Counting unate and balanced monotone Boolean functions (Extended abstract)

Aniruddha Biswas and Palash Sarkar
Indian Statistical Institute
203, B.T.Road, Kolkata
India 700108.
Email: {anib_r, palash}@isical.ac.in

April 27, 2023

Abstract

For $n \leq 6$, we provide the number of n-variable unate and monotone Boolean functions under various restrictions. Additionally, we provide the number of balanced 7-variable monotone Boolean functions.
Keywords: Boolean function, unate Boolean function, monotone Boolean function, Dedekind number.
MSC: 05A99.

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.

Unate functions have been studied in the literature from various viewpoints such as switching theory, combinatorial aspects, and complexity theoretic aspects. Monotone Boolean functions have been studied much more extensively than unate functions and have many applications so much so that it is difficult to mention a few representative works. The focus of the present work is on counting unate and monotone Boolean functions under various restrictions.

A Boolean function is said to be balanced if it takes the values 0 and 1 equal number of times. Two Boolean functions are said to be equivalent, if one can be obtained from the other by a permutation of the variables. We say that two functions are inequivalent if they are not equivalent.

The number of n-variable Boolean functions is 2^{2^n} and the number of n-variable balanced Boolean functions is $\binom{2^n}{2^{n-1}}$. 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 becomes a reasonably simple problem. Non-triviality of counting Boolean functions arises for $n \geq 6$. Often though, it becomes difficult to obtain results for n more than 7 or 8.

The number of n-variable monotone Boolean functions is called the Dedekind number, denoted $D(n)$, after Dedekind who posed the problem in 1897. Till date $D(n)$ is known only up to $n = 8$ (see [6]). A closed form summation formula for $D(n)$ was given in [2], though it was pointed out in [3] that using the formula to compute $D(n)$ has the same complexity as direct enumeration of all n-variable monotone Boolean functions. The numbers of n-variable inequivalent monotone Boolean functions are known for n up to 8 (see [7, 4]). To the best of our knowledge, there is no work in the literature on counting the number of n-variable (inequivalent) balanced monotone Boolean functions.

The number of NPN-equivalence classes[1] of unate Boolean functions has been studied (see $A003183$ in [6]). Even though the problem is to count NPN inequivalent unate functions, the entry for $A003183$ in [6] shows that by using simple operations, the problem can be reduced to that of counting monotone functions under certain restrictions. A proper subclass of unate functions is the class of unate cascade functions which have been studied in [5]. Entry $A005612$ in [6] provides counts of unate cascade functions. To the best of our knowledge, there is no work in the literature on counting the number of n-variable (inequivalent) unate functions and the number of n-variable (inequivalent) balanced unate functions.

The following notation will be used.

UBF_n	: The set of all n-variable *unate* Boolean functions (UBFs).
MBF_n	: The set of all n-variable *monotone* Boolean functions (MBFs).
$U(n)$,$V(n)$	: Number of all n-variable UBFs and balanced UBFs respectively.
$W(n)$,$X(n)$	: Number of all n-variable inequivalent UBFs and balanced inequivalent UBFs respectively.
$D(n)$,$E(n)$	: Number of all n-variable MBFs and balanced MBFs respectively.
$F(n)$	: Number of all n-variable balanced inequivalent MBFs.

Our Contributions. We obtain the values of $U(n), V(n), W(n), X(n)$ and $F(n)$ for $n \leq 6$, and the values of $E(n)$ for $n \leq 7$. None of these values were previously known.

2 Mathematical Results

Let f be an n-variable Boolean function. By $\overline{f}$, we will denote the negation of f, i.e. $\overline{f}(\mathbf{x}) = 1$ if and only if $f(\mathbf{x}) = 0$. The weight $\mathsf{wt}(f)$ of f is the size of its support, i.e. $\mathsf{wt}(f) = \#\{\mathbf{x} : f(\mathbf{x}) = 1\}$. An n-variable Boolean function f can be uniquely represented 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 will 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 which represents f.

[1] Two Boolean functions are said to be NPN equivalent, if one can be obtained from the other by some combination of the following operations: a permutation of the variables, negation of a subset of the variables, and negation of the output. We say that two functions are NPN inequivalent if they are not NPN equivalent.

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 $f = g||h$.

In addition to the previous notation, we will also require the following notation.

$\mathcal{U}_{n,w}$, $\mathcal{M}_{n,w}$: Number of n-variable UBFs and MBFs of weight w respectively.

We record a known fact about MBFs.

Fact 1 [1] *Let g and h be n-variable Boolean functions and $f = g||h$. Then f is an MBF if and only if g and h are both MBFs and $g \leq h$.*

Next we present some new results on unate and monotone Boolean functions which will be useful in our enumeration strategy. However, the complete proofs will be presented in the full version.

Proposition 1 *Let g and h be n-variable Boolean functions and $f = g||h$. Then f is a UBF if and only if g and h are both UBFs satisfying the following two conditions.*

1. *For each variable, g and h are either both monotone increasing, or both monotone decreasing.*
2. *Either $g \leq h$ or $h \leq g$.*

Proposition 2 *If f is a UBF then $\overline{f}$ is also a UBF.*

Proposition 3 *For any $n \geq 1$ and weight $w \in [0, 2^n]$, $\mathcal{U}_{n,w} = \mathcal{U}_{n,2^n-w}$ and $\mathcal{M}_{n,w} = \mathcal{M}_{n,2^n-w}$.*

3 Enumeration Strategies

To generate all n-variable unate Boolean functions, the direct method is to generate all n-variable Boolean functions and then check each function for unateness. The problem with this approach is that there is no easy method to check whether a function is unate. So we adopted the recursive strategy which follows from Proposition 1 to generate unate functions which does not require the generation of all Boolean functions and hence is more efficient than the naive generate-and-check strategy. To generate all $(n+1)$-variable unate functions, our recursive algorithm requires considering all pairs of n-variable unate functions, i.e. a total of $(U(n))^2$ options. This is feasible for $n \leq 5$, but not for higher values of n. However, there is some subtlety in developing a recursive generation algorithm based on Proposition 1 which will be described in the full version. To obtain balanced functions, from the set of all unate functions, we filter out the ones that are unbalanced.

A similar and somewhat simpler method to recursively generate all n-variable monotone functions can be obtained from Fact 1. This method also becomes infeasible for $n > 6$. It is, however, possible to generate all 7-variable balanced monotone functions using a modified version of the recursive enumeration. We provide a general description of the method.

Suppose the set of all n-variable monotone functions have already been generated. Partition these functions into weight classes, where the number of n-variable monotone functions in weight class w is $\mathcal{M}_{n,w}$, $w = 0, \ldots, 2^n$. The method for generating all $(n+1)$-variable monotone functions

based on Fact 1 is modified as follows. Choose g from weight class w and h from weight class $2^n - w$ and check whether $g \leq h$. If the check passes, then generate $f = g||h$. The procedure ensures that the weight of f is 2^n, so that f is an $(n+1)$-variable monotone balanced function. The number of pairs of n-variable unate functions that need to be considered is $\sum_{w=0}^{2^n} \mathcal{M}_{n,w}\mathcal{M}_{n,2^n-w} = \sum_{w=0}^{2^n} (\mathcal{M}_{n,w})^2$, where the equality follows from Proposition 3. This is a substantial reduction from $(D(n))^2$ that would be otherwise required. To generate all balanced 7-variable monotone functions, the generate-and-filter strategy would have required considering $(D(6))^2 \approx 2^{45}$ pairs. The modified strategy requires considering $\sum_{w=0}^{64} (\mathcal{M}_{6,w})^2 \approx 2^{40}$ pairs, which makes the enumeration much faster. The modified method for generating balanced monotone functions can also be adapted to generate balanced unate functions and for generating all $(n+1)$-variable balanced unate functions requires considering $\sum_{w=0}^{2^n} (\mathcal{U}_{n,w})^2$ pairs.

Along with enumerating all unate functions, we also enumerate inequivalent unate functions. Similarly for the other classes of functions we enumerate inequivalent functions in the respective classes. Given a permutation π of $\{1, \ldots, n\}$ and an n-variable function f, let f^π denote the function such that for all $(x_1, \ldots, x_n) \in \{0,1\}^n$, $f^\pi(x_1, \ldots, x_n) = f(x_{\pi(1)}, \ldots, x_{\pi(n)})$. Consider the set $\mathcal{S}$ to be filterd is given as a list of 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.

We are interested in the number of inequivalent n-variable unate functions. So we may apply the above inequivalent filtering procedure to UBF_n. It is possible to gain efficiency by noting that the weight of a function is invariant under permutation of variables. So instead of applying the inequivalent filtering procedure to UBF_n, we apply it to the weight-wise partition of UBF_n. This leads to a gain in efficiency, since a function of weight w is checked for equivalence only with other functions of weight w. The strategy for inequivalent filtering works in the same way for balanced unate functions, monotone functions and balanced monotone functions. This allows us to also find the number of inequivalent functions in these classes.

The results of the above enumeration procedures for the different classes of functions are shown in Tables 1 to 3.

n	1	2	3	4	5	6
$U(n)$	4	14	104	2170	230540	499596550
$V(n)$	2	4	14	296	18202	31392428

Table 1: Number of n-variable (balanced) UBFs for $n \leq 6$

n	1	2	3	4	5	6
$W(n)$	4	10	34	200	3466	829774
$X(n)$	2	2	6	24	254	50172

Table 2: Number of n-variable (balanced) inequivalent UBFs for $n \leq 6$

n	1	2	3	4	5	6	7
$E(n)$	1	2	4	24	621	492288	81203064840
$F(n)$	1	1	2	4	16	951	–

Table 3: Number of balanced (inequivalent) monotone functions.

4 Concluding Remarks

With access to a adequate computing resources, it should be possible to obtain $F(7)$, i.e. the number of inequivalent balanced monotone functions, and $V(7)$, i.e. the number of balanced unate functions. Both of these require computations which is somewhat more that 2^{50}. The computations can be parallelised and distributed across a large number of cores. With access to a sufficiently large computational cluster, the computations will be feasible. On the other hand, obtaining the values of $U(n), W(n)$ and $X(n)$ for $n \geq 7$ and the values of $V(n), E(n)$ and $F(n)$ for $n \geq 8$ will require new ideas. The running times for the enumeration of the corresponding sets using the techniques used in the present work are likely to remain infeasible in the forseeable future.

References

[1] Valentin Bakoev. Generating and identification of monotone Boolean functions. In *Mathematics and Education in Mathematics, Sofia*, pages 226–232, 2003.

[2] Andrzej Kisielewicz. A solution of Dedekind's problem on the number of isotone Boolean functions. *Journal fur die Reine und Angewandte Mathematik*, 1988(386):139 – 144, 1988.

[3] Aleksej D. Korshunov. Monotone Boolean functions. *Russian Mathematical Surveys*, 58(5):929 – 1001, 2003.

[4] Bartłomiej Pawelski. On the number of inequivalent monotone Boolean functions of 8 variables. `https://arxiv.org/pdf/2108.13997.pdf`, 2021.

[5] Tsutomu Sasao and Kozo Kinoshita. On the number of fanout-free functions and unate cascade functions. *IEEE Transactions on Computers*, 28(1):66–72, 1979.

[6] Neil J.A. Sloane. The online encyclopedia of integer sequences. `https://oeis.org/`, 1964.

[7] Tamon Stephen and Timothy Yusun. Counting inequivalent monotone Boolean functions. *Discrete Applied Mathematics*, 167:15–24, 2014.