

### FIELD OF SCIENCE: NATURAL SCIENCES SCIENTIFIC DISCIPLINE: MATHEMATICS

## **DOCTORAL DISSERTATION**

# Counting and generating monotone Boolean functions

Author: Bartłomiej Pawelski

Supervisor: Prof. dr hab. Andrzej Szepietowski

Gdańsk, 2024

### **OŚWIADCZENIE**

Ja, niżej podpisany oświadczam, że przedłożona praca dyplomowa została wykonana przeze mnie samodzielnie, nie narusza praw autorskich, interesów prawnych i materialnych innych osób.

data

podpis

# OŚWIADCZENIE

Wyrażam zgodę na udostępnienie osobom zainteresowanym mojej pracy dyplomowej dla celów naukowo-badawczych. Zgoda na udostępnienie pracy dyplomowej nie oznacza wyrażenia zgody na kopiowanie pracy dyplomowej w całości lub w części.

data podpis

### Contents

List of	List of papers included in the dissertation				
Abstra	act in English	7			
Stresz	czenie w języku polskim	9			
Summ	ary of the dissertation	11			
1	Introduction	11			
2	Preliminaries	13			
	2.1 Posets	13			
	2.2 Permutations acting on a Boolean function	15			
3	Counting inequivalent monotone Boolean functions (Paper A and C)	16			
4	Divisibility properties of Dedekind numbers (Paper B)	18			
5	Counting self-dual monotone Boolean functions (Paper D)	19			
6	Concluding remarks and future work	20			
Biblio	graphy	21			
Article	es in the scope of dissertation	24			
	Bartłomiej Pawelski, On the number of inequivalent monotone Boolean func-				
	tions of 8 variables	24			
	Bartłomiej Pawelski, Andrzej Szepietowski, Divisibility properties of Dedekind				
	numbers	35			
	Bartłomiej Pawelski, On the number of inequivalent monotone Boolean func-				
	tions of 9 variables	50			
	Bartłomiej Pawelski, Andrzej Szepietowski, Counting self-dual monotone				
	Boolean functions	57			

### List of papers included in the dissertation

- (A) B. Pawelski. On the number of inequivalent monotone Boolean functions of 8 variables.*J. Integer Sequences* 25 (2022)
- (B) B. Pawelski, A. Szepietowski. Divisibility properties of Dedekind numbers. J. Integer Sequences 26 (2023)
- (C) B. Pawelski, On the number of inequivalent monotone Boolean functions of 9 variables. *IEEE Trans. Inf. Theory* **70** (2024)
- (D) B. Pawelski, A. Szepietowski. Counting self-dual monotone Boolean functions. Arxiv preprint (2024)

### **Abstract in English**

In this dissertation, we present algorithms and theorems relevant to the enumeration of various classes of monotone Boolean functions. This work consists of four papers, three published in peer-reviewed journals and one under review.

We denote the number of monotone Boolean functions of n variables as  $d_n$ . Two monotone Boolean functions are said to be equivalent if one can be obtained from the other function through any permutation of input variables. Let  $r_n$  denote the number of inequivalent monotone Boolean functions of n variables. By  $\lambda_n$  we denote the number of self-dual monotone Boolean functions of n variables, and by  $q_n$  we denote the number of inequivalent self-dual monotone Boolean functions of n variables.

In Paper A we calculate the value:

 $r_8 = 1392195548889993358.$ 

In Paper B we prove the congruence:

$$d_9 \equiv 6 \pmod{210}.$$

In Paper C, we calculate the value:

 $r_9 = 789204635842035040527740846300252680.$ 

In Paper D, we confirm the previously known result:

$$\lambda_9 = 423295099074735261880,$$

and we calculate:

$$q_8 = 6001501$$

### Streszczenie w języku polskim

W niniejszej rozprawie prezentujemy nowe metody i algorytmy do zliczania pewnych klas monotonicznych funkcji boolowskich. Składa się ona z czterech artykułów, z których trzy zostały opublikowane w recenzowanych czasopismach, a jeden jest w trakcie recenzji.

Liczbę wszystkich monotonicznych funkcji boolowskich n zmiennych oznaczamy jako  $d_n$ . Dwie monotoniczne funkcje boolowskie nazywamy *równoważnymi*, jeśli pierwszą funkcję można otrzymać z drugiej poprzez dowolną permutację zmiennych wejściowych. Przez  $r_n$  oznaczamy liczbę nierównoważnych monotonicznych funkcji boolowskich n zmiennych. Przez  $\lambda_n$  oznaczamy liczbę samodualnych monotonicznych funkcji boolowskich n zmiennych, a przez  $q_n$  oznaczamy liczbę nierównoważnych samodualnych monotonicznych funkcji boolowskich n zmiennych, a przez  $q_n$  oznaczamy liczbę nierównoważnych samodualnych monotonicznych funkcji boolowskich n zmiennych.

W Artykule A obliczamy wartość:

 $r_8 = 1392195548889993358.$ 

W Artykule B dowodzimy kongruencję:

$$d_9 \equiv 6 \pmod{210}.$$

W Artykule C obliczamy wartość:

 $r_9 = 789204635842035040527740846300252680.$ 

W Artykule D potwierdzamy znany wynik:

$$\lambda_9 = 423295099074735261880,$$

oraz obliczamy:

$$q_8 = 6001501.$$

### Summary of the dissertation

### **1** Introduction

Consider the set  $B = \{0, 1\}$ . A Boolean function is a function of the form  $f : B^n \to B$ . There are  $2^n$  elements in  $B^n$  and  $2^{2^n}$  Boolean functions of n variables. There is a partial order on B:  $0 \le 0, 0 \le 1$ , and  $1 \le 1$ . This partial order extends naturally to  $B^n$ : for any two elements  $x = (x_1, \ldots, x_n)$  and  $y = (y_1, \ldots, y_n)$  in  $B^n$ , we have  $x \le y$  if and only if  $x_i \le y_i$  for all i. A Boolean function f is said to be *monotone* if for any  $x, y \in B^n, x \le y$  implies  $f(x) \le f(y)$ . Let  $D_n$  denote the set of all monotone Boolean functions of n variables and let  $d_n$  denote the cardinality of  $D_n$ , also known as the n-th Dedekind number. Dedekind numbers are listed in the *On-Line Encyclopedia of Integer Sequences* (OEIS) sequence A000372 (see Table 1.1). The term  $d_n$  also corresponds to the number of simple games with n players in minimal winning form, the number of antichains of subsets of an n set, the number of Sperner families and the cardinality of a free distributive lattice on n generators [18].

We represent Boolean functions of n variables using binary sequences of length  $2^n$  as follows: the two functions in  $D_0$  are 0 and 1. The set  $D_{n+1}$  is represented by the set of all concatenations  $f_0 \cdot f_1$ , where  $f_0, f_1 \in D_n$ , and  $f_0 \leq f_1$ . Hence, the three functions in  $D_1$  are: 00, 01, and 11. The six functions in  $D_2$  are: 0000, 0001, 0011, 0101, 0111, and 1111.

This representation of the function  $f \in D_n$  is the binary sequence of its values. For example, the sequence 0011 represents the function  $f \in D_2$  where f(00) = 0, f(01) = 0, f(10) = 1, and f(11) = 0.

For  $f, g \in D_n$ , the union of these functions is written as  $f \lor g$ , and the intersection as  $f \land g$ . In terms of their binary representations, the union is given by the bitwise OR operation and the intersection by the bitwise AND operation.

Boolean functions can also be represented by Boolean expressions. Monotone Boolean functions are functions defined only with unions and intersections (without negation). For example, for n = 2, the six functions in  $D_2$  can be represented as follows:

0000:  $f(x_1, x_2) = 0$ 0001:  $f(x_1, x_2) = x_1 \land x_2$ 0011:  $f(x_1, x_2) = x_1$ 0101:  $f(x_1, x_2) = x_2$ 0111:  $f(x_1, x_2) = x_1 \lor x_2$ 1111:  $f(x_1, x_2) = 1$ 

For n = 3, the majority function 00010111 belonging to  $D_3$  can be expressed as  $f(x_1, x_2, x_3) = (x_1 \wedge x_2) \lor (x_1 \wedge x_3) \lor (x_2 \wedge x_3)$ .

Determining the exact values of  $d_n$  has been a longstanding computational challenge. This problem was posed by Dedekind in 1897 [9], who published the values of  $d_n$  for  $n \leq 4$ . The values  $d_5$  and  $d_7$  were provided by Church [5, 6], while Ward [26] determined the value of  $d_6$ . In 1990, Wiedemann calculated  $d_8$  [27]. His result was confirmed in 2001 by Fidytek, Mostowski, Somla, and Szepietowski [10]. The calculation of  $d_9$  was independently achieved in 2023 by Jäkel [11] and Van Hirtum, De Causmaecker, Goemaere, Kenter, Riebler, Lass, and Plessl [16]. We announced that  $d_9 \equiv 6 \mod 210$  in the preprint of Paper B, shortly before these papers appeared.

Two monotone Boolean functions are said to be *equivalent* if one can be obtained from the other through any permutation of input variables. Let  $R_n$  represent the set of all equivalence classes of  $D_n$ , and let  $r_n$  denote the cardinality of the set  $R_n$ , also known as the number of inequivalent monotone Boolean functions of n variables. The values of  $r_n$  are listed in the OEIS sequence A003182 (see Table 1.2).

In 1985 and 1986, Liu and Hu [12, 13] calculated  $r_n$  for n up to 7. Independently,  $r_7$  was calculated by Stephen and Yusun [24]. In Paper A (2021), we calculate the value of  $r_8$ . The value of  $r_8$  was independently reported in 2022 by Carić and Živković [7]. In Paper C (2024), we report the value of  $r_9$ .

For each  $x \in D_n$ , we have dual  $x^* \in D_n$ , which is obtained by reversing and negating all bits. For example,  $1111^* = 0000$  and  $0001^* = 0111$ . An element  $x \in D_n$  is *self-dual* if  $x = x^*$ . For example, 0101 and 0011 are self-duals in  $D_2$ . Let  $\Lambda_n$  be the set of all self-dual monotone Boolean functions of n variables, and let  $\lambda_n$  denote the cardinality of this set. The value  $\lambda_n$  is also known as the n-th Hosten-Morris number (A001206 in OEIS). Let  $Q_n$  denote the set of all equivalence classes in  $\Lambda_n$  and let  $q_n$  denote  $|Q_n|$ . Values of  $q_n$  are described by A008840 OEIS sequence.

n	$d_n$	
0	2	Dedekind (1897)
1	3	Dedekind (1897)
2	6	Dedekind (1897)
3	20	Dedekind (1897)
4	168	Dedekind (1897)
5	7581	Church [5] (1940)
6	7828354	Ward [26] (1946)
7	2414682040998	Church [6] (1965)
8	56130437228687557907788	Wiedemann [27] (1991)
9	286386577668298411128469151667598498812366	Jäkel [11] (2023)
		Van Hirtum et al. [16] (2023)

Table 1.1: Known values of  $d_n$ .

The first attempt to solve the problem of determining the values of  $\lambda_n$  was made, in 1968, by Riviere [17], who determined all values up to  $\lambda_5$ . In 1972, Brouwer and Verbeek provided the values up to  $\lambda_7$  [8]. The value of  $\lambda_8$  was determined by Mills and Mills [15] in 1978. The most recent known term,  $\lambda_9$ , was obtained by Brouwer, Mills, Mills, and Verbeek [3] in 2013. The value of  $\lambda_n$  corresponds to the number of non-dominated coteries on *n* members [2, Section 1]; and also corresponds to the number of maximal linked systems.

In Paper D, we derive several algorithms for counting self-dual monotone Boolean functions. We confirm the result of [3] that  $\lambda_9$  equals 423,295,099,074,735,261,880. Furthermore, employing Burnside's lemma and techniques discussed in [19, 23], we calculate  $q_8$  to be 6,001,501.

### 2 Preliminaries

#### 2.1 Posets

A binary relation that is reflexive, antisymmetric and transitive, when defined on a set P (also called the carrier), forms a *partially ordered set*, or simply a *poset*. In Introduction, we define partial orders on B and  $B^n$ , making them posets. A partial order on  $D_n$  is defined as follows: for two functions  $f, g \in D_n$ ,  $f \leq g$  if  $f(x) \leq g(x)$  for every  $x \in B^n$ . The interval [f, g] is the set of functions  $h \in D_n$  that satisfy  $f \leq h \leq g$ . Let #[f, g] denote the cardinality of [f, g].

n	$r_n$	
0	2	
1	3	
2	5	
3	10	
4	30	
5	210	
6	16353	Liu and Hu [12] (1985)
7	490013148	Liu and Hu [13] (1986)
		Stephen and Yusun [24] (2014)
8	1392195548889993358	Pawelski [19](Paper A) (2021)
		Carić and Živković [7] (2022)
9	789204635842035040527740846300252680	Pawelski [20](Paper C) (2023)

Table 1.2: Known values of  $r_n$ .

We define  $\perp$  as the constant function that always returns 0. Similarly, we define  $\top$  as the constant function that always returns 1. Hence,  $\#[\perp, f]$  denotes the number of functions g that satisfy  $g \leq f$ , while  $\#[f, \top]$  denotes the number of functions g that satisfy  $f \leq g$ .

Two posets  $P = (X, \leq)$  and  $Q = (Y, \leq)$  are said to be *isomorphic* if there exists a bijection  $f: X \to Y$  so that for any elements  $x_1, x_2$  in  $P, x_1 \leq x_2$  holds true iff  $f(x_1) \leq f(x_2)$ . We use the equals symbol = to represent an isomorphism between two posets.

Given two posets  $(X, \leq)$  and  $(Y, \leq)$ , their Cartesian product is represented as  $X \times Y$ . In this product,  $(a, b) \leq (c, d)$  holds if and only if  $a \leq c$  and  $b \leq d$ . For two disjoint posets  $(X, \leq)$  and  $(Y, \leq)$ , the disjoint union (sum) is denoted by X + Y, with the order defined as follows:  $a \leq b$  if and only if  $(a, b \in X \text{ and } a \leq b)$  or  $(a, b \in Y \text{ and } a \leq b)$ .

Let  $P_n$  denote the chain, the poset with n elements where every pair of elements is comparable. Let  $A_n$  denote the antichain of order n, a poset with n elements where no two distinct elements are related.

A function  $f : X \to Y$  is monotone when, for any elements  $x, y \in X$  with  $x \leq y$ , we have  $f(x) \leq f(y)$ . We represent the set of all monotone functions from X to Y as  $Y^X$ . When considering two functions  $f, g \in Y^X$ , we define  $f \leq g$  to be true if  $f(x) \leq g(x)$  for every  $x \in X$ . Following this notation, we have  $B^n = B^{A_n}$  and  $D_n = B^{B^n}$ .

Lemma 1.1. [23] For three posets R, S, T:

(1) If S and T are disjoint, then  $R^{S+T} = R^S \times R^T$ .

n	$\lambda_n$	$q_n$
0	0	0
1	1	1
2	2	1
3	4	2
4	12	3
5	81	7
6	2646	30
7	1422564	716
8	229809982112	6001501
9	423295099074735261880	

Table 1.3: Known values of  $\lambda_n$  (A001206) and  $q_n$  (A008840).

(2) 
$$R^{S \times T} = (R^S)^T$$
 and  $R^{S \times T} = (R^T)^S$ .

Lemma 1.2. [23]

(a)  $A_{k+m} = A_k + A_m$ 

(b) 
$$B^{k+m} = B^k \times B^m$$

(c) 
$$D_{k+m} = (D_k)^{B^m}$$

These lemmas, presented in different formulations, are commonly used in the literature for computing Dedekind numbers. Wiedemann [27] used the isomorphism  $D_8 = (D_6)^{B^2}$  to compute  $d_8$ , and Jäkel [11] used the isomorphism  $D_9 = (D_5)^{B^4}$  to compute  $d_9$ .

#### 2.2 Permutations acting on a Boolean function

Let  $S_n$  denote the set of all permutations of  $A_n = \{1, 2, ..., n\}$ . Each  $\pi \in S_n$  acts on:

- $A_n$  by directly permuting its elements,
- $B^n$  as follows: for  $x \in B^n$ ,  $\pi(x) = x \circ \pi^{-1}$ ,
- $D_n$  as follows: for  $f \in D_n$ ,  $\pi(f) = f \circ \pi$ .

For example, the permutation  $\pi = (12) \in S_2$  generates the permutation (00)(01, 10)(11) in  $B^2$  and the permutation (0000)(0001)(0011, 0101)(0111)(1111) in  $D_2$ .

Each permutation  $\pi \in S_n$  can be represented as a product of disjoint cycles. The cycle type of  $\pi$  is the tuple of lengths of these cycles in increasing order. For example, (12)(345)(6789) has cycle type (2, 3, 4), and its total length is 9.

Given a set X and a permutation  $\pi$  acting on X, the *orbit* of an element  $x \in X$  under the action of  $\pi$  is the set  $\{\pi^k(x) \mid k \in \mathbb{N}\}$ .

An element x is said to be a *fixed point* of a permutation  $\pi$  if  $\pi(x) = x$ . Let  $\Phi_n(\pi)$  denote the set of all fixed points of  $\pi$  acting on  $D_n$ .

We define an equivalence relation  $\sim$  on  $D_n$  as follows:  $f \sim g$  if there exists a permutation  $\pi \in S_n$  such that  $f = \pi(g)$ . For a function  $f \in D_n$ , its equivalence class is  $[f] = \{g \in D_n : g \sim f\}$ . We denote by  $\gamma(f)$  the cardinality of [f].

### 3 Counting inequivalent monotone Boolean functions (Paper A and C)

Let us recall that by  $r_n$  we denote the number of inequivalent monotone Boolean functions of n variables. In Papers A and C, we address the challenge of determining the values of  $r_8$  and  $r_9$ . We follow Liu and Hu, who explored this problem in the 1980s [12, 13] and used Burnside's lemma for calculating the values of  $r_n$  up to n = 7.

By Burnside's lemma, the value of  $r_n$  can be calculated as follows [7, 19, 23]:

$$r_n = \frac{1}{n!} \sum_{i=1}^k \mu_i \cdot \phi(\pi_i),$$
(1.1)

where:

- $-\phi_n(\pi) = |\Phi_n(\pi)|$ , the number of fixed points in  $D_n$  under permutation  $\pi$
- k is the number of different cycle types in  $S_n$ ,
- -i is the index of a cycle type,
- $-\mu_i$  is the number of permutations  $\pi \in S_n$  with cycle type *i*,
- $-\pi_i$  is a representative permutation  $\pi \in S_n$  with cycle type *i*.

Using this technique, the fundamental challenge was to develop effective algorithms for counting fixed points in  $D_n$ . Due to our inability to translate the Chinese papers, we could not use the techniques described by Liu and Hu [12, 13] and consequently had to develop our techniques from scratch.

As the basis for our algorithms to generate or count fixed points in  $D_n$ , we use the following lemmas:

Lemma 1.3 (Paper C, Lemma 6).  $\Phi_n(\pi) = B^{B^n(\pi)}$ ,

where  $B^n(\pi)$  is the poset of orbits of  $B^n$  under  $\pi \in S_n$ , where two orbits  $C_1$  and  $C_2$  are in relation  $C_1 \leq C_2$  if there exists a  $c_1 \in C_1$  and  $c_2 \in C_2$  such that  $c_1 \leq c_2$ .

For example, the permutation  $\pi = (123) \in S_3$  acting on  $B^3$  has four orbits:

$$\{\{000\}, \{001, 010, 100\}, \{011, 101, 110\}, \{111\}\}.$$

They form the chain  $P_4$ .

The essence of the algorithm based on this lemma is to count  $\phi_n(\pi)$  by generating all downsets of  $B_n(\pi)$  recursively – the set of such downsets is equivalent to  $B^{B^n(\pi)}$ . For a pseudocode refer to (Paper A, Algorithm 1). A key limitation of this approach is its high memory requirement, as it stores all downsets rather than just counting them. Our calculations, performed on a machine with 128 GB of main memory, reach up to  $\phi_9((123)(456789)) = 218542866$ . This value represents the upper limit for our implementation and hardware.

**Theorem 1.4** ([23], Theorem 4). Consider a partition of the antichain  $A_n = \{1, ..., n\}$  into two disjoint antichains  $A_k = \{1, ..., k\}$  and  $A_m = \{k + 1, ..., n\}$ , where n = k + m; and two permutations: one  $\pi$  acting on  $A_k$  and  $\rho$  acting on  $A_m$ . Suppose that each cycle of  $\pi$ , when acting on  $B^k$ , has the length which is coprime with the length of every cycle of  $\rho$ , when acting on  $B^m$ . Then,

$$\Phi_n(\pi \circ \rho) = \Phi_k(\pi)^{B^m(\rho)}.$$

A detailed pseudocode of the algorithm based on a special case of this theorem, where m = 1 and  $\rho$  is the identity permutation, can be found in (Paper A, Algorithm 2). Carić and Živković elaborate on this algorithm for situation where m = 2 [7, Theorem 3.1]. In Paper C, we extend this special case to address cases up to m = 4.

In determining  $r_8$ , there was a singular instance where the previously mentioned algorithms fail to work due to constraints in computational resources, specifically for  $\phi_8((12)(34)(56)(78))$ . In the computation of  $r_9$ , however, there were three scenarios where challenges arose:  $\phi_9((12)(34)(56)(78))$ ,  $\phi_9((12)(34)(56)(789))$ , and  $\phi_9((123)(456)(789))$ . We describe solutions for these cases in (Paper C, Section 3C-E). Notably, for calculating  $\phi_9((12)(34)(56)(789))$ , we use the implementation of Theorem 1.4, with  $\pi = (12)(34)(56)$ and  $\rho = (789)$ .

With  $d_8$  and  $d_9$  precomputed, our calculation of  $r_8$  took approximately a few minutes, and the calculation of  $r_9$  took approximately 10 days. This represents a significant improvement compared to the technique used by Stephen and Yusun in 2014 [24]. The source code for the implementations of the algorithms we use to calculate  $r_8$  and  $r_9$ , is available in the GitHub repository [21]. Our result concerning the value of  $r_8$  was confirmed by Carić and Živković [7] approximately one year after the publication of the preprint containing our result. The value of  $r_9$  has not yet been confirmed by any other research team.

#### **4** Divisibility properties of Dedekind numbers (Paper B)

In 1990, Wiedemann calculated  $d_8$  [27]. His result was confirmed in 2001 by Fidytek, Mostowski, Somla, and Szepietowski [10]. The impulse for writing Paper B came from the letter from Wiedemann to Sloane [28] informing about the computation of  $d_8$ , specifically this fragment: "Unfortunately, I don't see how to test it...". Wiedemann only knew that  $d_8$  is even. As far as we know, the only study concerning divisibility of Dedekind numbers is Yamamoto's paper [29], where he shows that if n is even, then  $d_n$  is also even; he also states (without proof) that  $d_9$  is even and  $d_{11}$  is odd.

In Paper B, we observe that  $d_n \equiv \lambda_n \pmod{2}$ . The value of  $\lambda_9$  was reported in 2013 [3] and is even. Hence,  $d_9 \equiv 0 \pmod{2}$ . We confirm the value of  $\lambda_9$  in Paper D (the calculation took about 76 seconds on a 32-thread Xeon machine).

Let us recall that  $P_n$  denotes the chain, the poset with *n* elements where every pair of elements is comparable. To check divisibility of Dedekind numbers by 3, we use the fact that  $D_{n+3}$  is isomorphic to  $D_n^{B^3}$ . Moreover, there are specific symmetries in  $B^3$  which allow us to show that:

$$d_{n+3} \equiv |D_n^{P_4}| \pmod{3}$$

Since  $|D_n^{P_4}| = \phi_n((123))$ , we use methods from Papers A and C to calculate  $|D_6^{P_4}| = 868329572680304346696$ , which is divisible by 3. Hence,  $d_9$  is divisible by 3.

To find the remainders of  $d_9$  when divided by 5 and by 7, we use the following congruence.

Theorem 1.5 (Paper B, Theorem 7).

$$d_{n+2} \equiv \sum_{x \in R_n \cap E_{n,m}^c} \sum_{y \in E_{n,m}^c} \gamma(x) \cdot G(x,y) \pmod{m}.$$

where:

- $\gamma(x) = \#\{z \in D_n : x \sim z\},$
- $E_{n,m} = \{x \in D_n : \gamma(x) \equiv 0 \pmod{m}\},\$
- $E_{n,m}^c = D_n E_{n,m}$ ,

•  $G(x,y) = \#[(x \lor y),\top] \cdot \#[\bot, (x \land y)].$ 

Then, using the Java implementation of Theorem 1.5 on a 32-thread Xeon computer, we calculate:

$$\sum_{x \in R_7 \cap E_{7,5}^c} \sum_{y \in E_{7,5}^c} \gamma(x) \cdot G(x,y) = 1404812111893131438640857806,$$

and

$$\sum_{x \in R_7 \cap E_{7,7}^c} \sum_{y \in E_{7,7}^c} \gamma(x) \cdot G(x,y) = 2998951776450668253756264$$

Hence, we have  $d_9 \mod 5 = 1$  and  $d_9 \mod 7 = 6$ . In summary, we have:

 $d_9 \equiv 0 \pmod{2},$   $d_9 \equiv 0 \pmod{3},$   $d_9 \equiv 1 \pmod{5},$  $d_9 \equiv 6 \pmod{7}.$ 

The Chinese remainder theorem simplifies these to:

$$d_9 \equiv 6 \pmod{210}$$

The source code implementing the algorithms for calculating the divisibility properties of Dedekind numbers, including the Java implementation of Theorem 1.5 and the procedures for determining the remainders modulo 2, 3, 5, and 7, is available alongside the paper [22].

### **5** Counting self-dual monotone Boolean functions (Paper D)

In Paper D, we present three novel algorithms for computing  $\lambda_n$  values and we verify the previously reported  $\lambda_9$  value. These algorithms are based on the following results:

**Theorem 1.6** (Paper D, Theorem 10). For every  $n \ge 0$ , we have:

$$\lambda_{n+2} = \sum_{b \in R_n} \gamma(b) \cdot \#[(b \lor b^*), \top].$$

**Theorem 1.7** (Paper D, Theorem 13). For every  $n \ge 0$ , we have:

$$\lambda_{n+4} = \sum_{\substack{a,b,c\in D_n\\h\geq (a\vee b\vee c\vee a^*\vee b^*\vee c^*)}} \#[a\vee b\vee c,h]\cdot \#[a\vee b^*\vee c^*,h]\cdot \#[b\vee a^*\vee c^*,h]\cdot \#[c\vee a^*\vee b^*,h].$$

Using those two theorems as a basis, we prepare and implement algorithms for calculating  $\lambda_n$  (for pseudocode refer to Paper D, Section 6). We obtain:

 $\lambda_9 = 423295099074735261880,$ 

which confirms the result of Brouwer et al. [3].

In addition, we calculate the number of inequivalent self-dual monotone Boolean functions,  $q_8 = 6001501$ . We derive the result using Burnside's lemma and techniques similar to those we describe in Papers A and C.

### 6 Concluding remarks and future work

In this dissertation, we present algorithms and theorems relevant to the enumeration of various classes of monotone Boolean functions. We use these algorithms and theorems to calculate the numbers  $r_8$ ,  $r_9$ , and  $q_8$ . Furthermore, we confirm the value of  $\lambda_9$ , and prove that  $d_9 \equiv 6 \pmod{210}$ .

Our capabilities reach a limit in counting inequivalent monotone Boolean functions using methods we know. The value of  $r_{10}$  cannot be determined using our methods until the value of  $d_{10}$  is established. Following the trend in determining subsequent Dedekind numbers (see Table 1.1), we estimate that it may take approximately 20-30 years for the value of  $d_{10}$  to be reported.

Regarding the divisibility of Dedekind numbers, our further research could focus on developing more efficient procedures. However, it appears that at this moment the upper limit for a divisor in this context is n!. We think that surpassing this boundary, if feasible, could lead to significant advancements and the development of superior algorithms for calculating Dedekind numbers.

The most promising direction for future work seems to be the determination of the value of  $\lambda_{10}$ . We estimate that the complexity of this task is analogous to that of determining the value of  $d_9$ , which is achievable with current capabilities, as  $d_9$  has recently been computed by two independent research teams. The algorithms we describe in Paper D should be suitable for this purpose after appropriate modifications.

### **Bibliography**

- [1] M. Aigner. Combinatorial Theory. Springer, 1979.
- [2] J. C. Bioch and T. Ibaraki. Generating and approximating nondominated coteries in *IEEE Trans. Parallel Distrib. Syst.*, 6 (1995)
- [3] A. E. Brouwer, C. F. Mills, W. H. Mills, and A. Verbeek. Counting families of mutually intersecting sets in *Electron. J. Comb.*, 2013, Volume 20, Issue 2
- [4] F. a Campo, Relations between powers of Dedekind numbers and exponential sums related to them, J. Integer Sequences 21 (2018), Article 18.4.4.
- [5] R. Church. Numerical analysis of certain free distributive structures in *Duke Math. J.*, 1940, Volume 6
- [6] R. Church. Enumeration by rank of the free distributive lattice with 7 generators in *Notices Amer. Math. Soc.*, 1965, Volume 11
- [7] M. Carić, M. Živković, "The number of nonequivalent monotone Boolean functions of 8 variables," *IEEE Trans. Inf. Theory*, vol. 69, no. 6, pp. 4027-4034 (2023).
- [8] A. E. Brouwer and A. Verbeek. Counting families of mutually intersecting sets, Report ZN 41, March 1972, Math. Centr., Amsterdam.
- [9] R. Dedekind, Über Zerlegungen von Zahlen durch ihre größten gemeinsamen Teiler, *Festschrift Hoch. Braunschweig u. ges. Werke II* (1897) 103–148.
- [10] R. Fidytek, A. W. Mostowski, R. Somla, A. Szepietowski. Algorithms counting monotone Boolean functions in *Information Processing Letters*, 2001, issue 79, pages 203–209
- [11] C. Jäkel, A computation of the ninth Dedekind number, *Journal of Computational Algebra* 6-7 (2023)
- [12] 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.

- [13] 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.
- [14] D. E. Loeb, A. R. Conway. Voting Fairly: Transitive Maximal Intersecting Families of Sets, *Journal of Combinatorial Theory, Series A*, 91 (2000)
- [15] C. F. Mills and W. M. Mills. The calculation of  $\lambda(8)$ , preprint (1979).
- [16] L. Van Hirtum, P. De Causmaecker, J. Goemaere, T. Kenter, H. Riebler, M. Lass, and C. Plessl, A computation of D(9) using FPGA supercomputing, preprint, 2023. Available at https://arxiv.org/abs/2304.03039.
- [17] N. M. Riviere, Recursive formulas on free distributive lattices, J. Comb. Theory 5 (1968).
- [18] N. J. A. Sloane. The Online Encyclopedia of Integer Sequences https://oeis.org
- [19] B. Pawelski. On the number of inequivalent monotone Boolean functions of 8 variables, *J. Integer Sequences* 25 (2022), Article 25.7.7.
- [20] B. Pawelski. On the number of inequivalent monotone Boolean functions of 9 variables, *IEEE Trans. Inf. Theory* (2024)
- [21] B. Pawelski, Algorithms counting fixed points in sets of monotone Boolean functions, source code, 2023. [Online]. Available: https://github.com/bpawelski/r9.
- [22] B. Pawelski Szepietowski, Divisibility properties of and A. Dedekind numbers, code. 2023. [Online]. Available: source https://cs.uwaterloo.ca/journals/JIS/VOL26/Pawelski/pawelski\_source\_code.zip.
- [23] A. Szepietowski, Fixes of permutations acting on monotone Boolean functions, J. Integer Sequences 25 (2022), Article 25.9.6.
- [24] T. Stephen and T. Yusun, Counting inequivalent monotone Boolean functions, *Discrete Appl. Math.*, **167** (2014), 15–24.
- [25] M. Timotijević. Note on combinatorial structure of self-dual simplicial complexes, Matematički Vesnik, 71 (2019)
- [26] M. Ward, Note on the order of the free distributive lattice, *Bull. Amer. Math. Soc.* **52** (1946), 423.

- [27] D. Wiedemann, A computation of the eighth Dedekind number, Order 8 (1991), 5–6.
- [28] D. Wiedemann. Letter to N. J. A. Sloane (1990). Available at https://oeis.org/ A000372/a000372.pdf.
- [29] K. Yamamoto. Note on the order of free distributive lattices in *Sci. Rep. Kanazawa Univ.*, II (1953), 5–6.



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 <u>A000372</u> 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]:

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|.$$
 (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  $\pi$  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  $r_n$ :

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

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

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

where

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

The formula for determining  $\mu$  for each cycle type is as follows:

$$\mu_i = \frac{n!}{(l_1^{k_1} \cdot l_2^{k_2} \cdots l_r^{k_r})(k_1! \cdot k_2! \cdots k_r!)}$$
(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  $\mu$  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  $\emptyset, x_1, x_2, x_1x_2, x_3, \ldots, x_1x_2x_3 \cdots x_n$  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:

Ø	$x_1$	$x_2$	$x_1 x_2$	$x_3$	$x_1x_3$	$x_{2}x_{3}$	$x_1 x_2 x_3$
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)	Ø	$x_1$	$x_2$	$x_1 x_2$	$x_3$	$x_1 x_3$	$x_2 x_3$	$x_1x_2x_3$
$(1\ 2\ 3)$	Ø	$x_3$	$x_1$	$x_1 x_3$	$x_2$	$x_2 x_3$	$x_1 x_2$	$x_1x_2x_3$

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:

	<i>n</i> -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  $\pi$ :

- All points in the same orbit of  $\pi'$  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  $\pi$  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  $S = \phi_n(\pi)$ 1: Determine representative  $\pi \in S_n$  of cycle type i2: Lift  $\pi$  into  $\pi' \in S_{B^n}$ 3: Generate set  $\operatorname{Orb}_i$  containing all orbits in  $\pi'$ 4: Order  $Orb_i$  into poset P by set inclusion 5: Initialize set S of downsets of P6: Add two downsets:  $\{\}$  and  $\{0\}$  to S7: for all elements  $a \in P$  do for all elements  $b \in S$  do 8: if  $(b \cup a)$  is downset of P then 9: Add downset  $(b \cup a)$  to S 10: end if 11: end for 12:

13: 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  $\omega$  in  $D_{n+1}$  can be split into two functions  $(\alpha, \beta)$  from  $D_n$ . Moreover, there is a relation  $\alpha \leq \beta$ , 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  $\omega$  in  $\phi_{n+1}(\pi)$  can be split into two functions  $(\alpha, \beta)$ .

Constructing  $\omega$  from  $\alpha$  is simply adding new variable  $(x_{n+1})$  to  $\alpha$ .  $\beta$  contains data about each possible intersection of  $\alpha$  with  $(x_{n+1})$ . Hence,  $\alpha$  clearly belongs to  $\phi_n(\pi)$ —same as  $\beta$ ,

as its variables are regrouped in the same way. Only difference between them is additional variable  $(x_{n+1})$  which is fixed point of  $\pi$ , added to each element in  $\beta$ .

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

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.

<b>lgorithm 2</b> Generating $\phi_{n+1}(\pi)$ under permutation $\pi$ of cycle type of total length $n$	
<b>Input:</b> Cycle type $i$ of total length $n$	
<b>Output:</b> Set $S = \phi_{n+1}(\pi)$	
1: Use Algorithm 1 to generate $S' = \phi_n(\pi)$	
2: Convert all elements in $S'$ to integers of length $2^n$ bits	
3: Initialize set S of integers of length $2^{n+1}$ bits	
4: for all elements $a \in S'$ do	
for all elements $b \in S'$ do	
5: <b>if</b> $(a \mid b) = b$ <b>then</b> $\triangleright$ " " is bitwise "C	)R"
7: Add integer $((a \ll 2^n) \mid b)$ to $S \qquad \qquad \triangleright "<<"$ is logical s	$\operatorname{hift}$
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:  $\alpha_w \leq \beta_w \leq \delta_w, \alpha_w \leq \gamma_w \leq \delta_w$ .

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  $\tau$  denote such the permutation. In other words,  $\tau = \tau_1 \cdots \tau_x$ , and  $\tau_x$  is 2-cycle: (n+1, n+2). Let  $\sigma$  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:

- $\alpha$  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  $\gamma = \beta((1\ 2))$
- $\delta$  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:

<b>lgorithm 3</b> Determining $ \phi_{n+2}(\tau) $
<b>Input:</b> $D_n$ and $\phi_n(\sigma)$
Output: $ \phi_{n+2}(\tau) $
1: Initialize $k = 0$ ,
2: for all $\beta \in D_n$ do
B: Determine $\gamma = \beta((1\ 2))$
4: Initialize $down = 0, up = 0$
5: for all $\alpha \in \phi_n(\sigma)$ do
5: <b>if</b> $(\alpha \preceq (\beta \mid \gamma))$ <b>then</b> $\triangleright$ " " is bitwise "OR"
down = down + 1
end if
end for
b: for all $\delta \in \phi_n(\sigma)$ do
1: <b>if</b> $((\beta \& \gamma) \preceq \delta)$ <b>then</b> $\triangleright$ "&" is bitwise "AND"
2:   up = up + 1
B: end if
4: end for
5: $k = k + up \cdot down$
3: 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  $\pi$  has at least one disjoint 2-cycle. Performing calculations using a similar approach should speed-up counting, but the relation between  $\beta$  and  $\gamma$  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  $\mu_{11}$  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	$\pi_i$	$\mu_i$	$\phi_7(\pi_i)$
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
	$1 - \frac{k-10}{2}$		

$$r_7 = \frac{1}{5040} \sum_{i=1}^{4} \mu_i \phi_7(\pi_i) = 490013148$$

Table 6: Detailed calculation results for  $r_7$ .

i	$\pi_i$	$\mu_i$	$\phi_8(\pi_i)$
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 <u>A000041</u>, <u>A000372</u>, <u>A003182</u>, and <u>A181897</u>.)

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.



Journal of Integer Sequences, Vol. 26 (2023), Article 23.7.7

### **Divisibility Properties of Dedekind Numbers**

Bartłomiej Pawelski and Andrzej Szepietowski Institute of Informatics University of Gdańsk Poland bartlomiej.pawelski@ug.edu.pl andrzej.szepietowski@ug.edu.pl

#### Abstract

We study some divisibility properties of Dedekind numbers. We show that the ninth Dedekind number is congruent to 6 modulo 210.

#### 1 Introduction

We define  $D_n$  to be the set of all monotone Boolean functions of n variables. The cardinality of this set,  $d_n$ , is known as the *n*-th Dedekind number. Values of  $d_n$  are described by the OEIS (On-Line Encyclopedia of Integer Sequences) sequence A000372 (see Table 1).

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  $d_n$  (A000372) and  $r_n$  (A003182).

In 1990, Wiedemann calculated  $d_8$  [11]. His result was confirmed in 2001 by Fidytek, Mostowski, Somla, and Szepietowski [4]. The impulse for writing our paper came from the letter from Wiedemann to Sloane [12] informing about the computation of the eighth Dedekind number, specifically this fragment: "Unfortunately, I don't see how to test it...". Wiedemann only knew that  $d_8$  is even. Despite its obvious importance, there is a lack of studies on the divisibility of Dedekind numbers. As far as we know, the only paper concerning this question is Yamamoto's paper [13], where he shows that if n is even, then  $d_n$  is also even; he also states (without proof) that  $d_9$  is even and  $d_{11}$  is odd.

Our research aims to fill this lack by proposing new methods to determine the divisibility of Dedekind numbers. As an application of these methods, we compute remainders of  $d_9$ divided by one-digit prime numbers, which (we hope) will help to verify the value  $d_9$  after its first computation.

Our main result is the following system of congruences:

$$d_9 \equiv 0 \pmod{2},$$
  

$$d_9 \equiv 0 \pmod{3},$$
  

$$d_9 \equiv 1 \pmod{5},$$
  

$$d_9 \equiv 6 \pmod{7}.$$

By the Chinese remainder theorem, we have

$$d_9 \equiv 6 \pmod{210}.$$

Recently, after the preprint of this paper was published on ArXiv, two independent research teams [5, 7] reported the same value:

 $d_9 = 286386577668298411128469151667598498812366,$ 

which confirms our results.

#### 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* (namely:  $0 \le 0, 0 \le 1$ ,  $1 \le 1$ ) and the following partial order in  $B^n$ . For any two elements,  $x = (x_1, \ldots, x_n)$ ,  $y = (y_1, \ldots, y_n)$  in  $B^n$ ,

 $x \leq y$  if and only if  $x_i \leq y_i$  for all  $1 \leq i \leq n$ .

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

$$x \le y \Longrightarrow h(x) \le h(y)$$
Let  $D_n$  denote the set of monotone functions with n variables and let  $d_n$  denote  $|D_n|$ . We have the partial order in  $D_n$  defined by:

$$g \leq h$$
 if and only if  $g(x) \leq h(x)$  for all  $x \in B^n$ .

We shall represent the elements of  $D_n$  as strings of bits of length  $2^n$ . Two elements of  $D_0$ will be represented as 0 and 1. Any element  $g \in D_1$  can be represented as the concatenation g(0) \* g(1), where  $g(0), g(1) \in D_0$  and  $g(0) \leq g(1)$ . Hence  $D_1 = \{00, 01, 11\}$ . Each element of  $g \in D_2$  is the concatenation (string) of four bits: g(00) \* g(10) \* g(01) \* g(11) which can be represented as a concatenation  $g_0 * g_1$ , where  $g_0, g_1 \in D_1$  and  $g_0 \leq g_1$ . Hence  $D_2 =$  $\{0000, 0001, 0011, 0101, 0111, 1111\}$ . Similarly, any element of  $g \in D_n$  can be represented as a concatenation  $g_0 * g_1$ , where  $g_0, g_1 \in D_{n-1}$  and  $g_0 \leq g_1$ . Therefore, we can treat functions in  $D_n$  as sequences of bits and as integers. We let  $\leq$  denote the total order in  $D_n$  induced by the total order in integers.

For a set  $Y \subseteq D_n$ , by  $Y^2$  we denote the Cartesian power  $Y^2 = Y \times Y$ , that is the set of all ordered pairs (x, y) with  $x, y \in Y$ . Similarly for more than two factors, we write  $Y^k$  for the set of ordered k-tuples of elements of Y. We let  $\top$  denote the maximal element in  $D_n$ , that is,  $\top = (1 \dots 1)$ ; and  $\bot$  denote the minimal element in  $D_n$ , that is,  $\bot = (0 \dots 0)$ . For two elements  $x, y \in D_n$ , we let x|y denote the bitwise or; and x & y denote the bitwise and. Furthermore, we let  $\operatorname{re}(x, y)$  denote  $|\{z \in D_n : x \leq z \leq y\}|$ . Note that  $\operatorname{re}(x, \top) = |\{z \in D_n : x \leq z\}|$  and  $\operatorname{re}(\bot, y) = |\{z \in D_n : z \leq y\}|$ .

### 2.1 Posets

A poset (partially ordered set)  $(S, \leq)$  consists of a set S together with a binary relation (partial order)  $\leq$  which is reflexive, transitive, and antisymmetric. Given two posets  $(S, \leq)$ and  $(T, \leq)$  a function  $f: S \to T$  is monotone, if  $x \leq y$  implies  $f(x) \leq f(y)$ . By  $T^S$  we denote the poset of all monotone functions from S to T with the partial order defined by

 $f \leq g$  if and only if  $f(x) \leq g(x)$  for all  $x \in S$ .

In this paper we use the following well-known lemma; see [3, 10]:

**Lemma 1.** The poset  $D_{n+k}$  is isomorphic to the poset  $D_n^{B^k}$ —the poset of monotone functions from  $B^k$  to  $D_n$ .

### 3 Divisibility of Dedekind numbers by 2

In 1952, Yamamoto [13] proved that if n is even, then  $d_n$  is also even; he also stated (without proof) that  $d_9$  is even and  $d_{11}$  is odd. In order to prove that  $d_9$  is even, we will leverage the duality property of Boolean functions. For each  $x \in D_n$ , we have dual  $x^d$  which is obtained by reversing and negating all bits. For example,  $1111^d = 0000$  and  $0001^d = 0111$ . An element  $x \in D_n$  is self-dual if  $x = x^d$ . For example, 0101 and 0011 are self-duals in  $D_2$ . If x is not

self-dual, and  $y = x^d \neq x$ , then  $y^d = x$ . Thus, non-self-duals form pairs of the form  $(x, x^d)$ , where  $x \neq x^d$ . Let  $k_n$  denote the number of these pairs and let  $\lambda_n$  denote the number of self-dual functions in  $D_n$ . We have that  $d_n = 2k_n + \lambda_n$ . Hence  $\lambda_n \equiv d_n \pmod{2}$ . Values of  $\lambda_n$  are described by the OEIS sequence A001206; see Table 2. The last known term of this sequence,  $\lambda_9$ , was calculated in 2013 by Brouwer et al. [2].

n	$\lambda_n$
0	0
1	1
2	2
3	4
4	12
5	81
6	2 646
7	1.422.564
8	229 809 982 112
9	423,295,099,074,735,261,880

Table 2: Known values of  $\lambda_n$  (A001206).

**Corollary 2.** We have  $d_9 \equiv \lambda_9 \equiv 0 \pmod{2}$ .

One can directly check that  $d_n \equiv \lambda_n \pmod{2}$  for  $n \leq 8$ .

## 4 Divisibility of Dedekind numbers by 3

By Lemma 1, the poset  $D_{n+3}$  is isomorphic to the poset  $D_n^{B^3}$ —the set of monotone functions from  $B^3 = \{000, 001, 010, 100, 110, 101, 011, 111\}$  to  $D_n$ . Now consider the group  $S_3$ —the permutations on  $\{1, 2, 3\}$ . The group  $S_3$  is isomorphic to the automorphism group  $\operatorname{Aut}(B^3)$ of the Boolean lattice  $B^3$ . The automorphism group  $\operatorname{Aut}(B^3)$  acts in a natural way on  $D_n^{B^3}$ by

$$\alpha(f) = f \circ \alpha^{-1}$$

for all  $\alpha \in \operatorname{Aut}(B^3)$  and all  $f \in D_n^{B^3}$ . Let  $\mathcal{O}(f) = \{\alpha(f) \in D_n^{B^3} : \alpha \in \operatorname{Aut}(B^3)\}$  denote the orbit of f under this action and by  $\gamma(f) = |\mathcal{O}(f)|$  its cardinality. The orbits form a partition of  $D_{n+3} = D_n^{B^3}$ . Each of these orbits has one, three, or six elements. Moreover, an orbit  $\mathcal{O}(f)$  has one element if and only if f(001) = f(010) = f(100) and f(011) = f(101) = f(110). Such a function f can be identified with a monotone function from the path  $P_4$  to  $D_n$ . Hence,

$$d_{n+3} \equiv |D_n^{P_4}| \pmod{3}$$

It is well known, see [1, 10], that the number of monotone functions from the path  $P_4 = (a < b < c < d)$  to a poset S is equal to the sum of the elements of the third power

of M(S)—the incidence matrix of S. For example, for the poset  $D_1 = \{00 < 01 < 11\}$ , we have

$$M(D_1) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$
$$M(D_1)^3 = \begin{pmatrix} 1 & 3 & 6 \\ 0 & 1 & 3 \end{pmatrix}$$

and

$$M(D_1)^3 = \left(\begin{array}{rrrr} 1 & 3 & 6\\ 0 & 1 & 3\\ 0 & 0 & 1 \end{array}\right).$$

The sum of the elements of  $(M(D_1)^3)$  is equal to 15, which is equal to  $|D_1^{P_4}|$ —the number of monotone functions from  $P_4$  to  $D_1$ .

Furthermore, consider  $D_2 = \{0000, 0001, 0011, 0101, 0111, 1111\}$  and its incidence matrix:

$$M(D_2) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Now consider the third power of the incidence matrix of  $D_2$ :

$$M(D_2)^3 = \begin{pmatrix} 1 & 3 & 6 & 6 & 14 & 20\\ 0 & 1 & 3 & 3 & 9 & 14\\ \hline 0 & 0 & 1 & 0 & 3 & 6\\ 0 & 0 & 0 & 1 & 3 & 6\\ \hline 0 & 0 & 0 & 0 & 1 & 3\\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The sum of the elements of  $(M(D_2)^3)$  is equal to 105, which is equal to  $|D_2^{P_4}|$ —the number of monotone functions from  $P_4$  to  $D_2$ . In a similar we can compute  $|D_n^{P_4}|$  for n = 3, 4, 5. Unfortunately, this method cannot be easily applied for n = 6, because  $M(D_6)$  is too large. However, Pawelski [8] proposed another method:  $|D_{(n+m)}^{P_4}| = |D_n^{P_4 \times B^m}| = |(D_n^{P_4})^{B^m}|$  (also see [10]). Using the same program as used in [8] to compute  $|D_5^{P_4}|$  we can calculate  $|D_6^{P_4}|$  and the result (see Table 3) is divisible by 3.

**Corollary 3.** As  $|D_6^{P_4}| = 868329572680304346696$  is divisible by 3, the quantity  $d_9$  is also divisible by 3.

One can directly check that  $d_{n+3} \equiv |D_n^{P_4}| \pmod{3}$  for  $n \leq 5$ .

n	$ D_n^{P_4} $	$ D_n^{P_4}  \mod 3$
0	5	2
1	15	0
2	105	0
3	3,490	1
4	2,068,224	0
5	262,808,891,710	1
6	868,329,572,680,304,346,696	0

Table 3: Known values of  $|D_n^{P_4}|$ . Note that  $d_{n+3} \equiv |D_n^{P_4}| \pmod{3}$ .

## 5 Main lemma

In the sequel we shall use another definition of a group acting on  $D_n^{B^k}$ . In order to do this it is convenient to identify the lattice  $D_n = B^{B^n}$  with the isomorphic up-set lattice  $\mathcal{U}_n$  of  $B^n$ . An isomorphism is given by  $h: D_n \to \mathcal{U}_n$ , where  $h(x) = x^{-1}(1)$  for all  $x \in D_n$ . For  $\beta \in \operatorname{Aut}(B^n)$  and  $x \in D_n$ , we have

$$h(\beta(x)) = \beta(h(x)).$$

Let P and Q be two posets and let  $P^Q$  be the poset of monotone functions from Q to P. Now, we can define an action of  $\operatorname{Aut}(P)$  on  $P^Q$  by setting

$$\beta(f) = \beta \circ f$$

for all  $\beta \in \operatorname{Aut}(P)$  and  $f \in P^Q$ . We let

$$\mathcal{O}(f) = \{\beta(f) : \beta \in \operatorname{Aut}(P)\}\$$

denote the orbit of f under this action. Additionally, for  $p \in P$ , we write

$$[p] = \{\beta(p) : \beta \in \operatorname{Aut}(P)\}\$$

for the orbit of p under the natural action of Aut(P) on P.

We use  $\sim$  to denote an equivalence relation on  $D_n$ . Namely, two functions  $p, r \in D_n$ are equivalent,  $p \sim r$ , if there is an automorphism  $\alpha \in \operatorname{Aut}(D_n)$  such that  $p = \alpha(r)$ . For a function  $p \in D_n$  its equivalence class is the set  $[p] = \{r \in D_n : r \sim p\}$ . We let  $\gamma(f)$ denote |[f]|. For m > 1, let  $E_{n,m} = \{p \in D_n : \gamma(f) \equiv 0 \pmod{m}\}$  and  $E_{n,m}^c = D_n - E_{n,m}$ . For the class [p], we define its canonical representative as the one element in [p] chosen to represent the class. One of the possible approaches is to choose its minimal (according to the total order  $\preceq$ ) element [11]. Sometimes we shall identify the class [p] with its canonical representative and treat [p] as an element in  $D_n$ . We let  $R_n$  denote the set of equivalence classes and  $r_n$  the number of equivalence classes; that is,  $r_n = |R_n|$ . Values of  $r_n$  are described by <u>A003182</u> OEIS sequence; see Table 1. **Lemma 4.** For every  $q \in Q$  and every  $f \in P^Q$ , the integer |[f(q)]| divides  $|\mathcal{O}(f)|$ .

*Proof.* For every  $p \in [f(q)]$  we define

$$\mathcal{G}(p) := \{ g \in \mathcal{O}(f) : g(q) = p \}.$$

The sets  $\mathcal{G}(p), p \in [f(q)]$  form a partition of  $\mathcal{O}(f)$  and the sets  $\mathcal{G}(p)$  have the same cardinality. Indeed, for every  $\beta \in \operatorname{Aut}(P)$ ,

$$g \in \mathcal{G}(f(q)) \iff \beta(g) \in \mathcal{G}(\beta(f(q))).$$

**Lemma 5.** For arbitrary subset  $W \subseteq Q$ , the cardinality of  $P^Q$  is congruent modulo m to the cardinality of

$$\{f \in P^Q : f(W) \subseteq E_{n,m}^c\}.$$

*Proof.* For each automorphism  $\alpha \in Aut(P)$  and for each  $p \in P$ , we have  $[p] = [\alpha(p)]$ . Hence,

$$p \in E_{n,m}^c \quad \iff \quad \alpha(p) \in E_{n,m}^c$$

and for every function  $f \in P^Q$ , we have

$$f(W) \subseteq E_{n,m}^c \iff \alpha \circ f(W) \subset E_{n,m}^c$$

Orbits  $\mathcal{O}(f)$  form a partition of  $P^Q$ . If  $f \sim g$  (or in other words if  $f \in \mathcal{O}(g)$ ), then there exists automorphism  $\alpha \in \operatorname{Aut}(P)$ , such that  $g = \alpha \circ f$  and

$$f(W) \subseteq E_{n,m}^c \iff g(W) \subseteq E_{n,m}^c.$$

So we have two kinds of orbits:

- orbits  $\mathcal{O}(f)$ , where  $g(W) \subseteq E_{n,m}^c$  for all  $g \in \mathcal{O}(f)$ ,
- orbits  $\mathcal{O}(f)$ , where  $g(W) \not\subseteq E_{n,m}^c$  for all  $g \in \mathcal{O}(f)$ .

Moreover, if  $f(W) \not\subseteq E_{n,m}^c$ , then there exists  $w \in W$  such that  $f(w) \in E_{n,m}$ , hence, m divides |[f(w)]| and by Lemma 4, m divides  $|\mathcal{O}(f)|$ .

# 6 Counting functions from $B^2$ to $D_n$

By Lemma 1, the poset  $D_{n+2}$  is isomorphic to the poset  $D_n^{B^2}$ —the poset of monotone functions from  $B^2 = \{00, 01, 10, 11\}$  to  $D_n$ . Consider the function G that, for every pair  $(x, y) \in D_n^2$ , takes the value

$$G(x, y) = \operatorname{re}(x|y, \top) \cdot \operatorname{re}(\bot, x\& y).$$

Observe that G(x, y) is equal to the number of functions  $f \in D_n^{B^2}$  with f(01) = x and f(10) = y. Function G is well-known, as it is discussed in [3, 4, 11].

For  $A \subseteq D_n \times D_n$  let G(A) denote  $\sum_{(x,y)\in A} G(x,y)$ . By Lemma 1, we have

$$d_{n+2} = G(D_n \times D_n) = \sum_{x \in D_n} \sum_{y \in D_n} G(x, y).$$

Consider the set  $W_2 = \{01, 10\}$ . By Lemma 5, we have

$$d_{n+2} \equiv G(D_n \times D_n) \equiv G(E_{n,m}^c \times E_{n,m}^c) \pmod{m}.$$

Observe that, for every automorphism  $\pi \in \operatorname{Aut}(B^n)$  and every  $x, y \in D_n$ , we have  $G(x, y) = G(\pi(x), \pi(y))$ .

**Lemma 6.** Let Y be a subset  $Y \subseteq D_n$  and suppose that  $\pi(Y) = Y$  for every automorphism  $\pi \in \operatorname{Aut}(B^n; \text{ and let } x \text{ and } y \text{ be two equivalent, } x \sim y, \text{ elements in } D_n$ . Then

- 1.  $G(\{x\} \times Y) = G(\{y\} \times Y)$ .
- 2.  $G([x] \times Y) = \gamma(x) \cdot G(\{x\} \times Y).$

*Proof.* Notice that condition  $\pi(Y) = Y$  implies that  $\pi$  is a bijection on Y, or in other words,  $\pi$  permutes the elements of Y.

For (1), observe that

$$G(\{x\} \times Y) = \sum_{s \in Y} G(x, s) = \sum_{s \in Y} G(\pi(x), \pi(s))$$
  
= 
$$\sum_{t \in \pi(Y)} G(\pi(x), t) = \sum_{t \in Y} G(\pi(x), t) = G(\{\pi(x)\} \times Y).$$

We use the fact that  $\pi(Y) = Y$ .

Observe that for every automorphism  $\pi \in \operatorname{Aut}(b^n)$ , we have and  $\pi(E_{n,m}^c) = E_{n,m}^c$ . Hence, by Lemma 6, we get

#### Theorem 7.

$$d_{n+2} \equiv \sum_{x \in R_n \cap E_{n,m}^c} \sum_{y \in E_{n,m}^c} \gamma(x) \cdot G(x,y) \pmod{m}.$$

Here we identify each class  $[x] \in R_n$  with its canonical representative.

**Example 8.** Consider the poset  $D_2 = \{0000, 0001, 0011, 0101, 0111, 1111\}$ . There are five equivalence classes: namely,  $R_2 = \{\{0000\}, \{0001\}, \{0011, 0101\}, \{0111\}, \{1111\}\}$ . Two elements: 0101 and 0011 are equivalent. For m = 2, we have  $E_{2,2} = \{0011, 0101\}$  and  $E_{2,2}^c = \{0000, 0001, 0111, 1111\}$ . Table 4 presents values of G(x, y) for  $x, y \in D_2$ . Let

 $Y = [0011] = \{0011, 0101\}$ . For every permutation  $\pi \in S_2$ , we have  $\pi(Y) = Y$ . Furthermore,  $G(\{0011\} \times Y) = G(\{0101\} \times Y) = 9 + 4 = 13$ ; and  $G([0011] \times Y) = 2 \cdot 13 = 26$ , which is divisible by 2.

Similarly, for  $Z = [0001] = \{0001\}$ , we have that  $\pi(Z) = Z$  for every permutation  $\pi \in S_2$ . Furthermore,  $G(\{0011\} \times Z) = G(\{0101\} \times Z) = 6$ ; and  $G([0011] \times Z) = 2 \cdot 6 = 12$ , which is divisible by 2. By summing up all values in Table 4 we obtain  $G(D_2 \times D_2) = 168 = d_4$ .

$\begin{bmatrix} y\\x \end{bmatrix}$	0000	0001	0011	0101	0111	1111
0000	6	5	3	3	2	1
0001	5	10	6	6	4	2
0011	3	6	9	4	6	3
0101	3	6	4	9	6	3
0111	2	4	6	6	10	5
1111	1	2	3	3	5	6

Table 4: Values of G(x, y) for  $x, y \in D_2$ .

**Example 9** (Continuation of Example 8). By summing the relevant values listed in Table 4, we obtain  $G(E_{2,2}^c \times E_{2,2}^c) = 6 + 5 + 2 + 1 + 5 + 10 + 4 + 2 + 2 + 4 + 10 + 5 + 1 + 2 + 5 + 6 = 70$ . By Theorem 7, we have  $d_4 \equiv 70 \pmod{2}$ . Indeed,  $d_4 = 168$ , which is even.

# 7 Counting functions from $B^3$ to $D_n$

In the next two sections, we show that similar techniques can be also applied to functions in  $D_n^{B^3}$  and  $D_n^{B^4}$ . Consider the function H which for every triple  $(x, y, z) \in D_n^3$  returns the value

$$H(x, y, z) = \operatorname{re}(\bot, x \& y \& z) \cdot \sum_{s \ge x|y|z} \operatorname{re}(x|y, s) \cdot \operatorname{re}(x|z, s) \cdot \operatorname{re}(y|z, s)$$

Observe that H(x, y, z) is equal to the number of monotone functions  $f \in D_n^{B^3}$  with f(001) = x, f(010) = y and f(100) = z. Thus, we have

$$d_{n+3} = H(D_n^3) = \sum_{x \in D_n} \sum_{y \in D_n} \sum_{z \in D_n} H(x, y, z)$$

Function H is discussed in [3]. Consider the set  $W_3 = \{001, 010, 100\}$ . By Lemma 5, we have

$$d_{n+3} \equiv G(D_n \times D_n \times D_n) \equiv G(E_{n,m}^c \times E_{n,m}^c \times E_{n,m}^c) \pmod{m}.$$

Observe that for every automorphism  $\pi \in \operatorname{Aut}(B^n)$  and every  $x, y, z \in D_n$ , we have  $H(x, y, z) = H(\pi(x), \pi(y), \pi(z))$ .

**Lemma 10.** Let Y and Z be subsets  $Y, Z \subseteq D_n$  and suppose that  $\pi(Y) = Y$  and  $\pi(Z) = Z$ for every automorphism  $\pi \in \operatorname{Aut}(B^n)$ ; and let x and y be two equivalent,  $x \sim y$ , elements in  $D_n$ . Then

- 1.  $H({x} \times Y \times Z) = H({y} \times Y \times Z).$
- 2.  $H([x] \times Y \times Z) = \gamma(x) \cdot H(\{x\} \times Y \times Z).$

Proof. (1)  $H(\{x\} \times Y \times Z) = \sum_{s \in Y} \sum_{t \in Z} H(x, s, t) = \sum_{s \in Y} \sum_{t \in Z} H(\pi(x), \pi(s), \pi(t)) = \sum_{u \in \pi(Y)} \sum_{v \in \pi(Z)} H(\pi(x), u, v) = \sum_{u \in Y} \sum_{v \in Z} H(\pi(x), u, v) = H(\{\pi(x)\} \times Y \times Z).$  We use the fact that  $\pi$  is a bijection on  $Y \times Z$  and permutes the elements of  $Y \times Z$ .

As an immediate corollary, we have the following:

#### Theorem 11.

$$d_{n+3} \equiv \sum_{x \in R_n \cap E_{n,m}^c} \sum_{y \in E_{n,m}^c} \sum_{z \in E_{n,m}^c} \gamma(x) \cdot H(x, y, z) \pmod{m}.$$

Here, again, we identify each class  $[x] \in R_n$  with its canonical representative.

**Example 12.** Consider  $D_4$ . There are 168 elements in  $D_4$  and 30 equivalence classes in  $R_4$ . The distribution of these equivalence classes based on their  $\gamma$  value is presented in Table 5. For instance, there are six equivalence classes [x] with  $\gamma(x) = 1$ , two equivalence classes with  $\gamma(x) = 3$ , and so forth. For m = 2, the set  $E_{4,2}^c$  contains only twelve elements and  $R_4 \cap E_{4,3}^c$  contains eight elements. Similarly, for m = 3, the set  $E_{4,3}^c$  contains 42 elements and  $R_4 \cap E_{4,3}^c$  consists of 15 elements.

**Example 13.** We employed a Java implementation of the Theorem 11. For n = 4 and m = 2, 3, 4, 6, 12 we have

 $d_7 \equiv 2320978352 \pmod{2}$ , and therefore  $d_7 \mod 2 = 0$ ,

 $d_7 \equiv 74128573428 \pmod{3}$ , and therefore  $d_7 \mod 3 = 0$ ,

 $d_7 \equiv 128268820802 \pmod{4}$ , and therefore  $d_7 \mod 4 = 2$ ,

 $d_7 \equiv 89637133284 \pmod{6}$ , and therefore  $d_7 \mod 6 = 0$ ,

 $d_7 \equiv 566167187562 \pmod{12}$ , and therefore  $d_7 \mod 12 = 6$ .

One can check these values directly by dividing  $d_7$  by 2, 3, 4, 6, and 12.

# 8 Counting functions from $B^4$ to $D_n$

By Lemma 1, the poset  $D_{n+4}$  is isomorphic to the poset  $D_n^{B^4}$ —the set of monotone functions from  $B^4$  to  $D_n$ . Consider the function F (also discussed in [3, 4]), which for every six elements  $a, b, c, d, e, f \in D_n$ , counts how many functions  $g \in D_n^{B^4}$  satisfy the following equations: g(0011) = a, g(0101) = b, g(1001) = c, g(0110) = d, g(1010) = e, g(1100) = f.

k	$ \{f \in R_4 : \gamma(f) = k\} $
1	6
3	2
4	9
6	6
12	7

Table 5: Number of  $f \in R_4$  with  $\gamma(f) = k$ .

For  $A \subseteq (D_n)^6$  let F(A) denote  $\sum_{(a,b,c,d,e,f)\in A} F(a,b,c,d,e,f)$ . By Lemma 1, we have

$$d_{n+4} = F(D_n^6) = \sum_{a \in D_n} \sum_{b \in D_n} \sum_{c \in D_n} \sum_{d \in D_n} \sum_{e \in D_n} \sum_{f \in D_n} F(a, b, c, d, e, f).$$

Consider the set  $W_4 = \{0011, 0101, 1001, 0110, 1010, 1100\}$ . By Lemma 5, we have

$$d_{n+4} \equiv F(D_n^6) \equiv F((E_{n,m}^c)^6) \pmod{m}$$

Observe that for every automorphism  $\pi \in \operatorname{Aut}(B^n)$  and every  $a, b, c, d, e, f \in D_n$ , we have  $F(a, b, c, d, e, f) = F(\pi(a), \pi(b), \pi(c), \pi(d), \pi(e), \pi(f))$ . Consider Cartesian product  $Y = Y_1 \times Y_2 \times Y_3 \times Y_4 \times Y_5$  and let  $\pi(y_1, \ldots, y_5) = (\pi(y_1), \ldots, \pi(y_5))$ . Observe that, if  $\pi(Y_i) = Y_i$  for every i, then  $\pi(Y) = Y$  and  $\pi$  permutes the elements of Y.

**Lemma 14.** Let Y be a subset  $Y \subseteq D_n^5$  and suppose that  $\pi(Y) = Y$  for every automorphism  $\pi \in \operatorname{Aut}(B^n)$ ; and let x and y be two equivalent,  $x \sim y$ , elements in  $D_n$ . Then

1.  $F(\{x\} \times Y) = F(\{y\} \times Y).$ 

2. 
$$F([x] \times Y) = \gamma(x) \cdot F(\{x\} \times Y).$$

Proof. (1)  $F({x} \times Y) = \sum_{s \in Y} F(x,s) = \sum_{s \in Y} F(\pi(x),\pi(s)) = \sum_{u \in \pi(Y)} F(\pi(x),u) = \sum_{u \in Y} F(\pi(x),u) = F({\pi(x)} \times Y)$ . We use the fact that  $\pi$  is a bijection on Y and permutes the elements of Y.

As a corollary we get the following result.

Theorem 15.

$$d_{n+4} \equiv \sum_{a \in R_n \cap E_{n,m}^c} \sum_{b \in E_{n,m}^c} \sum_{c \in E_{n,m}^c} \sum_{d \in E_{n,m}^c} \sum_{e \in E_{n,m}^c} \sum_{f \in E_{n,m}^c} \gamma(a) \cdot F(a, b, c, d, e, f) \pmod{m}.$$

**Example 16.** We utilized a Java implementation of the Theorem 15. For n = 4 and m = 2, 3, 4, 6, 12 we get

 $d_8 \equiv 53336702474849828$ , and therefore  $d_8 \mod 2 = 0$ ;

 $d_8 \equiv 3019662424037271148 \pmod{3}$ , and therefore  $d_8 \mod 3 = 1$ ;

 $d_8 \equiv 25754060568741983624 \pmod{4}$ , and therefore  $d_8 \mod 4 = 0$ ;

 $d_8 \equiv 14729824485525634108 \pmod{6}$ , and therefore  $d_8 \mod 6 = 4$ ;

 $d_8 \equiv 15054599294580333880 \pmod{12}$ , and therefore  $d_8 \mod 12 = 4$ .

One can check these values directly by dividing  $d_8$  by 2, 3, 4, 6, and 12.

## 9 Application

To compute remainders of  $d_9$  divided by 5 and 7, we chose the algorithm described in Section 6. Our implementation lists all 490,013,148 elements of  $R_7$  and calculates  $\gamma(x)$  and  $\operatorname{re}(\bot, x)$ for each element  $x \in R_7$ . This feat was previously accomplished only by Van Hirtum in 2021 [6]. It is worth noting that the number of elements x in  $R_n$  with  $\gamma(x) = n!$  for n > 1 can be found in the OEIS sequence <u>A220879</u> (see Table 6). Using the available precalculated sets, we can efficiently determine the 7th term of the sequence, which was not recorded in the OEIS before.

n	$\underline{A220879}(n)$
1	0
2	1
3	0
4	0
5	7
6	7281
7	468822749

Table 6: Inequivalent monotone Boolean functions of n variables with no symmetries.

Our program's most critical part, the Boolean function canonization procedure, is based on Van Hirtum's fast approach [6, Section 5.2.9] and implemented in Rust. Our program is running on a 32-thread machine with Xeon cores.

After the preprocessed data has been loaded into the main memory, the test was performed and the value of  $d_8$  was recomputed in just 16 seconds. However, using this method to check the divisibility of  $d_9$  for any value of m is significantly more challenging.

In order to determine which remainders can be computed by our methods, we can use the information in Table 7. Note that

$$|E_{7,m}^c| = \sum_{\substack{x \in R_7\\\gamma(x) \bmod m \neq 0}} \gamma(x).$$

The four smallest  $E_{7,m}^c$  are  $E_{7,7}^c$  with 9999 elements,  $E_{7,3}^c$  with 108873 elements,  $E_{7,21}^c$  with 118863 elements, and  $E_{7,5}^c$  with 154863 elements. Since  $d_9$  is already known to be divisible by 3, the next step is to compute the remainders of  $d_9$  divided by 5 and 7.

### 9.1 Remainder of $d_9$ divided by 5

$$\sum_{x \in R_7 \cap E_{7,5}^c} \sum_{y \in E_{7,5}^c} \gamma(x) \cdot G(x,y) = 1404812111893131438640857806,$$

k	$ \{f \in R_7 : \gamma(f) = k\} $
1	9
7	27
21	75
30	5
35	117
42	99
70	90
84	9
105	1206
120	4
140	702
210	3255
252	114
315	2742
360	18
420	26739
504	237
630	47242
720	4
840	75024
1260	1024050
1680	3128
2520	20005503
5040	468822749

Table 7: Number of  $f \in R_7$  with the given  $\gamma(f)$ .

therefore, by Theorem 7, we have  $d_9 \mod 5 = 1$ . We calculated this number in approximately 7 hours. Moreover, using Theorem 15 we have  $d_9 \equiv 157853570524864492086 \pmod{5}$ , which confirms that  $d_9 \mod 5 = 1$ .

# 9.2 Remainder of $d_9$ divided by 7

$$\sum_{x \in R_7 \cap E_{7,7}^c} \sum_{y \in E_{7,7}^c} \gamma(x) \cdot G(x,y) = 29989517764506682537562623,$$

therefore, by Theorem 7, we have  $d_9 \mod 7 = 6$ . We calculated this number in approximately half an hour.

### 10 Acknowledgments

We would like to thank the anonymous referee for his useful suggestions and references.

## References

- [1] M. Aigner. Combinatorial Theory. Springer, 1979.
- [2] A. E. Brouwer, C. F. Mills, W. H. Mills, and A. Verbeek. Counting families of mutually intersecting sets. *Electron. J. Comb.* 20 (2013).
- [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] R. Fidytek, A. W. Mostowski, R. Somla, and A. Szepietowski. Algorithms counting monotone Boolean functions. *Inform. Process. Lett.* **79** (2001), 203–209.
- C. Jäkel. A computation of the ninth Dedekind number, arxiv preprint arXiv:2304.00895 [math.CO], 2023.
   Available at https://arxiv.org/abs/2304.00895.
- [6] L. Van Hirtum. A path to compute the 9th Dedekind number using FPGA supercomputing (2021). Master's thesis, available at https://hirtum.com/thesis.pdf.
- [7] L. Van Hirtum, P. De Causmaecker, J. Goemaere, T. Kenter, H. Riebler, M. Lass, and C. Plessl. A computation of D(9) using FPGA supercomputing, preprint, 2023. Available at https://arxiv.org/abs/2304.03039.
- [8] B. Pawelski. On the number of inequivalent monotone Boolean functions of 8 variables. J. Integer Sequences 25 (2022), Article 25.7.7.
- [9] N. J. A. Sloane. The On-line Encyclopedia of Integer Sequences. Available at https://oeis.org.
- [10] A. Szepietowski. Fixes of permutations acting on monotone Boolean functions. J. Integer Sequences 25 (2022), Article 25.9.6.
- [11] D. Wiedemann. A computation of the eighth Dedekind number. Order 8 (1991), 5–6.
- [12] D. Wiedemann. Letter to N. J. A. Sloane (1990). Available at https://oeis.org/ A000372/a000372.pdf.
- [13] K. Yamamoto. Note on the order of free distributive lattices. Sci. Rep. Kanazawa Univ. 2 (1953), 5–6.

2020 Mathematics Subject Classification: Primary 06E30.

*Keywords*: Dedekind number, monotone Boolean function, ninth Dedekind number, congruence modulo, Chinese remainder theorem.

(Concerned with sequences <u>A000372</u>, <u>A001206</u>, <u>A003182</u> and <u>A220879</u>.)

Received March 15 2023; revised versions received March 16 2023; May 17 2023; June 13 2023; June 14 2023. Published in *Journal of Integer Sequences*, August 16 2023.

Return to Journal of Integer Sequences home page.

# On The Number of Inequivalent Monotone Boolean Functions of 9 Variables

Bartłomiej Pawelski

*Abstract*—The problem of counting all inequivalent monotone Boolean functions of nine variables is considered. We solve the problem using known algorithms and deriving new ones when necessary. We describe methods to count fixed points in sets of all monotone Boolean functions under a given permutation of input variables. With these techniques as a basis, we tabulate the cardinalities of these sets for nine variables. By applying Burnside's lemma and the numbers obtained, we calculate the number of inequivalent monotone Boolean functions of 9 variables, which equals 789,204,635,842,035,040,527,740,846,300,252,680.

Index Terms—Boolean functions, monotone Boolean functions, Dedekind number, inequivalent monotone Boolean functions

#### I. INTRODUCTION

ET *B* denote the set of two bits  $\{0,1\}$  and let  $B^n$  denote the set of *n*-element sequences of *B*. A Boolean function is any function  $f : B^n \to B$ . There are  $2^n$  elements in  $B^n$ and  $2^{2^n}$  Boolean functions of *n* variables.

We have a partial order in  $B: 0 \le 0, 0 \le 1$ , and  $1 \le 1$ . This partial order induces a partial order on  $B^n$ : for any two elements  $x = (x_1, \ldots, x_n)$  and  $y = (y_1, \ldots, y_n) \in B^n, x \le y$ if and only if  $x_i \le y_i$  for all *i*.

A Boolean function is said to be *monotone* if for any  $x, y \in B^n$ , when  $x \leq y$ , it follows that  $f(x) \leq f(y)$ .

We define  $D_n$  as the set of all monotone Boolean functions of n variables. Let  $d_n$  represent the cardinality of  $D_n$ , also known as the n-th Dedekind number. Dedekind numbers are listed in the On-Line Encyclopedia of Integer Sequences (OEIS) sequence A000372 (see Table I). The term  $d_n$  also corresponds to the number of simple games with n players in minimal winning form, the number of antichains of subsets of an n set, the number of Sperner families and the cardinality of a free distributive lattice on n generators [18]. An important and recent result is the calculation of  $d_9$ , which was independently achieved in 2023 by Jäkel [10] and Van Hirtum, De Causmaecker, Goemaere, Kenter, Riebler, Lass, and Plessl [9].

Monotone Boolean functions are instrumental in the design and study of nonlinear digital filters, like stack filters, which are widely used in image processing and other applications where noise sources are non–Gaussian or non–additive [17]. Other practical applications of monotone Boolean functions include, among others, machine learning algorithms which are not heuristic [13], optimization of consensus maximization algorithms for robust fitting in computer vision [20], game theory [8], and cryptography [6].

B. Pawelski was with the Institute of Informatics, University of Gdansk, Poland (e-mail: bartlomiej.pawelski@ug.edu.pl)

Two monotone Boolean functions are said to be *equivalent* if the first function can be obtained from the second function through any permutation of input variables. Let  $R_n$  represent the set of all equivalence classes of  $D_n$ , and let  $r_n$  denote the cardinality of this set, also known as the number of inequivalent monotone Boolean functions of n variables. The values of  $r_n$  are listed in the OEIS sequence A003182 (see Table I). Note that the value of  $r_9$ , first reported in this paper, has subsequently been included in the OEIS. There are at most n! permutations of n variables, so each equivalence class has at most n! Boolean functions. Therefore, a lower bound of  $r_n$  is given by  $\frac{d_n}{n!}$ .

1

In 1985 and 1986, Liu and Hu [11], [12] calculated  $r_n$  for values of n up to 7. Independently,  $r_7$  was calculated by Stephen and Yusun [19]. In 2021, the author calculated  $r_8$  [14], and this result was independently reported in 2022 by Carić and Živković [5]. The recent paper by Szepietowski [16] contributes to the topic by systematizing knowledge about studying monotone Boolean functions and counting fixed points of permutations acting on  $D_n$ . Notably, one of the algorithms from Szepietowski's paper has been applied in the current study (see Section III-C).

In this paper, our primary contribution is the computation of  $r_9$ . Our work focuses on counting the fixed points in  $D_9$  under permutations of the cycle types of its input variables. The OEIS sequence <u>A000041</u>, which is well-known for enumerating the number of integer partitions, also provides information on the number of different cycle types as n increases. Given this sequence, there are 29 cycle types for n = 9 (see Table V). Then, we apply a specialized version of Burnside's lemma, tailored specifically for these cycle types, similarly to the approaches found in [5], [11], [12], [14]. Given the complexity and diversity of these cycle types, we leverage existing algorithms where suitable and introduce new ones when demanded by specific computational challenges:

- 1) Algorithm III-A, corresponding to [14, Algorithm 1].
- 2) Algorithm III-B, a partially new algorithm, evolved from [14, Algorithm 2].
- 3) Algorithm III-C, corresponding to [16, Theorem 4.].
- Algorithms III-D and III-E: new algorithms, evolved from [14, Algorithm 3].

The methodology is further elaborated upon in Section III. Our efforts culminate in the calculation of

 $r_9 = 789204635842035040527740846300252680,$ 

marking the first-ever computation of this number.

This article has been accepted for publication in IEEE Transactions on Information Theory. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIT.2024.3379594

n	$d_n$	$r_n$
0	2	2
1	3	3
2	6	5
3	20	10
4	168	30
5	7581	210
6	7828354	16353
7	2414682040998	490013048
8	56130437228687557907788	1392195548889993358
9	286386577668298411128469151667598498812366	789204635842035040527740846300252680

TABLE I: Known values of  $d_n$  and  $r_n$ .

#### **II. PRELIMINARIES**

#### A. Integer representation of a Boolean function

In the context of the n-dimensional Boolean space, denoted as  $B^n$ , we can consider an array composed of  $2^n$  potential inputs for a given Boolean function. Each of these possible input combinations is associated with a specific output bit.

In this manner, we represent any Boolean function of n variables using a binary vector of length  $2^n$ , which is called a truth table representation of a Boolean function (also see [2] and [3, Chapter II]). Note that a Boolean function of up to six variables can be effectively represented as a 64-bit integer, which corresponds to the word length of modern CPUs, making it a particularly efficient representation for computing purposes.

For  $f, g \in D_n$ , the union of these functions is represented by  $f \cup g$ , and the intersection by  $f \cap g$ . In terms of their binary representations, the union can be represented using the bitwise OR operation and the intersection using the bitwise AND operation.

Example 1. Consider the following truth table:

000	001	010	011	100	101	110	111
0	1	0	1	0	1	0	1

The first row contains the lexicographically ordered elements of  $B^n$ , and the second row a function value f(x) for a given  $x \in B^n$ . We represent this Boolean function by the binary word 01010101, equivalent to the integer 85.

**Example 2.** The six monotone Boolean functions of two variables are: 0000 (0 as integer), 0001 (1 as integer), 0011 (3 as integer), 0101 (5 as integer), 0111 (7 as integer), and 1111 (15 as integer).

#### B. Posets

A binary relation that is reflexive, antisymmetric and transitive, when defined on a set P, forms a *partially ordered set*, or simply a *poset*. In the Introduction, we defined partial orders on B and  $B^n$  making them both posets.

Consider a poset  $P = (X, \leq)$ . A *downset* of P is defined as a subset  $S \subseteq X$  such that if  $x \in S$ , then all elements in Xsatisfying  $y \leq x$  also belong to S. The set  $D_n$  corresponds to the set of all downsets of  $B^n$ . As a result, every element of  $D_n$  can be associated with a specific downset of  $B^n$ . An *incidence matrix* of a poset, also referred to as an *array* in Szepietowski's paper [16, Section 3] is a binary matrix that represents the partial order relation between elements in the poset. For a poset  $P = (X, \leq)$ , its incidence matrix M has rows and columns indexed by elements in X. The entry M(x, y) is 1 if  $x \leq y$ , and 0 otherwise.

Consider two posets  $(X, \leq)$  and  $(Y, \leq)$ . The Cartesian product  $X \times Y$  is the poset with the relation  $\leq$  defined by  $(a,b) \leq (c,d)$  if and only if  $a \leq c$  and  $b \leq d$ . For two disjoint posets  $(X, \leq)$  and  $(Y, \leq)$ , by X + Y we denote the disjoint union (sum) with the order defined as follows:  $a \leq b$ iff  $(a, b \in X \text{ and } a \leq b)$  or  $(a, b \in Y \text{ and } a \leq b)$ .

Let us recall the function  $f : X \to Y$  is monotone if for any elements  $x, y \in X$  such that  $x \leq y$ , we have  $f(x) \leq f(y)$ .

We denote the set of all monotone functions from X to Y by  $Y^X$ . A partial order on  $Y^X$  can be defined as follows: given two functions  $f, g \in Y^X$ , we say that  $f \leq g$  if and only if  $f(x) \leq g(x)$  for all  $x \in X$ .

Two posets are said to be *isomorphic* if their order structures are analogous. Formally, posets  $P = (X, \leq)$  and  $P' = (X', \leq')$ are isomorphic, if there exists a bijection  $f : X \to X'$ such that for all elements  $x_1, x_2$  in  $P, x_1 \leq x_2$  if and only if  $f(x_1) \leq' f(x_2)$ . We use the equals sign = to denote an isomorphic relationship between posets.

In the current study, we use the following lemmas:

**Lemma 3.** [16] For three posets R, S, T:

- (1) If S and T are disjoint, then  $R^{S+T} = R^S \times R^T$ .
- (2)  $R^{S \times T} = (R^S)^T$  and also  $R^{S \times T} = (R^T)^S$ .

#### Lemma 4. [16]

(a)  $B^{k+m} = B^k \times B^m$ .

(b) 
$$D_{k+m} = (D_k)^{B^m}$$
.

These lemmas, in various formulations, are standard tools in the literature for counting monotone Boolean functions. For example, Wiedemann [21] used the isomorphism  $D_8 = (D_6)^{B^2}$  to count the monotone Boolean functions of eight variables.

#### C. Permutations acting on variables of a Boolean function

Let  $S_n$  represent the set of all permutations of the set  $\{1, 2, ..., n\}$ . In this context, we treat  $S_n$  as the set of all permutations of the *n* variables within a Boolean function. The action of a permutation  $\pi$  on a Boolean function *f* essentially

permutes the input variables of f. For a given Boolean function f and a permutation  $\pi$ , we denote the resulting function as  $\pi(f)$ .

Each permutation  $\pi \in S_n$  can be represented as a product of disjoint cycles. The *cycle type* of  $\pi$  is defined as the tuple of lengths of its disjoint cycles arranged in increasing order. For example, the type of permutation  $\pi = (12)(34)(5678)$  is (2, 2, 4), and its total length is 8.

Given a set X and a permutation  $\pi$  acting on X, the *orbit* of an element  $x \in X$  under the action of  $\pi$  is the set of all images of x obtained by applying  $\pi$  repeatedly. Formally, the orbit of x is:

$$\operatorname{Orb}_{\pi}(x) = \{\pi^{j}(x) \mid j \in \mathbb{N}\}$$

where  $\pi^{j}(x)$  denotes the result of applying the permutation  $\pi$  *j* times to *x*.

An element x is said to be a *fixed point* of  $\pi$  if it remains unchanged under the action of  $\pi$ . The set  $\Phi_n(\pi)$  contains all fixed points of  $\pi$  acting on  $D_n$ . In this context, let *e* represent the identity permutation, where none of the variables is swapped. This is shown in the first row of Table II, where each element of  $B^3$  remains unchanged.

e	000	001	010	011	100	101	110	111
(12)	000	010	001	011	100	110	101	111

TABLE II:  $B^3$  under  $\pi = (12)$ .

TABLE III: A Boolean function of three variables represented by 85 becomes 51 under  $\pi = (12)$ .

**Example 5.** As shown in Table III, the Boolean function of three variables represented by the integer 85 (in binary form as 01010101) is transformed under permutation  $\pi = (12)$ . After the permutation, the binary representation becomes 00110011, which corresponds to the integer 51. We therefore have  $\pi(85) = 51$ . This transformation allows us to illustrate two significant insights:

- 1) 51 and 85 are equivalent Boolean functions, illustrating that the application of a permutation on variables can yield a different, but equivalent, Boolean function;
- 2) neither 51 nor 85 are fixed points in  $D_3$  under the permutation  $\pi = (12)$ , illustrating that under certain permutations, original functions do not remain invariant.

A simple example of a fixed point in  $D_3$  under  $\pi = (12)$  is the function represented by the integer 255, corresponding to the binary word 11111111. Indeed, this function remains invariant under any permutation, which can be observed upon examining its binary representation.

111 q111 011 101 110110 101011001 c 010100010 001 1006000 6000

Fig. 1:  $B^3$  and  $B^3$  under  $\pi = (12)$  on Hasse diagrams

By applying Burnside's lemma,  $r_n$  can be calculated as follows [5], [14], [16]:

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

where:

- $\phi_n(\pi) = |\Phi_n(\pi)|,$
- k is the number of different cycle types in  $S_n$ ,
- -i is the index of a cycle type,
- $\mu_i$  is the number of permutations  $\pi \in S_n$  with cycle type i,
- $-\pi_i$  is a representative permutation  $\pi \in S_n$  with cycle type *i*.

The first application of Burnside's lemma for calculating  $r_n$  we found in the literature was by Liu and Hu [11], [12]. They used it in 1985 and 1986 to determine  $r_n$  for values of n up to 7. Computing  $\phi_9(\pi)$  for all cycle types in  $S_9$  (excluding the identity permutation) requires significantly less computational effort compared to calculating  $d_9$ , making it feasible with our available resources.

#### III. METHODOLOGY

To efficiently calculate  $\phi_9(\pi)$  for all cases, we employ several algorithms, the details of which are described in this section.

#### A. Algorithm III–A

Let  $B^n(\pi)$  denote the poset of orbits of  $B^n$  under  $\pi \in S_n$ , where two orbits  $C_1$  and  $C_2$  are in the relation  $C_1 \leq_{\pi} C_2$  if and only if for every  $c_1 \in C_1$ , there exists a  $c_2 \in C_2$  such that  $c_1 \leq c_2$ . An illustrative representation of this can be seen in Figure 2. Note that this figure includes two doublet orbits:  $\{001, 010\}$  and  $\{101, 110\}$ .



Fig. 2:  $B^3(\pi)$  - poset of orbits in  $B^3$  under  $\pi = (12)$ .

### **Lemma 6.** $\Phi_n(\pi) = B^{B^n(\pi)}$ .

*Proof.* Recall that  $\Phi_n(\pi)$  represents the set of fixed points of  $\pi$  acting on  $D_n$ . Given an orbit in  $B^n(\pi)$ , a function in  $\Phi_n(\pi)$  determines a specific bit value (either 0 or 1) for the entire orbit. This assignment of bit values to orbits can be represented as a function from  $B^n(\pi)$  to B, where each orbit is mapped to a single bit value. The set of all possible ways to assign bit values to orbits in  $B^n(\pi)$  thus forms the set  $B^{B^n(\pi)}$ . Since every function in  $\Phi_n(\pi)$  corresponds uniquely to such an assignment, and vice versa, we establish a one-toone correspondence between  $\Phi_n(\pi)$  and  $B^{B^n(\pi)}$ .

The essence of this algorithm is to count  $\phi_n(\pi)$  by generating all downsets of  $B_n(\pi)$  recursively and exploiting the fact proven in Lemma 6, that  $\phi_n(\pi) = |B^{B^n(\pi)}|$ . For a pseudocode of Algorithm III-A, please refer to [14, Algorithm 1].

A significant limitation of this approach, however, is that it requires storing all elements of the set rather than simply counting them. This constraint makes the application of this algorithm highly dependent on the available main memory. For reference, our calculations were performed on a machine with 128 GB of main memory, and we were able to compute up to  $\phi_9((123)(456789)) = 218542866$ . Hence, in practice, all sets of fixed points with cardinalities smaller than about two hundred million can be computed using the resources at our disposal. The number 218542866 represents the upper limit of our computational capabilities in the context of hardware limitations, and not limitations of the method or the algorithm itself.

#### B. Algorithm III-B

In order to count the fixed points in  $D_n$  under permutations containing at least one disjoint 1-cycle, we can leverage the isomorphism delineated by the subsequent lemma:

**Lemma 7.** Let k denote the number of 1-cycles in a permutation  $\pi$ . Then,  $\Phi_{n+k}(\pi) = \Phi_n(\pi)^{B^k}$ .

*Proof.* Start by recalling the definition  $\Phi_{n+k}(\pi) = B^{B^{n+k}(\pi)}$ . Using Lemma 4(a), we can express the term  $B^{B^{n+k}(\pi)}$  as  $B^{B^n(\pi) \times B^k}$ , given that the posets  $B^n(\pi)$  and  $B^k$  are disjoint and the last k variables are not affected by  $\pi$ .

Next, by applying Lemma 3(2),  $B^{B^n(\pi) \times B^k}$  can be expressed as  $(B^{B^n(\pi)})^{B^k} = \Phi_n(\pi)^{B^k}$ .

In [14, Section 3.2], we outline the simplest version of the algorithm that uses this isomorphism for the case where k = 1. Carić and Živković elaborate on this algorithm for the situation where k = 2 [5, Theorem 3.1]. Several studies in the literature provide detailed descriptions of algorithms for counting monotone Boolean functions. Notably, in the context of our work, these algorithms are applicable to cases involving the identity permutation. By Lemma 7, these algorithms can be adapted to our needs through modification of the input set. For example, Fidytek et al. [7] describe algorithms for  $k \in \{1, 2, 4\}$ , while a Campo [4, Section 4.1] presents algorithms for  $k \in \{1, 2, 3, 4\}$ .

In order to calculate  $r_9$ , we extend the application of Lemma 7 to handle cases up to k = 4. To illustrate our approach, for

example, to calculate  $\phi_9((12)(34))$ , we first generate the set  $\Phi_6((12)(34))$  using our implementation of Algorithm III-A, and then apply Algorithm III-B with k = 3.

In the calculation of  $r_8$ , there is only one case where the algorithms referenced above (detailed in [14]) did not work due to computational resource limitations: specifically,  $\phi_8((12)(34)(56)(78))$ . In the calculation of  $r_9$ , we have three cases:  $\phi_9((12)(34)(56)(78))$ ,  $\phi_9((12)(34)(56)(789))$ , and  $\phi_9((123)(456)(789))$ .

#### C. Calculation of $\phi_9((12)(34)(56)(789))$

Let  $k \in \{2,3\}$ . Let  $\tau$  represent a permutation where all cycle lengths are coprime to k. Define  $\epsilon$  as the permutation obtained from the composition of  $\tau$  and a single-cycle permutation of length k, ensuring that these permutations are disjoint. Let  $P_n$  denote a special type of poset, known as a *chain*, where each pair of n elements is comparable.

**Lemma 8.** [1], [16] For any  $n, m \in \mathbb{N}$ , the number of functions from  $(B^n)^{P_m}$  is equal to the sum of elements of the (m-1)-power of the incidence matrix of  $B^n$ .

**Corollary 9.** For  $k \in \{2, 3\}$ , as poset  $B^2((12))$  is isomorphic to the chain  $P_3$  and poset  $B^3((123))$  is isomorphic to the chain  $P_4$ , we have  $B^{n+k}(\epsilon) = B^n(\tau)^{P_{k+1}}$ .

Lemma 8 indicates that the number of functions from a given poset to another poset, being a chain, can be calculated by raising its incidence matrix to the appropriate power and summing all its elements. This provides a computationally efficient technique for dealing with some complex poset structures. In Corollary 9, we use this lemma leaning on two isomorphisms:  $B^2((12)) = P_3$  and  $B^3((123)) = P_4$ . This sets the stage for the subsequent corollary, first applied by Szepietowski [16].

**Corollary 10.** For  $k \in \{2,3\}$ ,  $\phi_{n+k}(\epsilon)$  is equal to the sum of elements of the (k-1)-power of the incidence matrix of  $(B^n(\tau))$ .

We use this property to compute

$$\phi_9((12)(34)(56)(789)) = 807900672006$$

by summing all the elements of the third power of the incidence matrix of  $\Phi_6((12)(34)(56))$ , which has  $8600 \times 8600$  elements. Notably, using Algorithm III-A to compute this number would have been impossible due to the immense memory required to store all elements.

#### D. Calculation of $\phi_9((12)(34)(56)(78))$

We adopt the notation from the previous subsection with minor modifications. Let  $\tau$  represent a permutation where all cycle lengths are either 1 or 2, and the total length equals n. Now, denote  $\epsilon$  as the permutation obtained from the composition of  $\tau$  and the single cycle permutation of length 2, ensuring that these permutations are disjoint.

**Lemma 11.** For k = 2 and  $n \ge k$ , there exists a decomposition of the poset  $B^{n+2}$  into four posets:  $R_{00}, R_{01}, R_{10}, R_{11}$ , which forms a poset isomorphic to  $B^2$ .

The permutation  $\epsilon$  acting on  $B^{n+2}$  affects these four parts in the following manner:

- 1)  $R_{00}(\epsilon)$  is isomorphic to  $B^n(\tau)$ .
- 2)  $R_{01}(\epsilon)$  and  $R_{10}(\epsilon)$  form  $2^n$  cycles of length 2: in each cycle, we have the *i*-th element of  $R_{01}$  and the *i*-th element of  $R_{10}(\tau)$ .
- 3)  $R_{11}(\epsilon)$  is isomorphic to  $B^n(\tau)$ .

This method is also described in [14, Algorithm 3] (see also [5, Section IV], [16, Section 6.5]) as a response to the need to calculate  $\phi_8(\pi)$  under  $\pi$  with all cycle lengths of 2. Upon specifying  $\epsilon$ , Lemma 11 can be proved by listing  $B^{n+2}(\epsilon)$  and decomposing it into four equal parts. Additionally, a computerassisted proof for the necessary poset  $B^9((12)(34)(56)(78))$ is available at [15] in the file P0. java.

**Example 12.** Decomposition of  $B^4((12)(34))$  according to Lemma 11:

- 1)  $R_{00}$  as ((0)(1 2)(3), which is isomorphic to  $B^2((12))$ ;
- 2)  $R_{01}, R_{10}$  as four 2–element cycles: (4 8) (5 10) (6 9) (7 11);
- 3)  $R_{11}$  as ((12)(13 14)(15)), which is isomorphic to  $B^2((12))$ .

**Lemma 13.** Each function  $x \in \Phi_9((12)(34)(56)(78))$  can be combined from these four functions:

1)  $a \in \Phi_7((12)(34)(56));$ 

- 2)  $b \in D_7, b \ge a;$
- 3)  $c = \tau(b), c \ge a;$
- 4)  $d \in \Phi_7((12)(34)(56)), d \ge b \cup c.$

*Proof.* Considering the decomposition of the poset  $B^{n+2}$  given by Lemma 11, we can infer that  $R_{00}$  and  $R_{11}$  are affected by permutation  $\tau = (12)(34)(56)$ , so they are isomorphic to the poset  $B^7((12)(34)(56))$ . The set of functions  $B^{B^7((12)(34)(56))}$  is equal to  $\Phi_7((12)(34)(56))$ , thus we have  $a, d \in \Phi_7((12)(34)(56))$ . Moving on to the subposets  $R_{01}$  and  $R_{10}$ , we find that while the functions b and c belong to  $D_7$ , they pair in a bounded manner, specifically  $c = \tau(b)$ . Moreover, as these four elements are organized on the poset  $B^2$ , we must adhere to the constraints:  $a \leq b \leq d$  and  $a \leq c \leq d$ .

Algorithm III-D Calculation of $\phi_9((12)(34)(56)(78))$					
<b>Input:</b> $D_7$ and $\Phi_7((12)(34)(56))$					
<b>Output:</b> $s = \phi_9((12)(34)(56)(78))$					
1: Initialize $s = 0$					
2: for all $b \in D_7$ do					
3: Initialize $c = \tau(b)$					
4: Calculate $down =$					
$ \{a \in \Phi_7((12)(34)(56)) : a \le (b \cap c)\} $					
5: Calculate $up =$					
$ \{d \in \Phi_7((12)(34)(56)) : d \ge (b \cup c)\} $					
$6: \qquad s = s + down \cdot up$					
7: end for					

 $\Phi_7(12)(34)(56)$  contains 12015832 elements, so we can store it in main memory. The most challenging part is storing  $D_7$  (which would require about 35 TB, see Table IV).

п	$d_n$	Size of function	Size of $D_n$
0	2	1b	2 B
1	3	2b	3 B
2	6	4b	6 B
3	20	8b	20 B
4	168	16b	336 B
5	7581	32b	29.61 KB
6	7828354	64b	59.725 MB
7	2414682040998	128b	35.138 TB

TABLE IV: Sizes of  $D_n$  up to  $D_7$ .

We overcome this by storing  $R_7$ , which contains 490013148 elements that fit in main memory. Then, for each  $x \in R_7$ , we execute an "unpacking" operation, exploiting 7! = 5040possible permutations of input variables. Finally, we obtained

 $\phi_9((12)(34)(56)(78)) = 17143334331688770356814.$ 

#### E. Calculation of $\phi_9((123)(456)(789))$

Once again, let us revisit the notation introduced in the previous subsection with a slight modification. This time, let  $\tau$  represent a permutation where all cycle lengths are either 1 or 3, and the total length equals n. Now, denote  $\epsilon$  as the permutation obtained from the composition of  $\tau$  and the single cycle permutation of length 3, ensuring that these permutations are disjoint. With these conventions in place, we are able to state and prove a lemma that resembles the previous one.

**Lemma 14.** For k = 3 and  $n \ge k$ , there exists a decomposition of the poset  $B^{n+3}$  into eight posets:  $R_{000}, R_{001}, R_{010}, \ldots, R_{111}$ , which forms a poset isomorphic to  $B^3$ .

The permutation  $\epsilon$  acting on  $B^{n+3}$  acts on those eight parts in the following way:

- 1)  $R_{000}(\epsilon)$  is isomorphic to  $B^n(\tau)$ .
- R<sub>001</sub>(ε), R<sub>010</sub>(ε), and R<sub>100</sub>(ε) form 2<sup>n</sup> cycles of length
   3: in each cycle, we have the *i*-th element of R<sub>001</sub>, the *i*-th element of R<sub>010</sub>(τ), and the *i*-th element of R<sub>100</sub>(τ<sup>2</sup>).
- 3)  $R_{011}(\epsilon)$ ,  $R_{101}(\epsilon)$ , and  $R_{110}(\epsilon)$  form  $2^n$  cycles of length 3: in each cycle, we have the *i*-th element of  $R_{011}$ , the *i*-th element of  $R_{101}(\tau)$ , and the *i*-th element of  $R_{110}(\tau^2)$ .
- 4)  $R_{111}(\epsilon)$  is isomorphic to  $B^n(\tau)$ .

If  $\epsilon$  is specified, Lemma 14 can be proved by listing  $B^{n+3}(\epsilon)$  and decomposing it into eight equal parts. Additionally, a computer-assisted proof for the necessary poset  $B^9((123)(456)(789))$  is available at [15] in the file P1.java.

**Example 15.** Decomposition of  $B^6((123)(456))$  according to Lemma 14:

- 1)  $R_{000}$  as ((0)(1 2 4)(3 6 5)(7)), which is isomorphic to  $B^3((123))$ ;
- 2)  $R_{001}, R_{010}, R_{100}$  as eight 3-element cycles: (8 16 32) (9 18 36) (10 20 33) (11 22 37) (12 17 34) (13 19 38) (14 21 35) (15 23 39);
- 3)  $R_{011}, R_{101}, R_{110}$  as eight 3–element cycles: (24 48 40) (25 50 44) (26 52 41) (27 54 45) (28 49 42) (29 51 46) (30 53 43) (31 55 47);
- 4)  $R_{111}$  as ((56)(57 58 60)(59 62 61)(63)), which is isomorphic to  $B^3((123))$ .

**Lemma 16.** Each function  $x \in \Phi_9((123)(456)(789))$  can be combined from these eight functions:

1)  $a \in \Phi_6((123)(456));$ 2)  $b \in D_6, b \ge a;$ 3)  $c = \tau(b), c \ge a;$ 

4)  $d = \tau(c), d \ge a;$ 

5)  $e \in D_6, e \ge b \cup c$ ;

6)  $f = \tau(e), f \ge b \cup d;$ 

7)  $g = \tau(f), g \ge c \cup d;$ 

8)  $h \in \Phi_6((123)(456)), h \ge e \cup f \cup g.$ 

*Proof.* Considering the decomposition of the poset  $B^{n+3}$  given by Lemma 14, we can infer that  $R_{000}$  and  $R_{111}$  are affected by permutation  $\tau = (123)(346)$ , so they are isomorphic to the poset  $B^6((123)(456))$ . The set of functions  $B^{B^6((123)(456))}$  is equal to  $\Phi_6((123)(456))$ .

Moving on to the subposets  $R_{001}$ ,  $R_{010}$  and  $R_{100}$ , we find that while the functions b, c and d belong to  $D_6$ , they are structured as bounded triplets, specifically  $c = \tau(b)$  and  $d = \tau(c)$ , A similar structure is evident with the subposets  $R_{011}$ ,  $R_{101}$  and  $R_{110}$ .

Moreover, as these eight elements are organized on the poset  $B^3$ , we must adhere to the constraints:  $b \ge a, c \ge a, \ldots, h \ge e \cup f \cup g$ .

Algorithm III-E	Calculation	of $\phi_9($	(123)	(456)	(789)	))
-----------------	-------------	--------------	-------	-------	-------	----

**Input:**  $D_6$  and  $\Phi_6((123)(456)$ **Output:**  $s = \phi_9((123)(456)(789))$ 1: Initialize s = 02: for all  $b \in D_6$  do Initialize  $c = \tau(b)$ 3: Initialize  $d = \tau(c)$ 4. Calculate down =5:  $|\{a \in \Phi_7((123)(456)) : a \le (b \cap c \cap d)\}|$ for all  $e \in D_6, e \ge b \cup c$  do 6: Initialize  $f = \tau(e)$ 7: Initialize  $g = \tau(f)$ 8: 9: Calculate down = $|\{h \in \Phi_7((123)(456)) : h \ge (e \cup f \cup g)\}|$ 10:  $s = s + down \cdot up$ end for 11: 12: end for

The essence of the algorithm allowing to calculate  $\phi_9((123)(456)(789))$  is fully captured in Lemma 16, which is similar in spirit to Lemma 13.  $D_6$  has 7828354 elements and  $\Phi_6((123)(456))$  has 562 elements, so they fit in the main memory. We completed a calculation in about six hours with a result

$$\phi_9((123)(456)(789)) = 221557843276152.$$

#### F. Computational results

The algorithms were implemented in Java. Additionally, for some cases, we have also written the implementation in Rust (for example, for Algorithm III-D) due to its native support of 128-bit integers. We utilized a machine with 32 Xeon threads, and each case was calculated separately.

After executing all the implemented algorithms, we estimate the total computation time to be about 10 days. The simplest cases are computed almost instantly, while the most challenging ones (for example,  $\phi_9((12)(34))$ ) take up to 2 days. The complexity is primarily due to the size and structure of the orbit poset associated with each permutation.

Note that  $\phi_9(e)$ , which represents the value for the identity permutation, is equal to  $d_9$ . This specific case was not computed by us and is therefore not included in the table.

$\pi_i$	$\mu_i$	(1)	$\phi_9(\pi_i)$
(12)	36	В	16278282012194909428324143293364
(123)	168	В	868329572680304346696
(1234)	756	В	5293103318608452
(12345)	3024	В	26258306096
(123456)	10080	В	2279384919
(1234567)	25920	А	3268698
(12345678)	45360	А	1144094
(123456789)	40320	Α	97830
(12)(34)	378	В	107622766375525877620879430
(12)(345)	2520	В	5166662396125146
(12)(3456)	7560	В	323787762940974
(12)(34567)	18144	В	70165054
(12)(345678)	30240	В	547120947
(12)(3456789)	25920	Α	80720
(123)(456)	3360	В	7107360458115201
(123)(4567)	15120	В	92605092
(123)(45678)	24192	Α	197576
(123)(456789)	20160	Α	218542866
(123)(456)(789)	2240	Е	221557843276152
(1234)(5678)	11340	В	503500313130
(1234)(56789)	18144	Α	10182
(12)(34)(56)	1260	В	328719964864138799170044
(12)(34)(567)	7560	А	14037774553676
(12)(34)(5678)	11340	А	66031909836340
(12)(34)(56789)	9072	А	3710840
(12)(345)(678)	10080	Α	866494196253
(12)(345)(6789)	15120	Α	22062570
(12)(34)(56)(78)	945	D	17143334331688770356814
(12)(34)(56)(789)	2520	С	807900672006

 $\sum_{i=2}^{\kappa} \mu_i \phi_9(\pi_i) = 586059264378237446637837193706034.$ 

TABLE V: Values of  $\phi_9(\pi)$  under all cycle types of  $\pi$  (excluding the identity). Column (1) indicates the algorithm used, corresponding to the subsections in Section 3 (e.g., 'A' refers to Section 3A, 'B' to Section 3B, etc.).

#### IV. CALCULATION OF $r_9$

In April 2023, two research teams independently reported the following value of  $d_9$  [10], [9]:

#### $d_9 = 286386577668298411128469151667598498812366.$

We can now finally make direct use of Equation 1, obtaining the following value:

#### $r_9 = 789204635842035040527740846300252680.$

We have a high degree of confidence in the accuracy of our results, supported by the fact that the entire sum:

$$\sum_{i=1}^k \mu_i \phi_9(\pi_i)$$

we obtained is divisible by 9! = 362880. The source code of the project is available in the GitHub repository [15].

#### V. CONCLUSION

The algorithms for generating and counting the fixed points in the set of monotone Boolean functions under given permutation of input variables have been described. Furthermore, the methods described were used as a basis for tabulating the numbers of fixed points under permutations of all cycle types in  $D_9$ . By applying Burnside's lemma and the cardinalities obtained, we computed the number of inequivalent monotone Boolean functions of 9 variables, which equals  $r_9 = 789204635842035040527740846300252680$ .

#### REFERENCES

- [1] M. Aigner, Combinatorial theory. Springer Verlag, 1979.
- [2] V. Bakoev, Fast bitwise implementation of the algebraic normal form transform, *Serdica J. Computing* **11** (2017).
- [3] C. R. Baugh, Generation of representative functions of the NPN equivalence classes of unate Boolean functions, *IEEE Trans. Comput.* (1972).
- [4] F. a Campo, Relations between powers of Dedekind numbers and exponential sums related to them, J. Integer Sequences 21 (2018), Article 18.4.4.
- [5] M. Carić, M. Živković, The number of nonequivalent monotone Boolean functions of 8 variables, *IEEE Trans. Inf. Theory* (2022).
- [6] C. Carlet, D. Joyner, P. Stănică, and D. Tang, Cryptographic properties of monotone Boolean functions, J. Math. Cryptol., 10 (2016).
- [7] R. Fidytek, A. W. Mostowski, R. Somla, and A. Szepietowski, Algorithms counting monotone Boolean functions, *Inform. Process. Lett.* **79** (2001), 203–209.
- [8] J. Freixas, On the enumeration of some inequivalent monotone Boolean functions, *Optimization* (2022).
- [9] L. Van Hirtum, P. De Causmaecker, J. Goemaere, T. Kenter, H. Riebler, M. Lass, and C. Plessl, A computation of D(9) using FPGA supercomputing, preprint, 2023. Available at https://arxiv.org/abs/2304.03039.
- [10] C. Jäkel, A computation of the ninth Dedekind number, Journal of Computational Algebra 6-7 (2023)
- [11] 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.
- [12] 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.
- [13] B, Kovalerchuk, E. Triantaphyllou, A. S. Deshpande, and E. Vityaev, Interactive learning of monotone Boolean functions, *Inf. Sci.*, 94 (1996).
- [14] B. Pawelski, On the number of inequivalent monotone Boolean functions of 8 variables, J. Integer Sequences 25 (2022), Article 25.7.7.
- [15] B. Pawelski, Algorithms counting fixed points in sets of monotone Boolean functions (2023), GitHub repository
- [16] A. Szepietowski, Fixes of permutations acting on monotone Boolean functions, J. Integer Sequences 25 (2022), Article 25.9.6.
- [17] I. Shmulevich, O.P. Yli-Harja, J. Astola, and A.D. Korshunov, On the robustness of the class of stack filters, *IEEE Trans. Signal Process.*, 50 (2002).
- [18] OEIS Foundation Inc. (2023). Dedekind numbers, Entry A000372 in The On-Line Encyclopedia of Integer Sequences, https://oeis.org/A000372
- [19] T. Stephen and T. Yusun, Counting inequivalent monotone Boolean functions, Discrete Appl. Math., 167 (2014), 15–24.

- [20] R. Tennakoon, D. Suter, E. Zhang, T. Chin, and A. Bab-Hadiashar, Consensus maximisation using influences of monotone Boolean functions, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021).
- [21] D. Wiedemann, A computation of the eighth Dedekind number, Order 8 (1991), 5–6.

Bartłomiej Pawelski is affiliated with the University of Gdansk, Poland. His current research interests include Boolean functions, especially the integer sequences related to them.

# Counting self-dual monotone Boolean functions

Bartłomiej Pawelski and Andrzej Szepietowski Institute of Informatics University of Gdańsk Poland

bartlomiej.pawelski@ug.edu.pl
andrzej.szepietowski@ug.edu.pl

#### Abstract

Let  $D_n$  denote the set of monotone Boolean functions with n variables. Elements of  $D_n$  can be represented as strings of bits of length  $2^n$ . Two elements of  $D_0$  are represented as 0 and 1 and any element  $g \in D_n$ , with n > 0, is represented as a concatenation  $g_0 \cdot g_1$ , where  $g_0, g_1 \in D_{n-1}$  and  $g_0 \leq g_1$ . For each  $x \in D_n$ , we have *dual*  $x^* \in D_n$  which is obtained by reversing and negating all bits. An element  $x \in D_n$  is a *self-dual* if  $x = x^*$ . Let  $\Lambda_n$  denote the set of all self-dual monotone Boolean functions of n variables and let  $\lambda_n$  denote  $|\Lambda_n|$  The value  $\lambda_n$  is also known as the n-th Hosten-Morris number. Any two Boolean functions are said to be *equivalent* if one can be transformed into the other by a permutation of input variables. Let  $Q_n$  denote the set of all equivalence classes in  $\Lambda_n$  and let  $q_n$  denote  $|Q_n|$ . In this paper, we derive several algorithms for counting self-dual monotone Boolean functions and confirm the known result that  $\lambda_9$  equals 423,295,099,074,735,261,880. Furthermore, we calculate  $q_8$  to be 6,001,501, which marks the first-ever calculation of this value.

## 1 Introduction

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 is the order relation in *B* (namely:  $0 \le 0, 0 \le 1, 1 \le 1$ ) and the partial order in  $B^n$ : for any two elements:  $x = (x_1, \ldots, x_n),$  $y = (y_1, \ldots, y_n)$  in  $B^n, x \le y$  if and only if  $x_i \le y_i$  for all  $1 \le i \le n$ . A function  $h: B^n \to B$ is monotone if  $x \le y$  implies  $h(x) \le h(y)$ . Let  $D_n$  denote the set of monotone functions with *n* variables and let  $d_n$  denote  $|D_n|$ . We have the partial order in  $D_n$  defined by:

 $g \leq h$  if and only if  $g(x) \leq h(x)$  for all  $x \in B^n$ .

We shall represent the elements of  $D_n$  as strings of bits of length  $2^n$ . Two elements of  $D_0$  will be represented as 0 and 1. Any element  $g \in D_1$  can be represented as the concatenation

 $g(0) \cdot g(1)$ , where  $g(0), g(1) \in D_0$  and  $g(0) \leq g(1)$ . Hence,  $D_1 = \{00, 01, 11\}$ . Each element of  $g \in D_2$  is the concatenation (string) of four bits:  $g(00) \cdot g(10) \cdot g(01) \cdot g(11)$  which can be represented as a concatenation  $g_0 \cdot g_1$ , where  $g_0, g_1 \in D_1$  and  $g_0 \leq g_1$ . Hence,  $D_2 = \{0000, 0001, 0011, 0101, 0111, 1111\}$ . Similarly, any element of  $g \in D_n$  can be represented as a concatenation  $g_0 \cdot g_1$ , where  $g_0, g_1 \in D_{n-1}$  and  $g_0 \leq g_1$ .

For each  $x \in D_n$ , we have dual  $x^* \in D_n$ , which is obtained by reversing and negating all bits. For example,  $1111^* = 0000$  and  $0001^* = 0111$ . An element  $x \in D_n$  is a self-dual if  $x = x^*$ . For example, 0101 and 0011 are self-duals in  $D_2$ . Let  $\Lambda_n$  be the set of all self-dual monotone Boolean functions of n variables, and let  $\lambda_n$  denote the cardinality of this set. The value  $\lambda_n$  is also known as the *n*-th Hosten-Morris number (A001206 in On-Line Encyclopedia of Integer Sequences).

The first attempt to solve the problem of determining the values of  $\lambda_n$  was made, in 1968, by Riviere [6], who determined all values up to  $\lambda_5$ . In 1972, Brouwer and Verbeek provided the values up to  $\lambda_7$  [3]. The value of  $\lambda_8$  was determined by Mills and Mills [5] in 1978.

The most recent known term,  $\lambda_9$ , was obtained by Brouwer, Mills, Mills, and Verbeek [2] in 2013. The value of  $\lambda_n$  corresponds to the number of non-dominated coteries on n members [1, Section 1]; and also corresponds to the number of maximal linked systems, see Section 2.1 below and [2, Section 1].

Any two Boolean functions are said to be *equivalent* if one can be transformed into the other by a permutation of input variables (see Section 2.3). Let  $Q_n$  denote the set of all equivalence classes in  $\Lambda_n$  and let  $q_n$  denote  $|Q_n|$ . Values of  $q_n$  are described by <u>A008840</u> OEIS sequence.

In this paper, we derive several algorithms for counting self-dual monotone Boolean functions. We also confirm the result of [2] that  $\lambda_9$  equals 423,295,099,074,735,261,880. Furthermore, employing Burnside's lemma and techniques as discussed in [8, 9], we calculate  $q_8$  to be 6,001,501, which marks the first-ever calculation of this value.

n	$\lambda_n$	$q_n$
0	0	0
1	1	1
2	2	1
3	4	2
4	12	3
5	81	7
6	2,646	30
7	1,422,564	716
8	229,809,982,112	6,001,501
9	423,295,099,074,735,261,880	_

Table 1: Known values of  $\lambda_n$  (A001206) and  $q_n$  (A008840).

### 2 Preliminaries

By  $\top$  we denote the maximal element in  $D_n$ , that is  $\top = (1 \dots 1)$ , and by  $\bot$  the minimal element in  $D_n$ , that is  $\bot = (0 \dots 0)$ . For two elements  $x, y \in D_n$ , by x|y we denote the *bitwise or*; and by x & y the *bitwise and*. Furthermore, let re(x, y) denote  $|\{z \in D_n : x \leq z \leq y\}|$ . Note that  $re(x, \top) = |\{z \in D_n : x \leq z\}|$  and  $re(\bot, y) = |\{z \in D_n : z \leq y\}|$ . For  $x \in D_n$ , by  $\ell(x)$  we denote the number of ones in x, also known as its Hamming weight. For example,  $\ell(0000) = 0$  and  $\ell(0101) = 2$ .

**Lemma 1.** For each  $x, y \in D_n$ , we have:

1.  $x^{**} = x$ 2. if  $x \le y$  then  $y^* \le x^*$ 3.  $(x|y)^* = x^* \& y^*$ 4.  $(x \& y)^* = x^* | y^*$ 

### 2.1 Maximal linked system

Let  $X = \{1, \ldots, n\}$  and  $\mathcal{P}(X)$  be the power set of X. A family  $\mathcal{W} \subseteq \mathcal{P}(X)$  is *linked* if for all A and B in  $\mathcal{W}$ ,  $A \cap B$  is not empty. A family  $\mathcal{U} \subseteq \mathcal{P}(X)$  is *maximal linked system (mls)* on X if  $\mathcal{U}$  is linked and for all  $\mathcal{W}$  with  $\mathcal{U} \subseteq \mathcal{W} \subseteq \mathcal{P}(X)$ , either  $\mathcal{W} = \mathcal{U}$  or  $\mathcal{W}$  is not linked.

If  $\mathcal{U}$  is linked, then  $\emptyset \notin \mathcal{U}$  and for each set  $A \in \mathcal{P}(X)$ , it is not possible that both A and its complement  $A^c = X - A$  belong to  $\mathcal{U}$ .

For n = 0,  $X = \emptyset$ ,  $\mathcal{P}(X) = \{\emptyset\}$  and we have one mls, namely, the empty family. Notice that the set of self-duals  $\Lambda_0$  is empty and  $\lambda_0 = 0$ .

**Lemma 2.** If  $n \ge 1$  and a family  $\mathcal{U} \subset \mathcal{P}(X)$  is an mls then:

(L1)  $\mathcal{U}$  is an upset, i.e. if  $A \subseteq B$  and  $A \in \mathcal{U}$ , then  $B \in \mathcal{U}$ .

(L2) For every subset  $A \in \mathcal{P}(X)$ , exactly one of the two subsets: A or  $A^c$ , is in  $\mathcal{U}$ .

*Proof.* (L1) For each  $C \in \mathcal{U}$ ,  $A \cap C \neq \emptyset$  and  $B \cap C \neq \emptyset$ . Hence, either  $B \in \mathcal{U}$  or  $\mathcal{U}$  is not maximal.

(L2) We have two cases:

Case 1. For every  $B \in \mathcal{U}$ ,  $A \cap B \neq \emptyset$ . Then either  $A \in \mathcal{U}$  or  $\mathcal{U}$  is not maximal.

Case 2. There is  $B \in \mathcal{U}$  such that  $B \cap A = \emptyset$  and  $B \subseteq A^c$ . Hence,  $A^c \in \mathcal{U}$ .

**Lemma 3.** If  $\mathcal{U} \subseteq \mathcal{P}(X)$  satisfies conditions (L1) and (L2), then  $\mathcal{U}$  is an mls.

*Proof.* First, we prove that  $\mathcal{U}$  is linked. Suppose, for a contradiction, that there are two subsets  $A, B \in \mathcal{U}$  with  $A \cap B = \emptyset$ . Then  $B \subseteq A^c$  and  $A^c \in \mathcal{U}$ , a contradiction.

 $\mathcal{U}$  is maximal, because it contains  $2^{n-1}$  subsets.

For n = 1, the family  $\{\{1\}\}\$  is the only mls on  $X = \{1\}$ .

For n = 2, we have two mlses:  $\{\{1\}, \{1, 2\}\}$  and  $\{\{2\}, \{1, 2\}\}$ .

Notice that  $\mathcal{P}(X)$  is isomorphic to  $B^n$  and any subset of  $\mathcal{P}(X)$  can be represented by a function from  $B^n$  to B. For n = 1, the mls  $\{\{1\}\}$  can be represented as the string 01, which is the only self-dual in  $\Lambda_1$ . For n = 2, the two mlses can be represented as 0101 and 0011 and they form the set  $\Lambda_2$ , and so on. For  $n \ge 1$ , the set of mlses on  $\{1, \ldots, n\}$  can be represented as the self-duals  $\Lambda_n$ .

### 2.2 Poset

A poset (partially ordered set)  $(S, \leq)$  consists of a set S (called the carrier) together with a binary relation (partial order)  $\leq$  which is reflexive, transitive, and antisymmetric. For example,  $B, B^n$ , and  $D_n$  are posets. Given two posets  $(S, \leq)$  and  $(T, \leq)$ , a function  $f: S \to T$ is monotone, if  $x \leq y$  implies  $f(x) \leq f(y)$ . By  $T^S$  we denote the poset of all monotone functions from S to T with the partial order defined by:

 $f \leq g$  if and only if  $f(x) \leq g(x)$  for all  $x \in S$ .

Notice that  $D_n = B^{B^n}$  and  $B^n = B^{A_n}$ , where  $A_n$  is the antichain with the carrier  $\{1, \ldots, n\}$ . In this paper we use the well known lemma:

**Lemma 4.** The poset  $D_{n+k}$  is isomorphic to the poset  $D_n^{B^k}$ —the poset of monotone functions from  $B^k$  to  $D_n$ .

### 2.3 Permutations and equivalence relation

Let  $S_n$  denote the set of permutations on  $\{1, \ldots, n\}$ . Every permutation  $\pi \in S_n$  defines the permutation on  $B^n$  by  $\pi(x) = x \circ \pi^{-1}$  (we treat each element  $x \in B^n$  as a function  $x : \{1, \ldots, n\} \to \{0, 1\}$ ). The permutation  $\pi$  also generates the permutation on  $D_n$ . Namely, by  $\pi(g) = g \circ \pi$ . By ~ we denote an equivalence relation on  $D_n$ . Namely, two functions  $f, g \in D_n$  are equivalent,  $f \sim g$ , if there is a permutation  $\pi \in S_n$  such that  $f = \pi(g)$ . For a function  $f \in D_n$  its equivalence class is the set  $[f] = \{g \in D_n : g \sim f\}$ . By  $\gamma(f)$  we denote |[f]|. For the class [f], its representative is its minimal element (according to the total order induced on  $D_n$  by the total order in integers). Sometimes, we identify the class  $[f] \in R_n$  with its representative and treat [f] as an element in  $D_n$ . By  $R_n$  we denote the set of equivalence classes and by  $r_n$  we denote the number of the equivalence classes; that is  $r_n = |R_n|$ . Let  $Q_n$ denote the set of all equivalence classes in  $\Lambda_n$ , and let  $q_n$  denote  $|Q_n|$ .

- **Lemma 5.** 1. For every element  $x \in D_n$  and every permutation  $\pi \in S_n$ , we have  $\pi(x^*) = (\pi(x))^*$ .
  - 2. If  $x \in D_n$  is a self-dual, then every equivalent  $y \in [x]$  is a self-dual.

For n = 2, we have  $d_2 = 6$  monotone functions,

$$D_2 = \{0000, 0001, 0011, 0101, 0111, 1111\}$$

and  $r_2 = 5$  equivalence classes in  $D_2$ , namely

$$R_2 = \{\{0000\}, \{0001\}, \{0011, 0101\}, \{0111\}, \{1111\}\}.$$

Furthermore, there are two self-duals  $\Lambda_2 = \{0011, 0101\}$  and they form one equivalence class. Hence  $\lambda_2 = 2$  and  $q_2 = 1$ .

For n = 3, we have  $d_3 = 20$  monotone functions and  $r_3 = 10$  equivalence classes in  $D_3$ . There are four self-duals

$$\Lambda_3 = \{01010101, 00110011, 00001111, 00010111\}$$

and they form two equivalence classes:

 $Q_3 = \{\{01010101, 00110011, 00001111\}, \{00010111\}\}.$ 

Hence,  $\lambda_3 = 4$  and  $q_3 = 2$ .

## **3** Counting functions from B to $D_n$

Let  $n \ge 0$ . By Lemma 4, the poset  $D_{n+1}$  is isomorphic to the poset  $D_n^B$ —the poset of monotone functions from  $B = \{0, 1\}$  to  $D_n$ . Consider a monotone function  $H : B \to D_n$ . It can be represented as the concatenation:

$$H = H(0) \cdot H(1)$$

with  $H(0), H(1) \in D_n$  and  $H(0) \leq H(1)$ . The dual of H is

$$H^* = H(1)^* \cdot H(0)^*.$$

Thus, if  $H \in D_{n+1}$  is a self-dual then it is of the form  $b \cdot b^*$  with  $b \in D_n$  and  $b \leq b^*$ . And vice versa, if  $b \in D_n$  and  $b \leq b^*$ , then the concatenation  $b \cdot b^*$  is a self-dual in  $D_{n+1}$ . Therefore, we have proved the following theorem.

**Theorem 6.** For every  $n \ge 0$ , the number of self-duals  $\lambda_{n+1}$  is equal to the number of elements  $b \in D_n$  that satisfy condition  $b \le b^*$ . In other words

$$\lambda_{n+1} = \sum_{\substack{b \in D_n \\ b \le b^*}} 1.$$

Furthermore, Lemma 5 implies

$$\lambda_{n+1} = \sum_{\substack{b \in R_n \\ b \le b^*}} \gamma(b)$$

Here we identify each class  $[b] \in R_n$  with its representative.

The following corollary is presented, in a different form, in [10]

**Corollary 7.** For every  $n \ge 0$ , we have  $\lambda_{n+1} \le d_n$ .

**Lemma 8.** Let b be a function in  $D_n$ . Then

- $b \leq b^*$ , only if  $\ell(b) \leq 2^{n-1}$ .
- if  $\ell(b) = 2^{n-1}$  and  $b \leq b^*$ , then  $b = b^*$ , and b is a self-dual in  $D_n$ .
- if b is a self-dual, then  $\ell(b) = 2^{n-1}$
- $\ell(b^*) = 2^n \ell(b)$ , hence,  $|\{x \in D_n : \ell(x) < d_{n-1}\}| = |\{x \in D_n : \ell(x) > d_{n-1}\}|$

As a corollary we have the following theorem.

Theorem 9.

$$\lambda_{n+1} = \lambda_n + \sum_{\substack{b \in R_n \\ \ell(b) < 2^{n-1} \\ b \le b^*}} \gamma(b).$$
$$\lambda_{n+1} \le \lambda_n + \frac{1}{2}(d_n - \lambda_n) = \frac{1}{2}(d_n + \lambda_n).$$

The second part of the theorem is presented, in a different form, in [10]. Notice that  $\lambda_{n+1} = \frac{1}{2}(d_n + \lambda_n)$  for each  $n \leq 3$ .

# 4 Counting functions from $B^2$ to $D_n$

Let  $n \ge 0$ . By Lemma 4, the poset  $D_{n+2}$  is isomorphic to the poset  $D_n^{B^2}$ —the poset of monotone functions from  $B^2 = \{00, 01, 10, 11\}$  to  $D_n$ . Consider a monotone function  $H : B^2 \to D_n$ . It can be represented as the concatenation:

$$H(00) \cdot H(01) \cdot H(10) \cdot H(11)$$

and its dual as

$$H(11)^* \cdot H(10)^* \cdot H(01)^* \cdot H(00)^*.$$

If H is a self-dual then  $H(00) = H(11)^*$ ,  $H(01) = H(10)^*$ ,  $H(10) = H(01)^*$ , and  $H(11) = H(00)^*$ .



Figure 1: Structure of  $H: B^2 \to D_n$  if H is a self-dual.

If *H* is a self-dual then it is of the form  $d^*bb^*d$  with  $b, d \in D_n$  and  $d \ge b|b^*$ , see Figure 1. And vice versa, if  $b, d \in D_n$ , and  $d \ge b|b^*$ , then the concatenation  $d^* \cdot b \cdot b^* \cdot d$  is a self-dual in  $D_{n+2}$ . Hence, we have proved the following theorem.

**Theorem 10.** For every  $n \ge 0$ , the number of self-duals  $\lambda_{n+2}$  is equal to the number of pairs  $b, d \in D_n$  which satisfy condition  $d \ge b|b^*$ . In other words

$$\lambda_{n+2} = \sum_{b \in D_n} re[b|b^*, \top].$$

Furthermore, Lemma 5 implies

$$\lambda_{n+2} = \sum_{b \in R_n} \gamma(b) \cdot re[b|b^*, \top].$$

Here we identify each class  $[b] \in R_n$  with its representative.

**Corollary 11.** For every  $n \ge 0$ , we have  $\lambda_{n+2} \ge d_n$ .

# 5 Counting functions from $B^4$ to $D_n$

Let  $n \ge 0$ . By Lemma 4, the poset  $D_{n+4}$  is isomorphic to the poset  $D_n^{B^4}$ —the set of monotone functions from

 $B^4 = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$  to  $D_n$ .

Consider a monotone function  $H: B^4 \to D_n$ . It can be represented as the concatenation:

 $H(0000) \cdot H(0001) \cdot H(0010) \cdot H(0011) \cdot H(0100) \cdot H(0101) \cdot H(0110) \cdot H(0111) \cdot$ 

 $H(1000) \cdot H(1001) \cdot H(1010) \cdot H(1011) \cdot H(1100) \cdot H(1101) \cdot H(1110) \cdot H(1111)$  and its dual as

$$\begin{split} H(1111)^* \cdot H(1110)^* \cdot H(1101)^* \cdot H(1100)^* \cdot H(1011)^* \cdot H(1010)^* \cdot H(1001)^* \cdot H(1000)^* \cdot \\ H(0111)^* \cdot H(0110)^* \cdot H(0101)^* \cdot H(0100)^* \cdot H(0011)^* \cdot H(0010)^* \cdot H(0001)^* \cdot H(0000)^*. \\ \text{If $H$ is a self-dual then $H(0000) = H(1111)^*$, $H(0001) = H(1110)^*$, $H(0010) = H(1101)^*$, $H(0011) = H(1100)^*$, $H(0100) = H(1011)^*$, $H(0101) = H(1010)^*$, $H(0110) = H(1001)^*$, $H(0111) = H(1000)^*$. \end{split}$$

**Theorem 12.** For each  $a, b, c \in D_n$ , for each  $h \in D_n$  such that  $h \ge a|b|c|a^*|b^*|c^*$ , for each  $d, e, f, g \in D_n$  such that  $a|b|c \le d \le h$  $a|b^*|c^* \le e \le h$  $b|a^*|c^* \le f \le h$  $c|a^*|b^* \le g \le h$ the concatenation

$$h^* \cdot g^* \cdot f^* \cdot a \cdot e^* \cdot b \cdot c \cdot d \cdot d^* \cdot c^* \cdot b^* \cdot e \cdot a^* \cdot f \cdot g \cdot h$$

represents a self-dual in  $D_{n+4}$ , see Figure 2. And vice versa, each self-dual in  $D_{n+4}$  is of the above form.



Figure 2: Structure of  $H: B^4 \to D_n$  if H is a self-dual.

**Theorem 13.** The number of self-duals

$$\lambda_{n+4} = \sum_{\substack{a,b,c\in D_n\\h\ge (a|b|c|a^*|b^*|c^*)}} re[a|b|c,h] \cdot re[a|b^*|c^*,h] \cdot re[b|a^*|c^*,h] \cdot re[c|a^*|b^*,h] \cdot$$

**Theorem 14.** For each  $h \in D_n$  such that  $h \ge h^*$ , for each  $a, b, c \in D_n$ ,  $h^* \le a, b, c \le h$ for each  $d, e, f, g \in D_n$  such that  $a|b|c \le d \le h$  $a|b^*|c^* \le e \le h$  $b|a^*|c^* \le f \le h$  $c|a^*|b^* \le g \le h$ the concatenation

$$h^* \cdot g^* \cdot f^* \cdot a \cdot e^* \cdot b \cdot c \cdot d \cdot d^* \cdot c^* \cdot b^* \cdot e \cdot a^* \cdot f \cdot g \cdot h$$

represents a self-dual in  $D_{n+4}$ ; see Figure 2. And vice versa, each self-dual in  $D_{n+4}$  is of the above form.

Let

$$F(h) = \sum_{\substack{a,b,c \in D_n \\ h \ge a,b,c \ge h^*}} re[a|b|c,h] \cdot re[a|b^*|c^*,h] \cdot re[b|a^*|c^*,h] \cdot re[c|a^*|b^*,h].$$

Observe that F(h) is the number of self-dual functions  $H \in D_n^{B^4}$ , with  $H(0000) = h^*$ and H(1111) = h.

$$\lambda_{n+4} = \sum_{\substack{h \in D_n \\ h^* \le h}} F(h)$$

Furthermore, Lemma 5 implies that for any two elements  $h_1 \sim h_2$  we have  $F(h_1) = F(h_2)$ . Hence, we have

$$\lambda_{n+4} = \sum_{\substack{h \in R_n \\ h \ge h^*}} \gamma(h) \cdot F(h).$$

Here again we identify the class  $[h] \in R_n$  with its representative. Similarly as in Lemma 8 we can observe that  $h \ge h^*$ , only if  $\ell(h) \ge 2^{n-1}$ . Furthermore, if  $\ell(h) = 2^{n-1}$  and  $h \ge h^*$ , then  $h = h^*$ , and we have only one self-dual function  $H \in D_n^{B^4}$  with H(0000) = H(1111) = h. Hence,

$$\lambda_{n+4} = \lambda_n + \sum_{\substack{h \in R_n \\ h^* \le h \\ \ell(h) > 2^{n-1}}} \gamma(h) \cdot F(h)$$
(1)

### 6 Implementation

In this section we present three algorithms based on the results from the previous sections. We implemented the algorithms in Rust and ran them on a 32-thread Xeon CPU.

```
Algorithm 1 Calculation of \lambda_{n+2}Input: R_n with re[x, \top] for all x \in R_nOutput: s = \lambda_{n+2}1: Initialize s = 0,2: for all b \in R_n do3: s = s + re[b|b^*, \top] \cdot \gamma(b)4: end for
```

Algorithm 1 is based on Theorem 6. After loading the preprocessed data into main memory,  $\lambda_9$  was computed in 15 seconds. However, preprocessing (the calculation of  $R_7$  and its intervals) took approximately 2,5 hours.

Algorithm 2 Calculation of  $\lambda_{n+4}$ 

**Input:**  $D_n$ ;  $R_n$ ; re[x, y] for all  $(x, y) \in D_n \times D_n$ **Output:**  $s = \lambda_{n+4}$ 1: Initialize s = 0, 2: for all  $a \in R_n$  do for all  $b \in D_n$  do 3: for all  $c \in D_n$  do 4: for all  $h \in D_n, h \ge (a|b|c|a^*|b^*|c^*)$  do 5: $s = s + re[a|b|c, h] \cdot re[a|b^*|c^*, h] \cdot re[b|a^*|c^*, h] \cdot re[c|a^*|b^*, h] \cdot \gamma(a)$ 6: end for 7:end for 8: end for 9: 10: **end for** 

Algorithm 2 is based on Theorem 13. Using our implementation of the algorithm, we calculated  $\lambda_9$  in 76 seconds, and the preprocessing was almost instantaneous.

**Algorithm 3** Calculation of  $\lambda_{n+4}$ 

**Input:**  $D_n$ ;  $R_n$ ; re[x, y] for all  $(x, y) \in D_n \times D_n$ Output:  $s = \lambda_{n+4}$ 1: Initialize  $s = \lambda_n$ , 2: for all  $h \in R_n, h^* \le h, \ell(h) > 2^{n-1}$  do for all  $a \in D_n, a < h$  do 3: for all  $b \in D_n, b < h$  do 4: for all  $c \in D_n, c < h$  do 5:  $s = s + re[a|b|c, h] \cdot re[a|b^*|c^*, h] \cdot re[b|a^*|c^*, h] \cdot re[c|a^*|b^*, h] \cdot \gamma(h)$ 6: end for 7: end for 8: end for 9: 10: **end for** 

Algorithm 3 is based on Equation 1. The calculation of  $\lambda_9$  using our implementation of the algorithm lasted approximately 25 minutes.

In all cases, we have obtained the following value:

 $\lambda_9 = 423295099074735261880,$ 

which confirms the result of Brouwer et al. [2].

## 7 Calculation of $q_n$

The number of inequivalent self-dual monotone Boolean functions  $(q_n)$  is listed on the OEIS <u>A008840</u> sequence. In order to calculate  $q_n$  for  $n \leq 7$ , we can use the following simple approach:

$$q_n = \sum_{\substack{a \in R_n \\ a = a^*}} 1.$$

However, the calculating of  $q_8$  using this approach is beyond reach. In order to calculate  $q_8$  we use Burnside's lemma, see [8, 9]:

$$q_n = \frac{1}{n!} \sum_{i=1}^k \mu_i |\Phi(\pi_i, \Lambda_n)| \tag{2}$$

where:

- $q_n$  is the number of equivalence classes in  $\Lambda_n$
- $\Phi(\pi_i, \Lambda_n)$  is the set of all elements in  $\Lambda_n$  which are fixed under  $\pi_i$

- k is the number of different cycle types in  $S_n$
- *i* is index of cycle type
- $\mu_i$  is the number of permutations in  $S_n$  with cycle type *i*
- $\pi_i$  is a permutation in  $S_n$  of the cycle type i

For n = 1, we have  $\lambda_1 = q_1 = 1$ . For n = 2, we have two permutations: the identity e with  $|\Phi(e, \Lambda_2)| = |\Lambda_2| = 2$ , and the inversion (12) with three cycles when acting on  $B^2$  namely:  $C_1 = (00), C_2 = (01, 10), \text{ and } C_3 = (11)$ . Two elements 01 and 10 form a cycle, hence, if a function  $f \in D_n$  is a fix of  $\pi$ , then f(01) = f(10). On the other hand, if f is a self dual, then it represents an mls on  $\{1, 2\}$ , and  $f(01) \neq f(10)$ , because 01 and 10 represent subsets  $\{1\}$  and  $\{2\}$  in  $\{1, 2\}$  which are the complements of each other. Thus, the set of fixes  $\Phi((12), \Lambda_2) = \emptyset$ . By Burnside's lemma, we have:

$$q_2 = \frac{1}{2}(|\Phi(e, \Lambda_2)| + |\Phi((12), \Lambda_2)|) = \frac{1}{2}(2+0) = 1.$$

Indeed, there is one equivalence class in  $\Lambda_2$ , namely {0101, 0011}.

We have just shown that  $\Phi((12), \Lambda_2) = \emptyset$ . Similarly, we can show that  $\Phi((\pi), \Lambda_8) = \emptyset$ , if the permutation  $\pi = (12)(34)(56)(78)$ . Indeed, for the permutations  $\pi$ , two elements 01010101, 10101010  $\in B^8$  form a cycle. Hence, if  $f \in D_8$  is a fix of  $\pi$ , then f(01010101) =f(10101010). On the other side, if f is a self-dual, then  $f(01010101) \neq f(10101010)$ , because 101010101 and 01010101 represent subsets of  $\{1, \ldots, 8\}$  which are the complements of each other.

**Lemma 15.** Suppose that n is even, and a permutation  $\pi$ , when acting on  $\{1, \ldots, n\}$ , is a product of disjoint cycles of even length. Then  $\Phi((\pi), \Lambda_n) = \emptyset$ .

*Proof.* If the permutation  $\pi$ , is a product of disjoint cycles of even length, then there exist two elements  $x, y \in B^n$  such that:

- x and y represent subsets of  $\{1, \ldots, n\}$  which are the complements of each other.
- $\pi(x) = y$  and  $\pi(y) = x$ , so x, y form a cycle in  $B^n$ .

Hence, if  $f \in D_n$  is a fix of  $\pi$ , then f(x) = f(y). On the other side, if f is a self-dual, then  $f(x) \neq f(y)$ .

Corollary 16.  $\Phi((\pi), \Lambda_8) = \emptyset$  for each of the following permutations: (12345678), (12)(345678), (1234)(5678), (12)(34)(5678), and (12)(34)(56)(78).

For n = 3, we have three cycles types:

• the identity e with  $|\Phi(e, \Lambda_3)| = |\Lambda_3| = 4$ 

- three inversions, with  $|\Phi((12), \Lambda_3)| = 2$
- two cycles of length 3 with  $|\Phi((123), \Lambda_3)| = 1$

By Burnside's lemma, we have:

$$q_3 = \frac{1}{6}(4 + 3 \cdot 2 + 2 \cdot 1) = 2.$$

Notice, that the element MAJ = 00010111  $\in D_3$  is a self-dual, and is a fix for every permutation  $\pi \in S_3$ . Hence, for every  $\pi \in S_3$ ,  $\Phi(\pi, \Lambda_3) \neq \emptyset$ . Similarly, we can show the following lemma:

**Lemma 17.** For each odd n and each permutation  $\pi \in S_n$ , we have  $\Phi(\pi, \Lambda_n) \neq \emptyset$ .

*Proof.* Consider the function MAJ  $\in D_n$ , which returns MAJ(x) = 1 if and only if  $\ell(x) > n/2$ . The function MAJ is a self-dual, and is a fix for every permutation  $\pi \in S_n$ .

## 7.1 Algorithms counting fixed points in $\Lambda_n$

In order to count or generate fixes of permutations in  $\Lambda_n$  we use Lemma 15 and two algorithms.

### Algorithm 4 Generation of $\Phi(\pi, \Lambda_n)$

```
Input: \Phi(\pi, D_n)

Output: S = \Phi(\pi, \Lambda_n)

1: Initialize S

2: for all b \in \Phi(\pi, D_n) do

3: if b = b^* then

4: Add b to S

5: end if

6: end for
```

First algorithm simply runs through the set of fixes  $\Phi(\pi, D_n)$  and selects self-duals. For example, there are five fixes in

 $\Phi((123), D_3) = \{0000000, 0000001, 00010111, 01111111, 1111111\}$ and only one of them is self-dual, namely 00010111, so  $|\Phi((123), \Lambda_3)| = 1$ . Algorithm 5 Calculation of  $|\Phi(\pi, \Lambda_{n+2})|$ 

Input:  $\Phi(\pi, D_n)$ Output:  $s = |\Phi(\pi, \Lambda_{n+2})|$ 1: Initialize s = 0, 2: for all  $b \in \Phi(\pi, D_n)$  do 3: Calculate  $up = |\{h \in \Phi(\pi, D_n) : h \ge (b \cup b^*)\}|$ 4: s = s + up5: end for

Second algorithm is based on the following facts. Consider a permutation  $\pi$  acting on  $B^n$  and on  $D_n$ . We can say that  $\pi$  also act on  $B^{n+2}$  and on  $D_{n+2}$ . By [9, Lemma 6],  $\Phi(\pi, D_{n+2}) = \Phi(\pi, D_n)^{B^2}$ . Every function  $F \in \Phi(\pi, D_n)^{B^2}$  can be represented as the concatenation

 $F = F(00) \cdot F(01) \cdot F(10) \cdot F(11),$ 

where F(00), F(01), F(10),  $F(11) \in \Phi(\pi, D_n)$ , and

$$F(00) \le F(01), F(10) \le F(11).$$

The dual of F can be represented as

$$F^* = F(11)^* \cdot F(10)^* \cdot F(01)^* \cdot F(00)^*.$$

If F is a self dual, then it is of the form

 $dbb^*d^*$ ,

where  $b, d \in \Phi(\pi, D_n)$  and  $d^* \ge b|b^*$ . Notice that this implies that  $d \le b\&b^*$ . On the other side, if  $b, d \in \Phi(\pi, D_n)$  and  $d^* \ge b|b^*$ , then  $dbb^*d^* \in \Phi(\pi, \Lambda_{n+2})$ .

### 7.2 Result tables

In this section we present three tables which contain the values of  $|\Phi(\pi_i, \Lambda_n)|$ , for  $n \in \{6, 7, 8\}$  and all permutations.

i	$\pi_i$	$\mu_i$	$ \Phi(\pi_i, \Lambda_6) $
1	(1)	1	2646
2	(12)	15	372
3	(123)	40	54
4	(1234)	90	18
5	(12345)	144	6
6	(123456)	120	0
7	(12)(34)	45	130
8	(12)(345)	120	18
9	(12)(3456)	90	0
10	(123)(456)	40	18
11	(12)(34)(56)	15	0

$$q_6 = \frac{1}{720} \sum_{i=1}^{11} \mu_i \cdot |\Phi(\pi_i, \Lambda_6)| = \frac{21600}{720} = 30$$

i	$\pi_i$	$\mu_i$	$ \Phi(\pi_i, \Lambda_7) $		
1	(1)	1	1422564		
2	(12)	21	43556		
3	(123)	70	1332		
4	(1234)	210	216		
5	(12345)	504	34		
6	(123456)	840	12		
7	(1234567)	720	3		
8	(12)(34)	105	7212		
9	(12)(345)	420	218		
10	(12)(3456)	630	76		
11	(12)(34567)	504	6		
12	(123)(456)	280	210		
13	(123)(4567)	420	6		
14	(12)(34)(56)	105	1284		
15	(12)(34)(567)	210	36		
$q_7 = \frac{1}{5040} \sum_{i=1}^{15} \mu_i \cdot  \Phi(\pi_i, \Lambda_7)  = \frac{3608640}{5040} = 716$					

	i	$\pi_i$	$\mu_i$	$ \Phi(\pi_i, \Lambda_8) $	
_	1	(1)	1	229809982112	-
	2	(12)	28	300991356	
	3	(123)	112	476120	
	4	(1234)	420	18984	
	5	(12345)	1344	662	
	6	(123456)	3360	296	
	7	(1234567)	5760	46	
	8	(12345678)	5040	0	
	9	(12)(34)	210	12716048	
	10	(12)(345)	1120	18384	
	11	(12)(3456)	2520	7952	
	12	(12)(34567)	4032	116	
	13	(12)(345678)	3360	0	
	14	(123)(456)	1120	21020	
	15	(123)(4567)	3360	120	
	16	(123)(45678)	2688	20	
	17	(1234)(5678)	1260	0	
	18	(12)(34)(56)	420	2230724	
	19	(12)(34)(567)	1680	3152	
	20	(12)(34)(5678)	1260	0	
	21	(12)(345)(678)	1120	1488	
	22	(12)(34)(56)(78)	105	0	_
$a_{\mathbf{s}} = -$	1	$\sum_{i=1}^{22} \mu_i \cdot  \Phi(\pi_i, \Lambda_{\mathfrak{s}}) $	$  = \frac{241}{2}$	$\frac{1980137280}{1980137280} = 60$	01501
40	)320	$\sum_{i=1}^{n} r^{i} r^{i} + 2(r^{i}, r^{i}, r^{i})$	I	40320	

## References

- J. C. Bioch and T. Ibaraki. Generating and approximating nondominated coteries in IEEE Trans. Parallel Distrib. Syst., 6 (1995)
- [2] A. E. Brouwer, C. F. Mills, W. H. Mills, and A. Verbeek. Counting families of mutually intersecting sets in *Electron. J. Comb.*, 2013, Volume 20, Issue 2
- [3] A. E. Brouwer and A. Verbeek. Counting families of mutually intersecting sets, Report ZN 41, March 1972, Math. Centr., Amsterdam.
- [4] D. E. Loeb, A. R. Conway. Voting Fairly: Transitive Maximal Intersecting Families of Sets, Journal of Combinatorial Theory, Series A, 91 (2000)
- [5] C. F. Mills and W. M. Mills. The calculation of  $\lambda(8)$ , preprint (1979).
- [6] N. M. Riviere, Recursive formulas on free distributive lattices, J. Comb. Theory 5 (1968).
- [7] N. J. A. Sloane. The Online Encyclopedia of Integer Sequences https://oeis.org
- [8] B. Pawelski, On the number of inequivalent monotone Boolean functions of 8 variables, J. Integer Sequences 25 (2022), Article 25.7.7.
- [9] A. Szepietowski, Fixes of permutations acting on monotone Boolean functions, J. Integer Sequences 25 (2022), Article 25.9.6.
- [10] M. Timotijević. Note on combinatorial structure of self-dual simplicial complexes, Matematički Vesnik, 71 (2019)

2020 Mathematics Subject Classification: Primary 06E30.

*Keywords*: Monotone Boolean functions, Self-dual monotone Boolean functions, Dedekind numbers, Hosten-Morris numbers

(Concerned with sequences  $\underline{A001206}$  and  $\underline{A008840}$ .)