Journal of Integer Sequences, Vol. 28 (2025),
Article 25.3.4

Counting Unate and Monotone Boolean Functions Under Restrictions of Balancedness and Non-Degeneracy

Aniruddha Biswas and Palash Sarkar
Indian Statistical Institute
203, B.T.Road
Kolkata 700108
India
aniruddhabiswas535@gmail.com
palash@isical.ac.in

Abstract

We consider the problem of counting the numbers of functions in various sub-classes of unate and monotone Boolean functions under the restrictions of balancedness and non-degeneracy. Further, we also consider the problem of counting the numbers of inequivalent and NPN-inequivalent functions in these sub-classes.

1 Introduction

For a positive integer n, an n-variable Boolean function f is a map $f : \{0,1\}^n \to \{0,1\}$. A Boolean function f is said to be monotone increasing (resp., decreasing) in the i-th variable if

$$f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \leq f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

$$(\text{resp., } f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \geq f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n))$$

for all possible $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \in \{0,1\}$.

The function f is said to be *locally monotone* or *unate*, if for each $i \in \{1, \ldots, n\}$, it is either monotone increasing or monotone decreasing in the i-th variable. The function

f is said to be *monotone increasing* (or, simply *monotone*) if for each $i \in \{1, \ldots, n\}$, it is monotone increasing in the i-th variable.

A Boolean function is *degenerate* on some variable if its output does not depend on the variable, and it is said to be *non-degenerate* if it is not degenerate on any of the variables. A Boolean function is said to be *balanced* if it takes the values 0 and 1 equal number of times. The number of n-variable balanced Boolean functions is $\binom{2^n}{2^{n-1}}$ (see A037293).

The number of n-variable monotone Boolean functions, denoted as $D(n)$, is said to be the n-th *Dedekind number* after Dedekind [7] who posed the problem of counting the number of n-variable monotone Boolean functions in 1897. The value of $D(n)$ is known for $n \leq 9$ (see [16, 7, 5, 18, 4, 19, 8, 11, 10] and A000372). Kisielewicz [12] gave a closed form summation formula for $D(n)$. However, Korshunov [13] pointed out that using the formula to compute $D(n)$ has the same complexity as direct enumeration of all n-variable monotone Boolean functions. The inverse binomial transform of $D(n)$ gives the number of n-variable non-degenerate monotone Boolean functions and hence the latter quantities are also known for $n \leq 9$ (see A006126).

Two Boolean functions on the same number of variables are said to be *equivalent* if one can be obtained from the other by a permutation of variables. Two Boolean functions on the same number of variables are said to be *NPN-equivalent* if one can be obtained from the other by a combination of the following operations: a permutation of the variables, negation of a subset of the variables, and negation of the output. Both of the above two notions of equivalence partition the set of all n-variable Boolean functions into equivalence classes. The number of equivalence classes is also said to be the number of inequivalent functions under the appropriate notion of equivalence. It is of interest to count the number of inequivalent Boolean functions satisfying some prescribed properties. For example, the number of inequivalent n-variable monotone Boolean functions is known for n up to 9 (see A003182, Stephen and Yusun [17] and Pawelski [14]). Another example is the number of n-variable NPN-inequivalent unate functions (see A003183 and Baugh [2]).

The focus of the present work is on counting unate and monotone Boolean functions under various restrictions. For $n \leq 5$, it is possible to enumerate all n-variable Boolean functions. Consequently, the problem of counting various sub-classes of n-variable Boolean functions become a reasonably simple problem. Non-triviality of counting Boolean functions arises for $n \geq 6$.

The problem of counting unate functions reduces to the problem of counting monotone Boolean functions. This follows from the work of Baumann and Strass [3]. (We were unaware of the work of Baumann and Strass when we obtained the relation between the numbers of unate and monotone Boolean functions.) Since the numbers of n-variable monotone Boolean functions are known for $n \leq 9$, these values provide the numbers of n-variable unate functions for $n \leq 9$. The numbers of n-variable unate functions are available from A245079. We updated A245079 by providing the value for $n = 9$ and correcting the values for $n = 7$ and $n = 8$.

We show that the problem of counting balanced unate functions reduces to the problem of counting balanced monotone Boolean functions. The number of n-variable balanced monotone Boolean functions is known for $n \leq 7$ (see A341633 and Church [6]). Consequently, we obtain the numbers of n-variable balanced unate functions for $n \leq 7$. We further extend these results to obtain the numbers of non-degenerate balanced monotone Boolean functions, non-degenerate unate functions, and non-degenerate balanced unate functions.

Unlike the situation for counting functions, the problem of counting the number of inequivalent unate functions does not reduce to the problem of counting the number of inequivalent monotone Boolean functions. So to count the number of inequivalent unate functions, we used a method to generate all n-variable unate functions and applied a filtering to the obtained set. This allowed us to obtain the numbers of inequivalent n-variable unate and balanced unate functions for $n \leq 6$. Moreover, we obtain the numbers of inequivalent n-variable balanced monotone Boolean functions, non-degenerate balanced monotone Boolean functions, non-degenerate unate functions, and non-degenerate balanced unate functions for $n \leq 6$.

The results that we present for monotone Boolean functions and unate functions are summarized in Table 1. For each entry of the table, we provide the corresponding sequence number of the On-Line Encyclopedia of Integer Sequences (OEIS) [16]. The word "New" in the column entitled "Comment" indicates that the sequence was added to OEIS based on the present paper, while the word "Updated" indicates that the existing entry was updated based on the present paper.

1.1 Outline of the paper

In Section 2 we describe the preliminaries and prove the mathematical results required to obtain the various counts. In Section 3 we address the problem of counting various subclasses of monotone Boolean functions and unate functions and in Section 4 we take up the problem of counting the numbers of inequivalent unate and monotone Boolean functions possessing a combination of several properties. Finally, Section 5 discusses some possible future research problems.

2 Mathematical results

We fix some terminology and the notation. The cardinality of a finite set S will be denoted by $\#S$. For $x, y \in \{0,1\}$, by xy and $x \oplus y$ we will denote the AND and XOR operations respectively, and by $\overline{x}$ we will denote the complement (or negation) of x.

Elements of $\{0,1\}^n$, for $n \geq 2$, are n-bit strings (or vectors) and are denoted using bold font. Given $n \geq 2$ and $1 \leq i \leq n$, let $\mathbf{e}_i$ denote the n-bit string whose i-th bit is 1 and is 0 elsewhere.

Let f be an n-variable Boolean function. The *weight* $\mathrm{wt}(f)$ of f is the size of its support, i.e., $\mathrm{wt}(f) = \#\{\mathbf{x} : f(\mathbf{x}) = 1\}$. We can uniquely represent an n-variable Boolean function f by a binary string of length 2^n in the following manner: for $0 \leq i < 2^n$, the i-th bit of the string is the value of f on the n-bit binary representation of i. We use the same notation

f to denote the string representation of f. So $f_0 \cdots f_{2^n-1}$ is the bit string of length 2^n that represents f.

Property	**Description**	**OEIS Number**	**Comment**
Monotone Boolean Functions	Number of n-variable non-degenerate balanced monotone Boolean functions for $n \leq 7$	A371722	New
	Number of equivalence classes of n-variable balanced monotone Boolean functions for $n \leq 7$	A371717	New
	Number of equivalence classes of n-variable non-degenerate balanced monotone Boolean functions for $n \leq 7$	A371718	New
	Number of NPN-equivalence classes of n-variable balanced monotone Boolean functions for $n \leq 6$	A378300	New
Unate Functions	Number of n-variable balanced unate functions for $n \leq 7$	A373690	New
	Number of n-variable non-degenerate balanced unate functions for $n \leq 7$	A373697	New
	Number of inequivalent n-variable unate functions for $n \leq 6$	A372495	New
	Number of inequivalent n-variable non-degenerate unate functions for $n \leq 6$	A374399	New
	Number of inequivalent n-variable balanced unate functions for $n \leq 6$	A374400	New
	Number of inequivalent n-variable non-degenerate balanced unate functions for $n \leq 6$	A374401	New
	Number of n-variable unate functions for $n = 7, 8, 9$	A245079	Updated
	Number of non-degenerate n-variable unate Boolean functions for $n = 9$	A305000	Updated
Misc.	Number of non-degenerate n-variable balanced Boolean functions	A378302	New
	Number of labeled antichains of finite sets spanning $n = 9$ vertices with singleton edges allowed (obtained as an inverse binomial transform of A305000).	A304999	Updated

Table 1: Summary of new and updated sequences obtained in this work.

Let $\overline{f}$ denote the negation of f, i.e., $\overline{f}(\mathbf{x}) = 1$ if and only if $f(\mathbf{x}) = 0$. Let f^r be a Boolean function defined as $f^r(x_1, \ldots, x_n) = f(\overline{x}_1, \ldots, \overline{x}_n)$. The bit string representation of f^r is the reverse of the bit string representation of f.

Let g and h be two n-variable Boolean functions having string representations $g_0 \cdots g_{2^n-1}$ and $h_0 \cdots h_{2^n-1}$. We write $g \leq h$ if $g_i \leq h_i$ for $i = 0, \ldots, 2^n - 1$. From g and h, it is possible to construct an $(n+1)$-variable function f whose string representation is obtained by concatenating the string representations of g and h. We denote this construction as $f = g||h$. For $(x_1, \ldots, x_{n+1}) \in \{0,1\}^{n+1}$, we have

$$f(x_1, \ldots, x_{n+1}) = \overline{x}_1 g(x_2, \ldots, x_{n+1}) \oplus x_1 h(x_2, \ldots, x_{n+1}). \tag{1}$$

For $n \geq 1$, an n-variable Boolean function f is said to be *non-degenerate* on the i-th variable, for $1 \leq i \leq n$, if there is an $\boldsymbol{\alpha} \in \{0,1\}^n$ such that $f(\boldsymbol{\alpha}) \neq f(\boldsymbol{\alpha} \oplus \mathbf{e}_i)$. The function f is said to be non-degenerate if it is non-degenerate on all the n variables.

By a property $\mathcal{P}$ of the set of all Boolean functions, we mean a subset of the set of all Boolean functions. For example, the property $\mathcal{P}$ could be the property of being balanced, being monotone, being unate, being non-degenerate, or a combination of these properties, where a combination of properties is given by the intersection of the corresponding subsets of Boolean functions. For $n \geq 0$, let P_n denote the number of n-variable Boolean functions possessing the property $\mathcal{P}$, and let nd-P_n denote the number of n-variable non-degenerate Boolean functions possessing the property $\mathcal{P}$. Since an n-variable function can be non-degenerate on i variables for some $i \in \{0, \ldots, n\}$ and the i variables can be chosen from the n variables in $\binom{n}{i}$ ways, we obtain the following result, which shows that the sequence $\{P_n\}_{n\geq 0}$ is given by the binomial transform of the sequence $\{\text{nd-}P_n\}_{n\geq 0}$.

Proposition 1. *For every property $\mathcal{P}$ of Boolean functions,*

$$P_n = \sum_{i=0}^{n} \binom{n}{i} \text{nd-}P_i. \tag{2}$$

Consequently,

$$\text{nd-}P_n = \sum_{i=0}^{n} (-1)^{n-i} \binom{n}{i} P_i. \tag{3}$$

For $n \geq 0$, let $A_n = 2^{2^n}$ be the number of all n-variable Boolean functions, and let $B_n = \binom{2^n}{2^{n-1}}$ be the number of n-variable balanced Boolean functions. Let nd-A_n be the number of all non-degenerate n-variable Boolean functions, and nd-B_n be the number of all non-degenerate n-variable balanced Boolean functions. Using Proposition 1, we obtain

$$\text{nd-}A_n = \sum_{i=0}^{n} (-1)^{n-i} \binom{n}{i} \cdot 2^{2^i} \quad \text{and} \quad \text{nd-}B_n = \sum_{i=0}^{n} (-1)^{n-i} \binom{n}{i} \cdot \binom{2^i}{2^{i-1}}$$

.

The entry for the number of non-degenerate n-variable Boolean functions is already in OEIS (A000618). Using the above formula, we obtain the number of nondegenerate balanced Boolean functions of n variables (see Table 2).

n	nd-B_n
0	0
1	2
2	2
3	58
4	12618
5	601016690
6	1832624137336299922
7	23951146041928082853307218802404658090
8	576865882344920633808974835786228688754860253363973736973066534096620726 7034

Table 2: Number of non-degenerate n-variable balanced Boolean functions for $0 \leq n \leq 8$ [A378302].

Remark 2. Baumann and Strass [3] (see Proposition 10 and Corollary 11) proved the special case of Proposition 1 where $\mathcal{P}$ is the property of being monotone.

For $n \geq 0$, let M_n, BM_n, U_n, and BU_n, denote the numbers of n-variable monotone, balanced-monotone, unate, and balanced-unate Boolean functions respectively, and further let nd-M_n, nd-BM_n, nd-U_n, and nd-BU_n, denote the corresponding numbers of non-degenerate functions. The relations between the number of n-variable functions possessing one of these properties and the number of non-degenerate n-variable functions possessing the corresponding property are obtained from Proposition 1. Note that M_n is the n-th Dedekind number $D(n)$.

The following result relates the numbers of monotone and unate Boolean functions.

Proposition 3. *For $n \geq 0$, the following holds.*

$$\text{nd-}U_n = 2^n \cdot \text{nd-}M_n, \tag{4}$$

$$\text{nd-BU}_n = 2^n \cdot \text{nd-BM}_n, \tag{5}$$

$$U_n \leq 2^n \cdot M_n, \tag{6}$$

$$\mathrm{BU}_n \leq 2^n \cdot \mathrm{BM}_n\,. \tag{7}$$

Proof. First we consider (4) and (5). We prove (4), the proof of (5) being similar.

Let f be an n-variable monotone Boolean function. Then it is easy to see that for every $\boldsymbol{\alpha} \in \{0,1\}^n$, the n-variable function $f_{\boldsymbol{\alpha}}$ is unate, where $f_{\boldsymbol{\alpha}}$ is defined as $f_{\boldsymbol{\alpha}}(\mathbf{x}) = f(\mathbf{x} \oplus \boldsymbol{\alpha})$ for all $\mathbf{x} \in \{0,1\}^n$. The proof of (4) follows from the following claim.

Claim: If f is monotone Boolean function, then the 2^n possible functions $f_{\boldsymbol{\alpha}}$ corresponding to the 2^n possible $\boldsymbol{\alpha}$'s are distinct if and only if f is non-degenerate.

Proof of the claim: Suppose f is degenerate on the i-th variable. Then f and $f_{\mathbf{e}_i}$ are equal. This proves one side of the claim. So suppose that f is non-degenerate. We have to show that for $\boldsymbol{\alpha} \neq \boldsymbol{\beta}$, $f_{\boldsymbol{\alpha}}$ and $f_{\boldsymbol{\beta}}$ are distinct. Let if possible $f_{\boldsymbol{\alpha}}$ and $f_{\boldsymbol{\beta}}$ be equal. Note that since f is non-degenerate, both $f_{\boldsymbol{\alpha}}$ and $f_{\boldsymbol{\beta}}$ are also non-degenerate. Since $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$ are distinct, there is a j in $\{1, \ldots, n\}$ such that $\alpha_j \neq \beta_j$. Suppose without loss of generality that $\alpha_j = 0$ and $\beta_j = 1$. Since f is monotone, it is monotone increasing in all variables and hence in the j-th variable. Further, since $\alpha_j = 0$, the function $f_{\boldsymbol{\alpha}}$ is monotone increasing in the j-th variable and since $\beta_j = 1$, the function $f_{\boldsymbol{\beta}}$ is monotone decreasing in the j-th variable. Since $f_{\boldsymbol{\alpha}}$ is monotone increasing in the j-th variable, for all $\mathbf{y} = (y_1, \ldots, y_n) \in \{0,1\}^n$ with $y_j = 0$, we have $f_{\boldsymbol{\alpha}}(\mathbf{y}) \leq f_{\boldsymbol{\alpha}}(\mathbf{y} \oplus \mathbf{e}_j)$. Further, since $f_{\boldsymbol{\alpha}}$ is non-degenerate, in particular it is non-degenerate on the j-th variable. Therefore, equality cannot hold everywhere, i.e., there is a $\mathbf{z} = (z_1, \ldots, z_n) \in \{0,1\}^n$ with $z_j = 0$, such that $f_{\boldsymbol{\alpha}}(\mathbf{z}) = 0$ and $f_{\boldsymbol{\alpha}}(\mathbf{z} \oplus \mathbf{e}_j) = 1$. Since $f_{\boldsymbol{\alpha}}$ and $f_{\boldsymbol{\beta}}$ are assumed to be equal, it follows that $f_{\boldsymbol{\beta}}(\mathbf{z}) = 0$ and $f_{\boldsymbol{\beta}}(\mathbf{z} \oplus \mathbf{e}_j) = 1$. This contradicts the fact that $f_{\boldsymbol{\beta}}$ is monotone decreasing in the j-th variable, and proves the claim.

Next we consider (6) and (7). We provide the proof of (6), and the proof of (7) being similar. From (4) and Proposition 1 we obtain (6) using the following calculation.

$$\begin{aligned} U_n = \sum_{i=0}^{n} \binom{n}{i} \text{nd-}U_i &= \sum_{i=0}^{n} \binom{n}{i} \cdot 2^i \cdot \text{nd-}M_i \qquad (8) \\ &\leq 2^n \cdot \sum_{i=0}^{n} \binom{n}{i} \text{nd-}M_i = 2^n \cdot M_n. \end{aligned}$$

□

2.1 Relation to the results of Baumann and Strass

Corollary 7 of Baumann and Strass [3] states (8). Baumann and Strass obtain Corollary 7 from Theorem 5 and Proposition 6 of their paper. (Baumann and Strass state Theorem 5 to be the main result of their paper.) The approach of Baumann and Strass to prove (8) is different from our approach. As a result, the proofs of Theorem 5 and Proposition 6 in Baumann and Strass' paper are quite a bit longer than the direct and simpler proof of Proposition 3.

Baumann and Strass [3] provide an informal argument that suggests that (4) holds. In particular, after Corollary 11, Baumann and Strass mentioned the following.

> bipolar Boolean functions have two choices for each non-redundant argument, namely, it can be either monotone or anti-monotone. Consequently, in case of i non-redundant arguments we have to consider the additional factor 2^i.

We note that while the above suggests that (4) holds, it is not sufficient to prove (4). There are subtleties to the counting argument that the proof of the claim in Proposition 3 covers.

Finally, we note that Baumann and Strass [3] did not obtain (5), which gives the relation between non-degenerate balanced unate functions and non-degenerate balanced monotone Boolean functions.

Next, we record two known facts about monotone Boolean functions.

Proposition 4 (Bakoev [1]). *Let g and h be n-variable Boolean functions and $f = g||h$. Then f is a monotone Boolean function if and only if g and h are both monotone Boolean functions and $g \leq h$.*

Proposition 5 (A003183)**.** *If f is a monotone Boolean function, then $\overline{f^r}$ is also a monotone Boolean function.*

Before proceeding further, we present some results on unate and monotone Boolean functions that are useful in our enumeration strategy. The first result is the analog of Proposition 4 for unate functions.

Proposition 6. *Let g and h be n-variable functions and $f = g||h$. Then f is an unate function if and only if g and h are both unate functions satisfying the following two conditions.*

(a) For each variable, both g and h are either monotone increasing, or both are monotone decreasing.

(b) Either $g \leq h$ or $h \leq g$.

Proof. First, consider the proof of the "if" part. Suppose g and h are unate functions satisfying the stated condition. We have to show that for each variable, f is either monotone increasing, or monotone decreasing. Consider the variable x_1. If $g \leq h$, then from (1), f is monotone increasing on x_1, while if $g \geq h$, then again from (1), it follows that f is monotone decreasing on x_1. Now consider a variable x_i, with $i \geq 2$. If g and h are both monotone increasing on x_i, then f is also monotone increasing on x_i, while if g and h are both monotone decreasing on x_i, then f is also monotone decreasing on x_i. Since for each variable, f is either monotone increasing, or monotone decreasing, it follows that f is a unate function.

For the converse, suppose that f is a unate function. Then for each variable x_i, where $i \geq 1$, f is either monotone increasing or monotone decreasing. From (1), it follows that for each variable x_i, where $i \geq 2$, g, and h are either both monotone increasing, or both monotone decreasing. So, in particular, g and h are unate. If f is monotone increasing for x_1, then $g \leq h$ and if f is monotone decreasing for x_1, then $g \geq h$. □

Proposition 7. *If f is a unate function then $\overline{f}$ is also a unate function.*

Proof. The proof is by induction on the number of variables n. The base case is $n = 1$ and is trivial. Suppose the result holds for some $n \geq 1$. Suppose that f is an $(n+1)$-variable unate function. Then f can be written as $f = g||h$, where g and h are n-variable unate functions satisfying the conditions in Proposition 6. Then $\overline{f} = \overline{g}||\overline{h}$. By induction hypothesis, $\overline{g}$ and $\overline{h}$ are n-variable unate functions and the conditions in Proposition 6 hold for $\overline{g}$ and $\overline{h}$. So $\overline{f}$ is a unate function. □

For $0 \leq w \leq 2^n$, let $M_{n,w}$ (resp., $U_{n,w}$) be the number of n-variable monotone (resp., unate) Boolean functions of weight w.

Proposition 8. *For every $n \geq 1$ and weight $w \in [0, 2^n]$, we have $M_{n,w} = M_{n,2^n-w}$.*

Proof. Proposition 5 sets up a one-one correspondence between n-variable monotone Boolean functions having weight w and n-variable monotone Boolean functions having weight $2^n - w$. This shows that $M_{n,w} = M_{n,2^n-w}$. □

Proposition 9. *For every $n \geq 1$ and weight $w \in [0, 2^n]$, we have $U_{n,w} = U_{n,2^n-w}$.*

Proof. Proposition 7 sets up a one-one correspondence between n-variable unate functions having weight w and n-variable unate functions having weight $2^n - w$. This shows that $U_{n,w} = U_{n,2^n-w}$. □

2.2 Equivalence

We recall the definition of equivalence of Boolean functions. Two Boolean functions are said to be *equivalent* if they have the same number of variables and one can be obtained from the other by a permutation of variables. Let $\mathcal{P}$ be a property of Boolean functions. The set $\mathcal{P}$ is partitioned into equivalence classes by the notion of equivalence. For $n \geq 0$, let $[P]_n$ denote the number of equivalence classes of n-variable functions possessing the property $\mathcal{P}$. Also, let nd-$[P]_n$ denote the number of equivalence classes of non-degenerate n-variable functions possessing the property $\mathcal{P}$.

Remark 10. We assume that for $n = 0$, there are two equivalence classes of n-variable, non-degenerate, monotone (and hence unate), and unbalanced Boolean functions given by [0] and [1].

We have the following analog of Proposition 1.

Proposition 11. *Let $\mathcal{P}$ be a property of Boolean functions that is closed under permutation of variables (i.e., if f is in $\mathcal{P}$ and g is obtained from f by applying a permutation to the variables, then g is also in $\mathcal{P}$). Then*

$$[P]_n = \sum_{i=0}^{n} \text{nd-}[P]_i. \tag{9}$$

Consequently, we have nd-$[P]_n = [P]_n - [P]_{n-1}$.

For $n \geq 0$, let $[A]_n$ denote the number of equivalence classes of n-variable Boolean functions and $[B]_n$ denote the number of equivalence classes of n-variable balanced Boolean functions. The values of $[A]_n$ and $[B]_n$ can be obtained using Polya's theory (see for example Roberts and Tesman [15, Sec. 8.3, p. 457]). Let nd-$[A]_n$ denote the number of equivalence classes of n-variable non-degenerate Boolean functions and nd-$[B]_n$ denote the number of equivalence classes of n-variable non-degenerate balanced Boolean functions. Using Proposition 11, it follows that nd-$[A]_n = [A]_n - [A]_{n-1}$ and nd-$[B]_n = [B]_n - [B]_{n-1}$.

For $n \geq 0$, let $[M]_n$, $[\text{BM}]_n$, $[U]_n$ and $[\text{BU}]_n$ denote the numbers of equivalence classes of n-variable monotone, balanced-monotone, unate, and balanced-unate functions respectively and let nd-$[M]_n$, nd-$[\text{BM}]_n$, nd-$[U]_n$, and nd-$[\text{BU}]_n$ denote the corresponding numbers of equivalence classes of non-degenerate functions. The following result is the analogue of Proposition 3.

Proposition 12. *For $n \geq 0$, the following holds.*

$$\text{nd-}[U]_n \leq 2^n \cdot \text{nd-}[M]_n, \tag{10}$$
$$\text{nd-}[\text{BU}]_n \leq 2^n \cdot \text{nd-}[\text{BM}]_n, \tag{11}$$
$$[U]_n \leq 2^n \cdot [M]_n, \tag{12}$$
$$[\text{BU}]_n \leq 2^n \cdot [\text{BM}]_n, \tag{13}$$

The relations given by (12) and (13) are analogues of (6) and (7) respectively. However, unlike (4) and (5), we do not have equality in (10) and (11). The reason is that two distinct input translations of a non-degenerate monotone Boolean function can lead to two unate functions that are equivalent. An example is the following. Suppose $f(X_1, X_2) = X_1X_2$, i.e., f is the AND function. Let $g(X_1, X_2) = f(1 \oplus X_1, X_2) = (1 \oplus X_1)X_2$ and $h(X_1, X_2) = f(X_1, 1 \oplus X_2) = X_1(1 \oplus X_2)$. Then $g(X_1, X_2) = h(X_2, X_1)$, i.e., g and h are distinct, but equivalent unate functions obtained by distinct input translations from the monotone Boolean function f.

Let f and g be two n-variable Boolean functions. We say that f and g are *NN-equivalent* if one can be obtained from the other by some combination of the following two operations: negation of a subset of the inputs, and negation of the output.

Proposition 13. *Let f and g be two n-variable monotone Boolean functions. Then f is NN-equivalent to g if and only if either $f = g$ or g is obtained from f by negation of the output and negation of all the variables on which f is non-degenerate.*

Consequently, every NN-equivalence class of n-variable monotone Boolean functions contains exactly two functions.

Proof. The "if" part follows from the definition of NN-equivalence, and we only need to prove the "only if" part. So suppose that f is NN-equivalent to g and assume $f \neq g$ (as otherwise there is nothing to prove).

If f is degenerate on some variable, then negating this variable does not change f. So we need to consider only the variables on which f is non-degenerate.

If f is the constant function 0 (resp., 1), then g is necessarily the constant function 1 (resp., 0). Next, suppose f is non-degenerate on exactly one variable, say X_1. Since $g \neq f$ and g is NN-equivalent to f, there are three possibilities for g, namely $1 \oplus f(X_1, \ldots)$, $f(1 \oplus X_1, \ldots)$ and $1 \oplus f(1 \oplus X_1, \ldots)$. Using the fact that f is monotone, it is easy to argue that the first two possibilities do not give rise to monotone Boolean functions, while the third possibility is a monotone Boolean function. Since g is also monotone, it follows that g must be equal to $1 \oplus f(1 \oplus X_1, \ldots)$. Now suppose that f is non-degenerate on exactly two variables, say X_1 and X_2. Since $g \neq f$ and g is NN-equivalent to f, there are seven possibilities for g. Among these, only $1 \oplus f(1 \oplus X_1, 1 \oplus X_2, \ldots)$ is monotone, while the other six are not. In more details, $f(1 \oplus X_1, X_2, \ldots)$ is not monotone on X_1, $f(X_1, 1 \oplus X_2, \ldots)$ is not monotone on X_2, $f(1 \oplus X_1, 1 \oplus X_2, \ldots)$ is not monotone on both X_1 and X_2, $1 \oplus f(1 \oplus X_1, X_2, \ldots)$ is not monotone on X_2, $1 \oplus f(X_1, 1 \oplus X_2, \ldots)$ is not monotone on X_1, and $1 \oplus f(X_1, X_2, \ldots)$

is not monotone on both X_1 and X_2. The above argument easily extends to the case where f is non-degenerate on $r \geq 1$ variables giving us the desired result. $\square$

Proposition 14. *The following holds.*

(a) For $n \geq 0$, the number of NN-inequivalent n-variable unate functions is equal to the number of NN-inequivalent n-variable monotone Boolean functions. Consequently, the number of NPN-inequivalent n-variable unate functions is equal to the number of NPN-inequivalent n-variable monotone Boolean functions.

(b) For $n \geq 0$, the number of NN-inequivalent n-variable balanced unate functions is equal to the number of NN-inequivalent n-variable balanced monotone Boolean functions. Consequently, the number of NPN-inequivalent n-variable balanced unate functions is equal to the number of NPN-inequivalent n-variable balanced monotone Boolean functions.

Proof. We prove only the first point since the proof of the second point is similar.

For every n-variable unate function f, by appropriately negating some of the inputs and also possibly the output, we obtain a monotone Boolean function g. This sets up a one-one correspondence between the class of NN-equivalent unate functions containing f and the class of NN-equivalent monotone Boolean functions containing g. So the number of NN-inequivalent n-variable unate functions is equal to the number of NN-inequivalent n-variable monotone Boolean functions.

The result on NPN-inequivalent functions follows since the one-one correspondence described above continues to hold when the notion of equivalence is extended by also considering the permutation of input variables. $\square$

3 Counting functions

In this section, we consider the problem of counting various sub-classes of monotone and unate Boolean functions.

3.1 Monotone Boolean functions

The values of BM_n are known for $n \leq 7$ (see A341633). From these values, using Proposition 1, we can obtain the values of nd-BM_n for $n \leq 7$. We present the obtained values of nd-BM_n in Table 3.

3.2 Unate functions

Baumann and Strass [3] showed that the problem of counting unate functions reduces to the problem of counting monotone Boolean functions. Using Proposition 1, this reduction can be seen as follows. First note that obtaining U_n reduces to the problem of obtaining nd-U_i,

n	BM_n	nd-BM_n
0	0	0
1	1	1
2	2	0
3	4	1
4	24	16
5	621	526
6	492288	488866
7	81203064840	81199631130

Table 3: Number of n-variable balanced monotone and non-degenerate balanced monotone Boolean functions for $0 \leq n \leq 7$ [A341633, A371722].

n	U_n	nd-U_n
0	2	2
1	4	2
2	14	8
3	104	72
4	2170	1824
5	230540	220608
6	499596550	498243968
7	309075799150640	309072306743552
8	14369391928071394429416818	14369391925598802012151296
9	146629927766168786368451678290041110762316052	146629927766168786239127150948525247729660416

Table 4: Number of n-variable unate and non-degenerate unate functions for $0 \leq n \leq 9$ [A245079, A305000].

for $0 \leq i \leq n$. From (4), obtaining nd-U_i reduces to the problem of obtaining nd-M_i for $0 \leq i \leq n$. Another application of Proposition 1 reduces the problem of obtaining nd-M_i to that of obtaining M_j for $0 \leq j \leq i$. So to obtain U_n, it is sufficient to know M_i for $0 \leq i \leq n$. Since the values of M_i are known for $0 \leq i \leq 9$, these provide the values of U_n for $0 \leq n \leq 9$. Due to the recent independent computation of M_9 by Van Hirtum et al. [10] and Jäkel [11], we are able to obtain the value of U_9. From the values U_n for $0 \leq n \leq 9$, using Proposition 1, we obtain the values of nd-U_n for $0 \leq n \leq 9$. Table 4 shows the values of U_n and nd-U_n.

Similarly, using Proposition 1 and (5), the problem of counting balanced unate functions reduces to the problem of counting balanced monotone Boolean functions. Since the values of BM_i are known for for $0 \leq i \leq 7$, we can obtain the values of BU_n for $0 \leq n \leq 7$. Using Proposition 1, this gives us the values of nd-BU_n for $0 \leq n \leq 7$. Table 5 shows the values of BU_n and nd-BU_n.

4 Counting equivalence classes of functions

In this section, we present the results on the numbers of equivalence classes of certain subsets of unate and monotone Boolean functions.

n	BU_n	nd-BU_n
0	0	0
1	2	2
2	4	0
3	14	8
4	296	256
5	18202	16832
6	31392428	31287424
7	10393772159334	10393552784640

Table 5: Number of n-variable balanced unate and non-degenerate balanced unate functions for $0 \leq n \leq 7$ [A373690, A373697].

4.1 Filtering procedure

The basic problem of enumerating equivalence classes is the following. Let $\mathcal{S}$ be a subset of the set of all n-variable Boolean functions. Given $\mathcal{S}$, we wish to generate a set $\mathcal{T} \subseteq \mathcal{S}$ of functions such that no two functions in $\mathcal{T}$ are equivalent, and each function in $\mathcal{S}$ is equivalent to some function in $\mathcal{T}$. The technique for such filtering is the following.

Given a permutation π of $\{1, \ldots, n\}$, we define a permutation $\pi^\star$ of $\{0, \ldots, 2^n - 1\}$ as follows. For $i \in \{0, \ldots, 2^n - 1\}$, let $(i_1, \ldots, i_n)$ be the n-bit binary representation of i. Then $\pi^\star(i) = j$, where the n-bit binary representation of j is $(j_{\pi(1)}, \ldots, j_{\pi(n)})$. Given an n-variable function f, let f^π denote the function such that for all $(x_1, \ldots, x_n) \in \{0,1\}^n$, the equation $f^\pi(x_1, \ldots, x_n) = f(x_{\pi(1)}, \ldots, x_{\pi(n)})$ holds. Suppose $f_0 \cdots f_{2^n-1}$ is the bit string representation of f. Then the bit string representation of f^π is $f_{\pi^\star(0)} \cdots f_{\pi^\star(2^n-1)}$.

Note that for each permutation π, the permutation $\pi^\star$ can be pre-computed and stored as an array say $P[0, \ldots, 2^n - 1]$. Suppose the bit string representation of f is stored as an array $A[0, \ldots, 2^n - 1]$. Then the bit string representation of f^π is obtained as the array $B[0, \ldots, 2^n - 1]$, where $B[i] = A[P[i]]$, for $i = 0, \ldots, 2^n - 1$. So obtaining f^π becomes simply a matter of array re-indexing.

Suppose the set of functions $\mathcal{S}$ to be filtered is given as a list of string representations of the functions. We incrementally generate $\mathcal{T}$ as follows. The first function in $\mathcal{S}$ is moved to $\mathcal{T}$. We iterate over the other functions in $\mathcal{S}$. For a function f in $\mathcal{S}$, we generate f^π for all permutations π of $\{1, \ldots, n\}$ using the technique described above. For each such f^π, we check whether it is present in $\mathcal{T}$. If none of the f^π's are present in $\mathcal{T}$, then we append f to $\mathcal{T}$. At the end of the procedure, $\mathcal{T}$ is the desired set of functions.

The check for the presence of f^π in $\mathcal{T}$ involves a search in $\mathcal{T}$. This is done using binary search. To apply binary search on a list, it is required that the list be sorted. To ensure this, we initially ensure that $\mathcal{S}$ is sorted (either by generating it in a sorted manner, or by sorting it after generation). This ensures that at every point, $\mathcal{T}$ is also a sorted list, so that binary search can be applied.

The reviewer of the paper suggested the following alternative method. Initially, $\mathcal{T}$ is

n	$[\mathrm{BM}]_n$	nd-$[\mathrm{BM}]_n$
0	0	0
1	1	1
2	1	0
3	2	1
4	4	2
5	16	12
6	951	935
7	16440466	16439515

Table 6: Number of equivalence classes of n-variable balanced monotone and non-degenerate balanced monotone Boolean functions for $0 \leq n \leq 7$ [A371717, A371718].

empty. As the algorithm proceeds $\mathcal{S}$ shrinks in size. The idea is the following. Pick the next function f from $\mathcal{S}$; for each permutation π of $\{1, \ldots, n\}$, generate f^π and drop f^π from $\mathcal{S}$ (if it is present); move f to $\mathcal{T}$. Since $\mathcal{S}$ shrinks in size as the algorithm proceeds, this method is in general faster than the above method. We have implemented both methods and obtained experimental support for the speed improvement.

4.2 Monotone Boolean functions

For $n \geq 0$, the numbers $[M]_n$ of equivalence classes of n-variable monotone Boolean functions form entry A003182 of OEIS. Using Proposition 11, it is possible to find the numbers nd-$[M]_n$ of equivalence classes of n-variable monotone Boolean functions (see A006602).

For $0 \leq n \leq 6$, we obtain the numbers $[\mathrm{BM}]_n$ of equivalence classes of n-variable balanced monotone Boolean functions by applying the filtering procedure described in Section 4.1 after enumerating the class of balanced monotone Boolean functions. This method, however, requires too much time to compute $[\mathrm{BM}]_7$. The techniques of Van Hirtum [9] provide a faster method. Although Van Hirtum [9] did not explicitly mention it, the associated code provides the values of $[\mathrm{BM}]_n$, for $n = 1, \ldots, 7$. By applying Proposition 11, we then obtain nd-$[\mathrm{BM}]_n$ for $n = 1, \ldots, 7$. Table 6 shows the values of $[\mathrm{BM}]_n$ and nd-$[\mathrm{BM}]_n$.

4.3 Unate functions

In the case of counting functions, the problems of counting unate and balanced unate functions reduce to the problems of counting monotone and balanced monotone Boolean functions respectively. In the case of counting equivalence classes of functions, such reduction is no longer possible (using the results that we could prove). The reason is that, unlike (4), which expresses the number of non-degenerate unate functions in terms of the number of non-degenerate monotone Boolean functions, the relation (10) only provides an upper bound on the number of equivalence classes of non-degenerate unate functions in terms of the number of equivalence classes of non-degenerate monotone Boolean functions.

In view of the above, for counting equivalence classes of unate functions, we resorted to the technique of enumerating unate functions and then using the technique described in Section 4.1 to obtain the number of equivalence classes.

The technique of generating all unate functions is based on Proposition 6. Along with the string representation of an unate function, we also need to record whether the function is increasing or decreasing in each of its variables. This is recorded as the signature of the function. The special cases of the two constant functions cause some complications in the definition of the signature.

For an n-variable unate function f, we define its signature, denoted $\mathrm{sig}(f)$, to be an element of $\{0,1\}^n \cup \{\mathfrak{z}, \mathfrak{o}\}$ in the following manner. If f is the constant function 1, then $\mathrm{sig}(f) = \mathfrak{o}$, if f is the constant function 0, then $\mathrm{sig}(f) = \mathfrak{z}$; otherwise $\mathrm{sig}(f)$ is an n-bit string α, where for $i = 1, \ldots, n$, $\alpha_i = 1$ if f is monotone increasing in the variable x_i, and $\alpha_i = 0$ if f is monotone decreasing in the variable x_i. The signature $\mathrm{sig}(f)$ encodes whether f is monotone increasing or monotone decreasing on each variable. The function f is both monotone increasing and monotone decreasing in all the variables if and only if it is a constant function. The signatures of the constant functions are defined appropriately.

For enumeration, the bit string representation of the functions is used. An unate function and its signature are stored as a pair. Consider the following recursive algorithm to generate all n-variable unate functions and their signatures for $n \geq 1$. At the base step, i.e., for $n = 1$, store the four pairs of 1-variable unate functions and their signatures as $(00, \mathfrak{z})$, $(01, 1)$, $(10, 0)$, and $(11, \mathfrak{o})$. Suppose that for some $n \geq 1$, we have already generated all n-variable unate functions and their signatures. The generation of all $(n+1)$-variable unate functions and their signatures are done as follows. For every two function-signature pairs $(g, \mathrm{sig}(g))$ and $(h, \mathrm{sig}(h))$, where g and h are n-variable unate functions (which are not necessarily distinct), perform the following checks:

1. Whether at least one of $\mathrm{sig}(g)$ or $\mathrm{sig}(h)$ is equal to either $\mathfrak{z}$ or $\mathfrak{o}$ (i.e., whether at least one of g or h is a constant function).
2. $\mathrm{sig}(g) = \mathrm{sig}(h) = \alpha$, and either $g \leq h$ or $h \leq g$ holds.

If either of the checks passes, then generate $f = g||h$, and determine $\mathrm{sig}(f)$ as follows.

$$\mathrm{sig}(f) = \begin{cases} \mathfrak{z}, & \text{if } \mathrm{sig}(g) = \mathrm{sig}(h) = \mathfrak{z}; \\ \mathfrak{o}, & \text{if } \mathrm{sig}(g) = \mathrm{sig}(h) = \mathfrak{o}; \\ 1^{n+1}, & \text{if } \mathrm{sig}(g) = \mathfrak{z},\ \mathrm{sig}(h) = \mathfrak{o}; \\ 0^{n+1}, & \text{if } \mathrm{sig}(g) = \mathfrak{o},\ \mathrm{sig}(h) = \mathfrak{z}; \\ 1||\alpha, & \text{if } \mathrm{sig}(g) = \mathfrak{z},\ \mathrm{sig}(h) = \alpha \in \{0,1\}^n; \\ 0||\alpha, & \text{if } \mathrm{sig}(g) = \mathfrak{o},\ \mathrm{sig}(h) = \alpha \in \{0,1\}^n; \\ 1||\alpha, & \text{if } \mathrm{sig}(g) = \alpha \in \{0,1\}^n,\ \mathrm{sig}(h) = \mathfrak{o}; \\ 0||\alpha, & \text{if } \mathrm{sig}(g) = \alpha \in \{0,1\}^n,\ \mathrm{sig}(h) = \mathfrak{z}; \\ 1||\alpha, & \text{if } g \leq h,\ \mathrm{sig}(g) = \mathrm{sig}(h) = \alpha \in \{0,1\}^n; \\ 0||\alpha, & \text{if } g \geq h,\ \mathrm{sig}(g) = \mathrm{sig}(h) = \alpha \in \{0,1\}^n. \end{cases}$$

Store $(f, \mathrm{sig}(f))$. Proposition 6 assures us that this recursive procedure generates all $(n+1)$-variable unate functions and their signatures.

To generate all $(n+1)$-variable unate functions, the above method requires considering all pairs of n-variable unate functions, i.e., a total of $\big(U(n)\big)^2$ options. Applying the filtering strategy of Section 4.1 we obtain the value of $[U]_n$. Next using Proposition 11 we obtain the value of nd-$[U]_n$. We could perform this computation for $n \leq 6$. Table 7 shows the obtained values of $[U]_n$ and nd-$[U]_n$. To generate all 7-variable unate functions using this option requires considering $\big(U(6)\big)^2 \approx 2^{57.8}$ pairs of functions. This is not feasible on the computing facility available to us.

n	$[U]_n$	nd-$[U]_n$
0	2	2
1	4	2
2	10	6
3	34	24
4	200	166
5	3466	3266
6	829774	826308

Table 7: Number of equivalence classes of n-variable unate and non-degenerate unate functions for $0 \leq n \leq 6$ [A372495, A374399].

To obtain the set of n-variable balanced unate functions, after generating the set of all n-variable unate functions, we remove the unbalanced ones. Then to the resulting set, we apply the technique of Section 4.1 to obtain the number $[\mathrm{BU}]_n$ of equivalence classes of n-variable balanced unate functions. Subsequently, we apply Proposition 11 to obtain the number nd-$[\mathrm{BU}]_n$ of equivalence classes of n-variable non-degenerate balanced unate functions. Table 8 shows the values of $[\mathrm{BU}]_n$ and nd-$[\mathrm{BU}]_n$.

n	$[\mathrm{BU}]_n$	nd-$[\mathrm{BU}]_n$
0	0	0
1	2	2
2	2	0
3	6	4
4	24	18
5	254	230
6	50172	49918

Table 8: Number of equivalence classes of n-variable balanced unate and non-degenerate balanced unate functions for $0 \leq n \leq 6$ [A374400, A374401].

4.4 NPN-equivalence:

From Proposition 14, we have that the number of NPN-inequivalent n-variable unate functions is equal to the number of NPN-inequivalent n-variable monotone Boolean functions. Also, the number of NPN-inequivalent n-variable balanced unate functions is equal to the number of NPN-inequivalent n-variable balanced monotone Boolean functions. As mentioned earlier, the number of NPN-inequivalent unate functions is already available in the OEIS (see A003183). So we consider the problem of counting the number of NPN-inequivalent balanced monotone (or unate) Boolean functions.

n	NPN-inequivalent M_n	NPN-inequivalent BM_n
0	1	0
1	2	1
2	3	1
3	6	2
4	17	4
5	112	15
6	8282	581

Table 9: Number of NPN-equivalence classes of n-variable monotone (or unate) and balanced monotone (or unate) Boolean functions for $0 \leq n \leq 6$ (see A003183, A378300).

From Proposition 13, to consider NPN-equivalence of monotone Boolean functions, it is sufficient to consider all possible permutations of the input variables along with simultaneous inversion of all the input variables and the output. Thus, for each monotone Boolean function, there are a total of $2 \times n!$ other functions to consider. We apply this idea in combination with the filtering technique described in Section 4.1. Since we are interested in inequivalent balanced monotone Boolean functions, we first prepare a list of balanced monotone Boolean functions and to this list, we apply the filtering technique. The number of functions left after the filtering procedure is complete is the number of NPN-inequivalent balanced monotone Boolean functions. The numbers of n-variable NPN-inequivalent balanced monotone Boolean functions for $n \leq 6$ are given in Table 9.

5 Future research

We mention some problems that may be tackled by future research.

The first problem is of counting balanced monotone Boolean functions. The recent algorithms designed by Jäkel [11] and Van Hirtum et al. [10] count all monotone Boolean functions. Modifying these algorithms to count balanced monotone Boolean functions is a possible research problem. The second problem is to apply the methods developed by Pawelski [14] to count the number of inequivalent balanced monotone Boolean functions. The third problem is to modify the methods of Pawelski [14] to count the number of inequivalent unate functions.

6 Acknowledgment

We are grateful to an anonymous reviewer of the Ph.D. thesis of the first author for suggesting that the number of n-variable non-degenerate unate functions is equal to 2^n times the number of n-variable non-degenerate monotone Boolean functions. We also thank the anonymous reviewer for providing valuable comments and suggestions, which have significantly improved the paper.

References

[1] Valentin Bakoev, Generating and identification of monotone Boolean functions, in *Mathematics and Education in Mathematics: Proc. of the Thirty Second Spring Conference of the Union of Bulgarian Mathematicians*, 2003, pp. 226–232.

[2] Charles R. Baugh, Generation of representative functions of the NPN equivalence classes of unate Boolean functions, *IEEE Trans. Comput.* **C-21** (1972), 1373–1379.

[3] Ringo Baumann and Hannes Strass, On the number of bipolar Boolean functions, *J. Logic Comput.* **27** (2017), 2431–2449.

[4] Joel Berman and Peter Köhler, Cardinalities of finite distributive lattices, *Mitt. Math. Sem. Giessen* **121** (1976), 103–124.

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

[6] Randolph Church, Enumeration by rank of the elements of the free distributive lattice with seven generators, *Notices Amer. Math. Soc.* **12** (1965), 724.

[7] Richard Dedekind, Über Zerlegung von Zahlen durch ihre grössten gemeinsamen Theiler, In *Festschrift Hoch. Braunschweig u. ges. Werke (II)*, 1897, pp. 103–148.

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

[9] Lennart Van Hirtum, A path to compute the 9th Dedekind number using FPGA supercomputing, Master's thesis, KU Leuven, 2023. Available at `https://hirtum.com/thesis.pdf`.

[10] Lennart Van Hirtum, Patrick De Causmaecker, Jens Goemaere, Tobias Kenter, Heinrich Riebler, Michael Lass, and Christian Plessl, A computation of the ninth Dedekind number using FPGA supercomputing, *ACM Transactions on Reconfigurable Technology and Systems* **17** (2024), 1–28.

[11] Christian Jäkel, A computation of the ninth Dedekind number, *J. Comput. Algebra* **6-7** (2023), 100006.

[12] Andrzej Kisielewicz, A solution of Dedekind's problem on the number of isotone Boolean functions, *J. Reine Angew. Math.* **386** (1988), 139–144.

[13] Aleksej D. Korshunov, Monotone Boolean functions, *Russian Math. Surveys* **58** (2003), 929–1001.

[14] Bartłomiej Pawelski, On the number of inequivalent monotone Boolean functions of 9 variables, *IEEE Trans. Inform. Theory* **70** (2024), 5358–5364.

[15] Fred Roberts and Barry Tesman, *Applied Combinatorics, 2nd edition*, Chapman and Hall/CRC, 2009.

[16] Neil J. A. Sloane, The On-Line Encyclopedia of Integer Sequences, `https://oeis.org/`, 2025.

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

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

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

2020 *Mathematics Subject Classification*: Primary 06E30.
Keywords: Boolean function, unate function, monotone Boolean function, Dedekind number.

(Concerned with the sequences A000372, A000618, A000721, A003182, A003183, A005612, A006126, A006602, A037293, A136094, A245079, A304999, A305000, A341633, A371717, A371718, A371722, A372495, A373690, A373697, A374399, A374400, A374401, A378300, and A378302.)

Received December 16 2023; revised versions received December 17 2023; December 27 2024; January 28 2025; April 16 2025. Published in *Journal of Integer Sequences*, April 17 2025.
