To do this we need some notions.

Let $g \in M_n$ and $k \leqslant n$. For every $\sigma = (\sigma_1, \ldots, \sigma_k) \in B^k$, let $g(\sigma)$ be the monotone function $g(\sigma) \in M_{n-k}$ defined by:

$$g(\sigma)(x_{k+1}, \ldots, x_n) = g(\sigma_1, \ldots, \sigma_k, x_{k+1}, \ldots, x_n).$$

Let w_σ denote the term

$$w_\sigma = x_1^{\sigma_1} \cdots x_k^{\sigma_k},$$

where we use the convention that $x^1 = x$ and $x^0 = 1$.

Observe that if b is a single bit, σ a string of bits, and $\tau = b\sigma$ is the concatenation of b and σ, then $g(b)(\sigma) = g(\tau)$, $w_\tau = w_\sigma$ if $b = 0$, and $w_\tau = w_\sigma \cdot x_1$ if $b = 1$.

Lemma 4. *If $g \in M_n$ and $\sigma \in B^k$ then $g(\sigma)$ is monotone, i.e., $g(\sigma) \in M_{n-k}$. Moreover, if $\sigma \leqslant \tau$ then $g(\sigma) \leqslant g(\tau)$.*

Using induction on k and Lemma 1 one can easily prove the following theorem:

Theorem 5. *Let $g \in M_n$ and $1 \leqslant k \leqslant n$. Then g can be represented in the form*

$$g(x_1, \ldots, x_n) = \sum_{\sigma \in B^k} g_\sigma(x_{k+1}, \ldots, x_n) \cdot w_\sigma$$

with
(1) $g_\sigma \in M_{n-k}$,
(2) *if $\sigma \leqslant \tau$ then $g_\sigma \leqslant g_\tau$.*
This representation is unique. Every function g defined as above belongs to M_n and $g_\sigma = g(\sigma)$ for every $\sigma \in B^k$.

Corollary 6. *There is one to one correspondence between elements of M_n and monotone functions from B^k into M_{n-k}.*

We shall use the corollary in order to construct three algorithms counting the elements of M_n.

3. First algorithm

The first algorithm uses the corollary with $k = 1$.

Algorithm 1.

input: $m_{n-1} = |M_{n-1}|$
$M1[1, \ldots, m_{n-1}]$ — elements of M_{n-1}
output: $m_n = |M_n|$
$M2[1, \ldots, m_n]$ — elements of M_n

Set $k := 0$;
For $i := 1$ to m_{n-1} do
  For $j := 1$ to m_{n-1} do
    If $M1[i] \leqslant M1[j]$ then
      $k := k + 1$;
      $M2[k] := M1[i] * M1[j]$
$m_n := k$

4. Second algorithm

In the second algorithm we count the monotone functions from B^2 into M_{n-2}. We do this in the following way. For every pair of elements $u, v \in M_{n-2}$ we count monotone functions g from B^2 into M_{n-2} satisfying: $g(00) = u$ and $g(11) = v$ and sum up the results.

Let $u, v \in M_{n-2}$. Observe that for every pair $w, x \in M_{n-2}$ satisfying $u \leqslant w, x \leqslant v$ there exists a monotone function g satisfying $g(00) = u$, $g(01) = w$, $g(10) = x$, $g(11) = v$. On the other hand, $u = g(00) \leqslant g(01)$, $g(10) \leqslant g(11) = v$. Hence, the number of functions satisfying $g(00) = u$ and $g(11) = v$ is equal to $(re[u, v])^2$, where $re[u, v]$ is the number of elements laying between u and v

$$re[u, v] = \big|\{w \mid u \leqslant w \leqslant v\}\big|$$

The elements of the matrix re can be easily computed because $re = r \times r$ where r is the matrix of the order relation in M_{n-2}. $r[u, v] = 1$ if $u \leqslant v$, and $r[u, v] = 0$ otherwise.

Algorithm 2.

input: m_{n-2}
$M[1..m_{n-2}]$ — elements of M_{n-2}
output: m_n

1. Compute the matrix r,
2. Compute the matrix $re := r \times r$
3. $m_n := \sum_{i,j \in M_{n-2}} (re[i, j])^2$

5. Third algorithm (sketch)

The next algorithm counts the functions from B^4 into M_{n-4}.

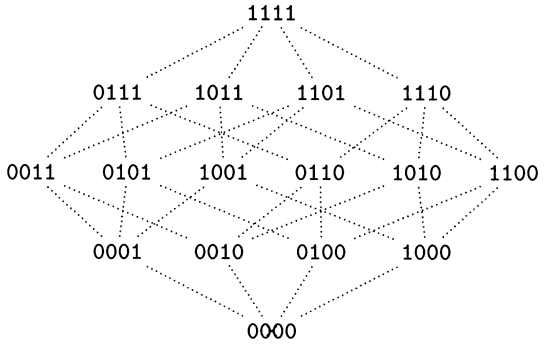

Fig. 1. The structure of B^4.

For every six elements $a, b, c, d, e, f \in M_{n-4}$ we count how many functions g satisfy: $g(0011) = a$, $g(0101) = b$, $g(1001) = c$, $g(0110) = d$, $g(1010) = e$, $g(1100) = f$ (see Fig. 1).

Or in other words we count in how many ways we can choose the values of g for other elements of B^4. Observe that the values for upper elements (1111), (0111), (1011), (1101), (1110) can be chosen independently from the values for lower elements (0000), (0001), (0010), (0100), (1000).

Consider now in how many ways one can choose the values for the upper elements (1111), (0111), (1011), (1101), (1110). First we choose $g(1111)$ which must be greater than or equal to each of the values a, b, c, d, e, f, so $g(1111)$ can be chosen from the values greater than or equal to $a + b + c + d + e + f$. Next the values $g(0111)$, $g(1011)$, $g(1101)$, $g(1110)$ can be chosen independently from each other. And $g(0111)$ must be greater than or equal to each one of $g(0011) = a$, $g(0101) = b$, $g(0110) = d$. Hence $a + b + d \leqslant g(0111) \leqslant g(1111)$. Thus $g(0111)$ can be chosen in $re[a + b + d, g(1111)]$ ways (remember that $re[u, v]$ is the number of elements between u and v). Similarly, $g(1011)$ can be chosen in $re[a + c + e, g(1111)]$ ways, $g(1101)$ in $re[b + c + f, g(1111)]$ ways, and $g(1110)$ in $re[d + e + f, g(1111)]$ ways.

The number of ways g can be extended to the upper elements is equal to

$$H = \sum_{u \geqslant a+b+c+d+e+f} re[a + b + d, u] \cdot re[a + c + e, u] \cdot re[b + c + f, u] \cdot re[d + e + f, u].$$

Similarly one can show that the number of ways g can be extended to the lower elements is equal to

$$h = \sum_{v \leqslant a \cdot b \cdot c \cdot d \cdot e \cdot f} re[v, a \cdot b \cdot c] \cdot re[v, a \cdot d \cdot e] \cdot re[v, b \cdot d \cdot f] \cdot re[v, c \cdot e \cdot f].$$

Altogether there are $H \cdot h$ functions satisfying: $g(0011) = a$, $g(0101) = b$, $g(1001) = c$, $g(0110) = d$, $g(1010) = e$ and $g(1100) = f$.

Algorithm 3 (sketch).

input: m_{n-4}
$M[1..m_{n-4}]$ — elements of M_{n-4}
output: m_n

1. Compute the matrix r,
2. Compute the matrix $re := r \times r$,
3. Set $m_n := 0$,
4. For every six elements $a, b, c, d, e, f \in M_{n-4}$ do
   (a) Set $H := 0$ and $h := 0$,
   (b) For all $u \in M_{n-4}$, $u \geqslant a + b + c + d + e + f$ do
   $$H := H + re[a + b + d, u] \cdot re[a + c + e, u] \cdot re[b + c + f, u] \cdot re[d + e + f, u],$$
   (c) For all $v \in M_{n-4}$, $v \leqslant a \cdot b \cdot c \cdot d \cdot e \cdot f$ do
   $$h := h + re[v, a \cdot b \cdot c] \cdot re[v, a \cdot d \cdot e] \cdot re[v, b \cdot d \cdot f] \cdot re[v, c \cdot e \cdot f],$$
   (d) Let $m_n := m_n + H \cdot h$.

6. Symmetries in B^4

In order to speed up the third algorithm we use some symmetries existing in B^4. We treat each element $x \in B^4$ as a function

$$x : \{1, 2, 3, 4\} \to \{0, 1\}.$$

Let S_4 denote the set of permutations on $\{1, 2, 3, 4\}$. Every permutation $\pi \in S_4$ defines the permutation on B^4:

$$\pi(x) = x \circ \pi.$$

Note that $x \leqslant y$ if and only if $\pi(x) \leqslant \pi(y)$. This permutation on B^4 generates the permutation on the set of functions from B^4 into M_{n-4}. Namely

$$\pi(g) = g \circ \pi.$$

Note that if g is monotone then $\pi(g)$ is monotone.

7. Third algorithm (more details)

We proceed with the computation in six stages. In the ith stage, $1 \leqslant i \leqslant 6$, we count functions g for which the set $\{a, b, c, d, e, f\}$ has cardinality equal to i. For $i = 1$ all six elements a, b, c, d, e, f are equal. For $i = 6$ all six elements are different.

We show how we used the symmetries described in the previous section to speed up the computation in the sixth stage ($i = 6$).

Let

$$A = 0011, \quad B = 0101, \quad C = 1001,$$

$$D = 1100, \quad E = 1010, \quad F = 0110.$$

For each set

$$\{a, b, c, d, e, f\}$$

of six different elements from M_{n-4} we count functions for which

$$g\big(\{A, B, C, D, E, F\}\big) = \{a, b, c, d, e, f\}.$$

First observe that it suffices to count functions satisfying $g(A) = a$ and then to multiply the result by 6. Indeed, for every $Z \in \{A, B, C, D, E, F\}$ there exists a permutation $\pi \in S_4$ such that $\pi(Z) = A$. The permutation π gives one to one correspondence between functions g with $g(A) = a$ and functions g with $g(Z) = a$. Observe also that always $|g^{-1}(a)| = 1$, hence for any two different $Y, Z \in \{A, B, C, D, E, F\}$ the set of functions with $g(Y) = a$ is disjoint with the set of functions with $g(Z) = a$.

In order to count functions with $g(A) = a$, we first count functions with $g(D) \neq b$. To do this we may again count only functions satisfying $g(B) = b$ and multiply the result by 4. This is because for every $Z \in \{B, C, E, F\}$, there is permutation $\pi \in S_4$ such that $\pi(A) = A$ and $\pi(Z) = B$, and π gives one to one correspondence between the set of functions

$$\big\{g \mid g(A) = a, g(B) = b\big\}$$

and the set

$$\big\{g \mid g(A) = a, g(Z) = b\big\}.$$

And because $g^{-1}(b)$ has always one element, for any two different $Y, Z \in \{B, C, E, F\}$

$$\big\{g \mid g(A) = a, g(Y) = b\big\} \cap \big\{g \mid g(A) = a, g(Z) = b\big\} = \emptyset.$$

To count functions with $g(D) = b$ we again may count only functions satisfying $g(B) = c$ and multiply the result by 4. To see this observe that for every $\pi \in S_4$, if $\pi(A) = A$ then $\pi(D) = D$; and for every $Z \in \{B, C, E, F\}$, there is permutation $\pi \in S_4$ such that $\pi(A) = A$, $\pi(D) = D$, and $\pi(Z) = B$, and π gives one to one correspondence between

$$\big\{g \mid g(A) = a,\ g(D) = b,\ g(B) = c\big\}$$

and

$$\big\{g \mid g(A) = a,\ g(D) = b,\ g(Z) = c\big\}.$$

And because $g^{-1}(c)$ has always one element, for any two different $Y, Z \in \{B, C, E, F\}$

$$\big\{g \mid g(A) = a,\ g(D) = b,\ g(Y) = c\big\} \cap \big\{g \mid g(A) = a,\ g(D) = b,\ g(Z) = c\big\} = \emptyset.$$

In the fifth stage ($i = 5$), for each set

$$\{a, b, c, d, e\}$$

of five different elements from M_{n-4} we count functions for which

$$g\big(\{A, B, C, D, E, F\}\big) = \{a, b, c, d, e\}.$$

For every

$$y \in \{a, b, c, d, e\}$$

we separately count functions g which satisfy $|g^{-1}(y)| = 2$. In each such case there exist three elements $u, v, w \in \{a, b, c, d, e\}$ such that $|g^{-1}(u)| = |g^{-1}(v)| = |g^{-1}(w)| = 1$ and we can use the symmetries as in the sixth stage.

In the fourth stage ($i = 4$) the situation is more complicated. For each set

$$\{a, b, c, d\}$$

of four different elements from M_{n-4} we count functions for which

$$g\big(\{A, B, C, D, E, F\}\big) = \{a, b, c, d\}.$$

There are two cases. First we count functions g for which there exists $y \in \{a, b, c, d\}$ such that $|g^{-1}(y)| = 3$. In this case there exist three elements $u, v, w \in \{a, b, c, d\}$ such that $|g^{-1}(u)| = |g^{-1}(v)| = |g^{-1}(w)| = 1$ and we can use the symmetries as in the sixth stage. The second case is when there are two elements $y, z \in \{a, b, c, d\}$ such that $|g^{-1}(y)| = |g^{-1}(z)| = 2$. In this case there are only two elements $u, v \in \{a, b, c, d\}$ such that $|g^{-1}(u)| = |g^{-1}(v)| = 1$ and we have to be more careful when using the symmetries.

The first stage is very easy. The second and the third stage can be computed without using symmetries, because the amount of computation they need is not big.

8. Testing algorithms

Using quite a common PC with 64 MB RAM memory, we tested Algorithm 1 (written in Pascal) for $n \leqslant 6$. It took 3 minutes to compute m_6 and we estimated that it would take about 19 months to compute m_7.

Using a program in Delphi we tested Algorithm 2, which took 1 second to compute m_6 and approximately 30 minutes to compute m_7. We did not even try to compute m_8 by Algorithm 2, since we estimated that a huge memory and absurdly long time would be needed for it. For details see [7].

Algorithm 3 computed m_7 in about 3 seconds. For computing m_8 by Algorithm 3 a much smaller memory is needed than that for the former algorithms. The time needed remains very big — by rough estimation 80 days.

9. Computation of m_8

The computation of m_8 was performed on about 40 PCs in parallel. We first implemented the third algorithm under Linux and performed computations on two computers. During 10 days they did about 30% of the work. In the meantime, a version of the algorithm working under plain DOS was prepared. Then we used 40 Pentium 130–366 MHz machines which finished the task in 3 days.

The algorithm was distributed among computers in the following way. As described in Section 7 there are six stages of computations. For $i = 1, 2, \ldots, 6$ we iterate over all subsets $K \subset M_4$ with $|K| = i$ and count (monotone) functions $g : B^4 \to M_4$ which map $\{A, B, C, D, E, F\}$ onto K. Our algorithm gives results for $i = 1, 2, 3$ in a few seconds. For $i = 4$, 5 and 6 the whole computation was divided into pieces each for 100 000–200 000 subsets of M_4 (the algorithm needs about 60 seconds to compute each piece).

Table 1

1-element subsets	2 779 585 901 901 800
2-element subsets	560 459 534 848 784 564
3-element subsets	34 135 891 057 507 346 100
4-element subsets	908 093 498 969 260 063 668
5-element subsets	10 684 727 332 563 422 442 480
6-element subsets	44 502 917 266 976 617 369 176

We enumerate the elements of M_4 as $z_1, z_2, \ldots, z_{168}$ and represent subsets of M_4 by strictly increasing sequences of numbers from $\{1, 2, \ldots, 168\}$. Each piece of computation is described by a line. For example the line

```
23 35 108-112 - - -
```

describes all sequences $y_1 < y_2 < \cdots < y_6$ for which $y_1 = 23$, $y_2 = 35$, $108 \leqslant y_3 \leqslant 112$ and y_4, y_5, y_6 are arbitrary.

The partition of the computation was saved in a text file containing one line for each piece. There were 331 pieces for 4-element subsets, 10 622 pieces for 5-element subsets and 126 477 for 6-element subsets. Now the pieces were put into packages each containing 500 pieces and these packages were distributed between computers. It took about 5–6 hours to compute one package. The results were gathered and saved back in the file and then summed up. They are presented in Table 1.

Altogether we get

$$m_8 = 56\,130\,437\,228\,687\,557\,907\,788.$$

Note that the numbers involved exceed the capacity of 64-bit arithmetic. Therefore we wrote extended arithmetic based on one 64- and one 32-bit variable.

References

[1] J. Berman, A. Burger, P. Koehler, The free distributive lattice on seven generators, Notices Amer. Math. Soc. 22 (1975) 75T-A242.

[2] J. Berman, P. Koehler, Cardinalities of finite distributive lattices, Mitteilungen aus dem Mathem. Seminar Giessen, Heft 121, Giessen, 1976.

[3] G. Birkhoff, Lattice Theory, 3rd edn., Amer. Math. Soc. Coll. Publications, Vol. 25, Amer. Math. Soc., Providence, RI, 1967.

[4] R. Church, Numerical analysis of certain free distributive structures, Duke. Math. J. 6 (1940) 732–734.

[5] R. Church, Enumeration by rank of the elements of the free distributive lattice with 7 generators, Notices Amer. Math. Soc. 12 (1965) 724.

[6] E. Czyżo, A.W. Mostowski, An algorithm for generating free distributive lattices, Bull. Acad. Polon. Sci. Series Math. Astronom. Phys. 16 (1968) 593–595.

[7] R. Fidytek, Counting elements of a free distributive lattice by a computer, Master degree thesis, Mathematical Institute, University of Gdańsk, 2000 (in Polish).

[8] D. Kleitman, On Dedekinds Problem: The number of monotone Boolean functions, Proc. Amer. Math. Soc. 21 (1969) 677–682.

[9] F. Lunnon, The IU function: The size of a free distributive lattice, in: D.J.A. Welsh (Ed.), Combinatorial Mathematics and its Applications, Academic Press, London, 1971, pp. 173–181.

[10] M. Ward, Note on the order of free distributive lattices, Bull. Amer. Math. Soc. 52 (1946) 423.