Journal of Integer Sequences, Vol. 25 (2022),
Article 22.7.7

On the Number of Inequivalent Monotone Boolean Functions of 8 Variables

Bartłomiej Pawelski
Institute of Informatics
University of Gdańsk
Wita Stwosza 57
80-952 Gdańsk
Poland
bartlomiej.pawelski@ug.edu.pl

Abstract

In this paper, we present algorithms for determining the number of fixed points in the set of monotone Boolean functions under a given permutation of input variables. Then, using Burnside's lemma, we determine the number of inequivalent monotone Boolean functions of 8 variables. The number obtained is 1,392,195,548,889,993,358.

1 Introduction

A monotone Boolean function (MBF) is any Boolean function that can be implemented using only conjunctions and disjunctions [10]. Let D_n be the set of all monotone Boolean functions of n variables, and d_n the cardinality of this set; d_n is also known as the n-th Dedekind number (sequence A000372 in the OEIS (*On-Line Encyclopedia of Integer Sequences*)).

Two Boolean functions are *equivalent* if the first function can be transformed into the second function by any permutation of input variables. Let I_n be the set of all n input variables of a Boolean function. There are $n!$ possible permutations of I_n—therefore there are at most $n!$ MBFs in one equivalence class. Let R_n denote the set of all equivalence classes of D_n and let r_n denote the cardinality of this set; r_n is described by OEIS sequence A003182.

In 1985, Chuchang and Shoben [4] came up with the idea to calculate the r_n using Burnside's lemma. In the following year they calculated r_7 [5]. Their result was confirmed by Stephen and Yusun in 2012 [10]. In 2018, Assarpour [1] gave lower bound of r_8: namely, 1,392,123,939,633,987,512.

In 1990, Wiedemann calculated d_8 [11]. His result was confirmed in 2001 by Fidytek, Mostowski, Somla, and Szepietowski [8].

In this paper we develop algorithms for counting fixed points in D_n under a given permutation of I_n. Then, we use Burnside's lemma to calculate $r_8 = 1,392,195,548,889,993,358$.

n	d_n	r_n
0	2	2
1	3	3
2	6	5
3	20	10
4	168	30
5	7,581	210
6	7,828,354	16,353
7	2,414,682,040,998	490,013,148
8	56,130,437,228,687,557,907,788	1,392,195,548,889,993,358

Table 1: Known values of d_n and r_n.

2 Idea of calculating r_n using Burnside's lemma

Burnside's lemma is a standard combinatorial tool for counting the orbits of set under group action. Let G denote a finite group that acts upon a set X. Burnside's lemma asserts that the number of orbits $|X/G|$ with respect to the action equals the average size of the sets $X^g = \{x \in X \mid gx = x\}$ when ranging over each $g \in G$ [6, 7]:

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|. \tag{1}$$

Define S_n to be the symmetric group of I_n. Each permutation $\pi \in S_n$ can be written as a product of disjoint cycles. Define the *cycle type* of π to be the tuple of lengths of its disjoint cycles in increasing order. For example, the cycle type of permutation $\pi = (1\ 2)(3\ 4\ 5)$ is $(2, 3)$, and its total length is 5. The number of different cycle types in S_n for the appropriate value of n is described by the OEIS sequence A000041. For $n = 7$ there are 15 cycle types, and for $n = 8$ there are 22 cycle types (see the detailed list in Table 6 and Table 7).

In 1985, Chuchang and Shoben [4] presented the following application of Burnside's lemma to calculate r_n:

$$r_n = \frac{1}{n!} \sum_{\pi \in S_n} |\phi_n(\pi)|, \tag{2}$$

where

- r_n = number of equivalence classes in D_n
- $\phi_n(\pi)$ = set of all fixed points in D_n under permutation $\pi \in S_n$.

They also used the fact that $|\phi_n(\pi)|$ is invariant under permutations with the same cycle type (also see [7, Remark 287]). We have

$$r_n = \frac{1}{n!} \sum_{i=1}^{k} \mu_i \phi(\pi_i), \tag{3}$$

where

- k = number of different cycle types in S_n
- i = index of the cycle type
- μ_i = number of permutations $\pi \in S_n$ with cycle type i
- π_i = representative permutation $\pi \in S_n$ with cycle type i.

The formula for determining μ for each cycle type is as follows:

$$\mu_i = \frac{n!}{(l_1^{k_1} \cdot l_2^{k_2} \cdots l_r^{k_r})(k_1! \cdot k_2! \cdots k_r!)} \tag{4}$$

with cycle type of r various lengths of cycles, and k_1 cycles of length l_1, k_2 cycles of length $l_2, \ldots, k_r$ cycles of length l_r [7, Proposition 69]. Note that in this formula 1-cycles are not suppressed. Precomputed values of μ can be found in the OEIS sequence A181897.

3 Algorithms counting fixed points in D_n under a given permutation of I_n

The most difficult subproblem to compute r_n using Burnside's lemma is fast counting the fixed points of D_n under a given permutation of I_n.

Let B^n denote the power set of I_n. Each element in B^n represents one of 2^n possible inputs of the Boolean function. Every permutation acting on I_n regroups elements in B^n and D_n. We use the notation $\varnothing, x_1, x_2, x_1x_2, x_3, \ldots, x_1x_2x_3\cdots x_n$ to describe elements in B^n. We represent each Boolean function of n variables by the binary string of length 2^n. Each i-th bit of function in this representation is Boolean output where the argument is an element from B^n standing in the same position.

For example, consider the following truth table:

$\varnothing$	x_1	x_2	x_1x_2	x_3	x_1x_3	x_2x_3	$x_1x_2x_3$
0	0	0	0	1	1	1	1

Table 2: MBF of three variables that returns true iff x_3 is true.

MBF from Table 2 can be represented as integer 15 for more convenient computer processing. All 6 MBFs in D_2 written as integers are: 0, 1, 3, 5, 7 and 15.

For counting fixed points in D_n after acting with a specific permutation $\pi \in S_n$ it is necessary to lift $\pi \in S_n$ to $\pi' \in S_{B^n}$. For example, consider permutation $\pi = (1\ 2\ 3)$ and look at how it regroups elements in B^3:

	0	1	2	3	4	5	6	7
(1)	$\varnothing$	x_1	x_2	x_1x_2	x_3	x_1x_3	x_2x_3	$x_1x_2x_3$
(1 2 3)	$\varnothing$	x_3	x_1	x_1x_3	x_2	x_2x_3	x_1x_2	$x_1x_2x_3$

Table 3: Regrouping elements in B^3 under $\pi = (1\ 2\ 3)$.

Therefore $\pi = (1\ 2\ 3)$ lifts to $\pi'(0)(1\ 2\ 4)(3\ 6\ 5)(7)$. Each cycle designates points belonging to the same orbit. Points in each orbit are set to the same value in each $x \in \phi_n(\pi)$.

In this case, two conditions must be met: each function in $\phi_n(\pi)$ under $\pi = (1\ 2\ 3)$ has to have:

- 1-st, 2-nd and 4-th bit set on the same value
- 3-rd, 5-th and 6-th bit set on the same value

Hence, all members of $\phi_3(\pi)$ under $\pi = (1\ 2\ 3)$ can be simply found by iteration through all 20 elements in D_3 and checking which are satisfying the above conditions:

	n-th bit of MBF							
MBF written as integer	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
23	0	0	0	1	0	1	1	1
127	0	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1

Table 4: List of five fixed points in D_3 under $\pi = (1\ 2\ 3)$.

3.1 Generating the set of all fixed points in D_n under permutation of cycle type of total length n

Instead of doing a naive lookup in D_n for functions satisfying given conditions, we can generate $\phi_n(\pi)$ directly.

Given a poset $P = (X, \leq)$, downset of P is such a subset $S \subseteq X$ that for each $x \in S$ all elements from $X \leq x \in S$. D_n is equivalent to the set of all downsets of B^n—therefore each element in D_n is equivalent to some downset of B^n [3].

Two conditions must be met to generate MBF which is the fixed point in D_n under the given permutation π:

- All points in the same orbit of π' should be set to the same value—0 or 1.
- Value of points must respect the order of set inclusion.

For example, consider permutation $\pi = (1\ 2)(3\ 4)$. After lifting it into permutation of B^4, we get $\pi' = (0)(1\ 2)(3)(4\ 8)(5\ 10)(6\ 9)(7\ 11)(12)(13\ 14)(15)$.

Now, let us transform this permutation into a binary poset of orbits ordered by set inclusion. Orbits in the following example are represented by their smallest representative:

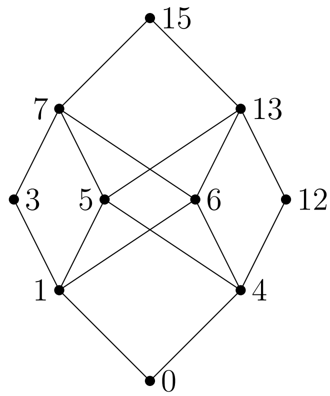
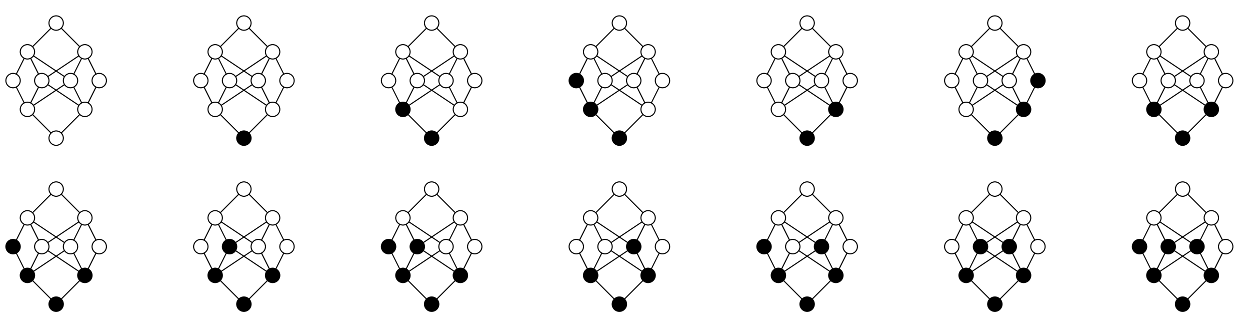
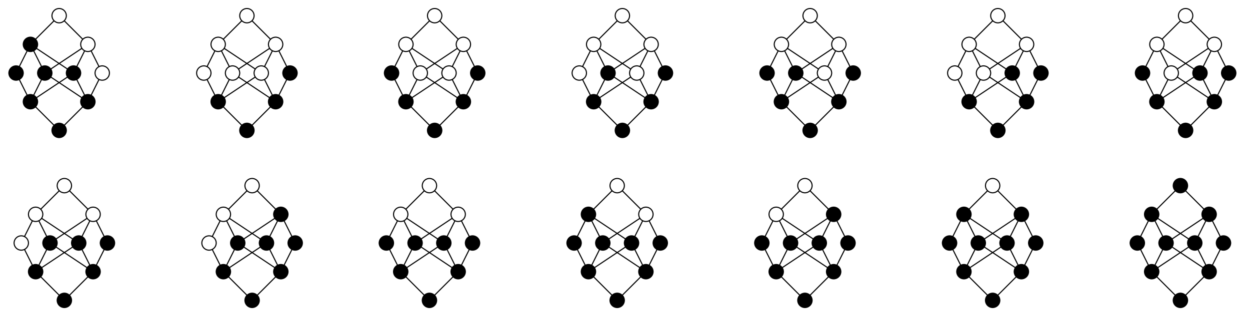


Figure 1: Poset of orbits of B_4 under $\pi = (1\ 2)(3\ 4)$ ordered by set inclusion.

Now it is only necessary to generate all downsets of this poset. In this case, the number of all downsets is 28:





The set of structures thus obtained is equivalent to $\phi_4(\pi)$ under $\pi = (1\ 2)(3\ 4)$. One can unpack the downsets obtained thereby to the integer representation of MBF of 2^n length.

This algorithm is being used only to generate $\phi_n(\pi)$ when π has a cycle type of total length n—for example, we use Algorithm 1 to generate $\phi_4(\pi)$ under $\pi = (1\ 2)(3\ 4)$, but to generate $\phi_5(\pi)$ under the same permutation it is cheaper computationally to use Algorithm 2.

Algorithm 1 Generate $\phi_n(\pi)$ under permutation of cycle type of total length n

Input: Cycle type i of total length n
Output: Set $S = \phi_n(\pi)$

Determine representative $\pi \in S_n$ of cycle type i
Lift π into $\pi' \in S_{B^n}$
Generate set Orb_i containing all orbits in π'
Order Orb_i into poset P by set inclusion
Initialize set S of downsets of P
Add two downsets: $\{\}$ and $\{0\}$ to S
for all elements $a \in P$ **do**
    for all elements $b \in S$ **do**
        if $(b \cup a)$ is downset of P **then**
            Add downset $(b \cup a)$ to S
        end if
    end for
end for

3.2 Generating the set of all fixed points in D_{n+1} under permutation of cycle type of total length n

Each ω in D_{n+1} can be split into two functions (α, β) from D_n. Moreover, there is a relation $\alpha \preceq \beta$, which means that for every i-th bit $\alpha_i \leq \beta_i$ [2, 8]. For all $\pi \in S_n$, as $\phi_{n+1}(\pi)$ is subset of D_{n+1}, each ω in $\phi_{n+1}(\pi)$ can be split into two functions (α, β).

Constructing ω from α is simply adding new variable (x_{n+1}) to α. β contains data about each possible intersection of α with (x_{n+1}). Hence, α clearly belongs to $\phi_n(\pi)$—same as β, as its variables are regrouped in the same way. Only difference between them is additional variable (x_{n+1}) which is fixed point of π, added to each element in β.

	0	1	2	3	4	5	6	7
(1)	$\varnothing$	x_1	x_2	x_1x_2	x_3	x_1x_3	x_2x_3	$x_1x_2x_3$
(1 2)	$\varnothing$	x_2	x_1	x_1x_2	x_3	x_2x_3	x_1x_3	$x_1x_2x_3$

Table 5: Regrouping of elements in B^3 under $\pi = (1\ 2)$.

Hence, we can take advantage of well-known algorithms for determining Dedekind numbers (for example [8, 11]), but instead of giving D_n on input, $\phi_n(\pi)$ will be given.

To construct Algorithm 2 we use a similar approach that was used by Fidytek et al. [8, Algorithm 1]. Note that any algorithm from [8] will do the job, however, other algorithms don't return a set, but its cardinality.

Algorithm 2 Generating $\phi_{n+1}(\pi)$ under permutation π of cycle type of total length n

Input: Cycle type i of total length n
Output: Set $S = \phi_{n+1}(\pi)$

Use Algorithm 1 to generate $S' = \phi_n(\pi)$
Convert all elements in S' to integers of length 2^n bits
Initialize set S of integers of length 2^{n+1} bits
for all elements $a \in S'$ **do**
  for all elements $b \in S'$ **do**
    if $(a \mid b) = b$ **then** ▷ “|” is bitwise “OR”
      Add integer $((a << 2^n) \mid b)$ to S ▷ “<<” is logical shift
    end if
  end for
end for

3.3 Determining $|\phi_8(\pi)|$ under $\pi = (1\ 2)(3\ 4)(5\ 6)(7\ 8)$

Determining $|\phi_8(\pi)|$ under $\pi = (1\ 2)(3\ 4)(5\ 6)(7\ 8)$ is too memory-intensive for Algorithm 1 considering the resources at hand. The width of the poset of orbits of the superset of $\pi = (1\ 2)(3\ 4)(5\ 6)(7\ 8)$ is 38, so the weak lower bound of $|\phi_8(\pi)|$ is $2^{38} = 274877906944$. In practice, even the machine with 128GB RAM is insufficient to store such a number of downsets—so there was a need to develop a better algorithm for this particular case.

The idea of a cheaper calculation of this number was based on Wiedemann's approach [11]. He used the fact that each function from D_{n+2} can be split into 4 functions from D_n: $\alpha_w, \beta_w, \gamma_w, \delta_w$, and there are following dependencies: $\alpha_w \preceq \beta_w \preceq \delta_w$, $\alpha_w \preceq \gamma_w \preceq \delta_w$.

We use a similar approach based on splitting each function from $\phi_{n+2}(\pi)$ into 4 parts. We focus on a special case—when $\pi \in S_{n+2}$ is the product of disjoint 1-cycles and at least one 2-cycle. Let τ denote such the permutation. In other words, $\tau = \tau_1 \cdots \tau_x$, and τ_x is 2-cycle: $(n{+}1\ n{+}2)$. Let σ denote permutation such that $\sigma \circ \tau_x = \tau$.

We can split each function from $\phi_{n+2}(\pi)$ into the following functions: $\alpha, \delta \in \phi_n(\sigma)$ and $\beta, \gamma \in D_n$. Moreover, $\alpha \preceq \beta \preceq \delta$, $\alpha \preceq \gamma \preceq \delta$, and $\gamma = \beta((1\ 2))$.

For example, $\tau = (1\ 2)(3\ 4)$ lifts to $\tau'(0)(1\ 2)(3)(4\ 8)(5\ 10)(6\ 9)(7\ 11)(12)(13\ 14)(15)$. $\sigma = (1\ 2)$. Using the above approach we break it down into three parts:

- α as $(0)(1\ 2)(3)$; being function from $\phi_2(\sigma)$
- $\beta\gamma$ as $(4\ 8)(5\ 10)(6\ 9)(7\ 11)$ being pairs of functions from D_n such that $\gamma = \beta((1\ 2))$
- δ as $(12)(13\ 14)(15)$, being function from $\phi_2(\sigma)$.

Knowing how each function in $\phi_{n+2}(\tau)$ can be split into two functions from D_n and two functions from $\phi_n(\sigma)$, we can derive Algorithm 3:

Algorithm 3 Determining $|\phi_{n+2}(\tau)|$

Input: D_n and $\phi_n(\sigma)$
Output: $|\phi_{n+2}(\tau)|$

Initialize $k = 0$,
for all $\beta \in D_n$ **do**
    Determine $\gamma = \beta((1\ 2))$
    Initialize $down = 0$, $up = 0$
    for all $\alpha \in \phi_n(\sigma)$ **do**
        if $(\alpha \preceq (\beta \mid \gamma))$ **then** ▷ "|" is bitwise "OR"
            $down = down + 1$
        end if
    end for
    for all $\delta \in \phi_n(\sigma)$ **do**
        if $((\beta\ \&\ \gamma) \preceq \delta)$ **then** ▷ "&" is bitwise "AND"
            $up = up + 1$
        end if
    end for
    $k = k + up \cdot down$
end for

As all above-described algorithms are sufficient to count $|\phi_8(\pi)|$ for all $\pi \in S_8$, we do not explore a more generalized case of Algorithm 3—when π has at least one disjoint 2-cycle. Performing calculations using a similar approach should speed-up counting, but the relation between β and γ is more complex than in above-described special case. However, derivation of such a generalized algorithm seems essential in the future computation of r_9–but it will only be countable after computation of d_9.

4 Implementation and results

The algorithms were implemented in Java and run on a computer with an Intel Core i7-9750H processor. The results were tested and compared with the results of Chuchang and Shoben [5] for r_7. We found two misprints in their paper, clearly made during the typing process. Namely, it says that μ_{11} is 540 (instead of 504), and $\phi_7(\pi_3)$ is 20688224 (instead of 2068224). We give therefore a complete, correct table of detailed calculation results for r_7. The total computation time of r_8 was approximately a few minutes (with d_8 precomputed).

i	π_i	μ_i	$\phi_7(\pi_i)$
1	(1)	1	2414682040998
2	(12)	21	2208001624
3	(123)	70	2068224
4	(1234)	210	60312
5	(12345)	504	1548
6	(123456)	840	766
7	(1234567)	720	101
8	(12)(34)	105	67922470
9	(12)(345)	420	59542
10	(12)(3456)	630	26878
11	(12)(34567)	504	264
12	(123)(456)	280	69264
13	(123)(4567)	420	294
14	(12)(34)(56)	105	12015832
15	(12)(34)(567)	210	10192

$$r_7 = \frac{1}{5040} \sum_{i=1}^{k=15} \mu_i \phi_7(\pi_i) = 490013148$$

Table 6: Detailed calculation results for r_7.

i	π_i	μ_i	$\phi_8(\pi_i)$
1	(1)	1	56130437228687557907788
2	(12)	28	101627867809333596
3	(123)	112	262808891710
4	(1234)	420	424234996
5	(12345)	1344	531708
6	(123456)	3360	144320
7	(1234567)	5760	3858
8	(12345678)	5040	2364
9	(12)(34)	210	182755441509724
10	(12)(345)	1120	401622018
11	(12)(3456)	2520	93994196
12	(12)(34567)	4032	21216
13	(12)(345678)	3360	70096
14	(123)(456)	1120	535426780
15	(123)(4567)	3360	25168
16	(123)(45678)	2688	870
17	(1234)(5678)	1260	3211276
18	(12)(34)(56)	420	7377670895900
19	(12)(34)(567)	1680	16380370
20	(12)(34)(5678)	1260	37834164
21	(12)(345)(678)	1120	3607596
22	(12)(34)(56)(78)	105	2038188253420

$$r_8 = \frac{1}{40320} \sum_{i=1}^{k=22} \mu_i \phi_8(\pi_i) = 1392195548889993358$$

Table 7: Detailed calculation results for r_8.

References

[1] A. Assarpour, *List, Sample, and Count*, Ph.D. thesis, CUNY Graduate Center, 2018. Available at https://academicworks.cuny.edu/gc_etds/2869.

[2] V. Bakoev, One more way for counting monotone Boolean functions, in *Thirteenth International Workshop on Algebraic and Combinatorial Coding Theory*, Pomorie, Bulgaria (2012), pp. 47–52. Available at http://www.moi.math.bas.bg/moiuser/~ACCT2012/b8.pdf.

[3] F. a Campo, Relations between powers of Dedekind numbers and exponential sums related to them, *J. Integer Sequences* **21** (2018), Article 18.4.4.

[4] C. C. Liu and S. B. Hu, A mechanical algorithm of equivalent classification for free distributive lattices, *Chinese J. Comput.* **3** (2) (1985), 128–135. In Chinese.

[5] C. C. Liu and S. B. Hu, A note on the problem of computing the number of equivalence classes of free distributive lattices, *J. Wuhan Univ. Natur. Sci. Ed.* (1986), no. 1, 13–17. in Chinese.

[6] P. Drube and P. Pongtanapaisan, Annular non-crossing matchings, *J. Integer Sequences* **19** (2016), Article 16.2.4.

[7] R. Earl, Groups and group actions, lecture notes. Available at `https://courses-archive.maths.ox.ac.uk/node/view_material/43836`.

[8] R. Fidytek, A. W. Mostowski, R. Somla, and A. Szepietowski, Algorithms counting monotone Boolean functions, *Inform. Process. Lett.* **79** (2001), 203–209.

[9] N. J. A. Sloane et al., The On-Line Encyclopedia of Integer Sequences, 2022. Available at `https://oeis.org`.

[10] T. Stephen and T. Yusun, Counting inequivalent monotone Boolean functions, *Discrete Appl. Math.*, **167** (2014), 15–24.

[11] D. Wiedemann, A computation of the eighth Dedekind number, *Order* **8** (1991), 5–6.

2010 *Mathematics Subject Classification*: Primary 06E30.

Keywords: Dedekind number, monotone Boolean function, inequivalent monotone Boolean function.

(Concerned with sequences A000041, A000372, A003182, and A181897.)

Received March 29 2022; revised versions received April 6 2022; August 15 2022; August 19 2022; August 23 2022; August 31 2022. Published in *Journal of Integer Sequences*, September 21 2022.

Return to Journal of Integer Sequences home page.