COUNTING INEQUIVALENT MONOTONE BOOLEAN FUNCTIONS

TAMON STEPHEN AND TIMOTHY YUSUN

Abstract. Monotone Boolean functions (MBFs) are Boolean functions $f : \{0,1\}^n \to \{0,1\}$ satisfying the monotonicity condition $x \leq y \Rightarrow f(x) \leq f(y)$ for any $x, y \in \{0,1\}^n$. The number of MBFs in n variables is known as the nth Dedekind number. It is a longstanding computational challenge to determine these numbers exactly – these values are only known for n at most 8. Two monotone Boolean functions are equivalent if one can be obtained from the other by permuting the variables. The number of inequivalent MBFs in n variables was known only for up to $n = 6$. In this paper we propose a strategy to count inequivalent MBFs by breaking the calculation into parts based on the profiles of these functions. As a result we are able to compute the number of inequivalent MBFs in 7 variables. The number obtained is 490013148.

1. Introduction

A *Boolean function on n variables* (BF) is a function $f : \{0,1\}^n \to \{0,1\}$. A *monotone Boolean function* (MBF) additionally satisfies the condition $x \leq y \Rightarrow f(x) \leq f(y)$, for any $x, y \in \{0,1\}^n$; these are also known as *isotone* Boolean functions. We write $x \leq y$ if $x_i \leq y_i$ for all $i = 1, 2, \ldots, n$, and $x < y$ if $x \leq y$ and $x_i < y_i$ for some i. A BF is monotone if and only if it can be written as a combination of conjunctions and disjunctions only.

Since each input state in $\{0,1\}^n$ has two possible output states, there are a total of 2^{2^n} Boolean functions on n variables. On the other hand, no exact closed form is known for the number of *monotone* Boolean functions on n variables. Denote this number by $D(n)$. The numbers $D(n)$ are called the *Dedekind numbers*, after R. Dedekind [Ded97]. The first few values are given in Table 1 [Slo11]. Currently, only values of $D(n)$ up to $n = 8$ are known.

Kisielewicz [Kis88] gave a logical summation formula for $D(n)$, however performing the computation using his summation has the same complexity as brute force enumeration of $D(n)$, see, e.g., [Kor03]. There are some asymptotic results concerning the behavior of $D(n)$, one of the earliest of which was a result of Kleitman in 1969, that $\log_2 D(n) \sim \binom{n}{\lfloor n/2 \rfloor}$ [Kle69]. So far, the most accurate one is given by Korshunov [Kor03], found in Table 2.

2010 *Mathematics Subject Classification.* Primary 68W05; Secondary 06E30, 05A05.

Key words and phrases. Dedekind numbers, monotone Boolean functions, isotone Boolean functions, inequivalent monotone Boolean functions.

n	$D(n)$	Source
0	2	Dedekind, 1897
1	3	
2	6	
3	20	
4	168	
5	7 581	Church, 1940 [Chu40]
6	7 828 354	Ward, 1946 [War46]
7	2 414 682 040 998	Church, 1965 [Chu65] (see also [BK76])
8	56 130 437 228 687 557 907 788	Wiedemann, 1991 [Wie91] (see also [FMSS01])

TABLE 1. Known Values of $D(n)$, (A000372, [Slo11]).

$D(n) \sim 2^{\binom{n}{n/2}} \cdot \exp\left[\binom{n}{\frac{n}{2}-1}\left(2^{-n/2} + n^2 2^{-n-5} - n2^{-n-4}\right)\right]$, for even n
$D(n) \sim 2^{\binom{n}{(n-1)/2}+1} \cdot \exp\left[\binom{n}{\frac{n-3}{2}}\left(2^{(-n-3)/2} - n^2 2^{-n-5} - n2^{-n-3}\right) + \binom{n}{\frac{n-1}{2}}\left(2^{(-n-1)/2} - n^2 2^{-n-4}\right)\right]$, for odd n

TABLE 2. Korshunov's Asymptotics

1.1. **Inequivalent MBFs.** We define an MBF $f(x_1, x_2, \ldots, x_n)$ to be equivalent to another MBF $g(x_1, x_2 \ldots, x_n)$, written as $f \sim g$, if there is a permutation $\sigma \in S_n$ such that $f(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)}) = g(x_1, x_2, \ldots, x_n)$. For example, the function $f(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_2 \wedge x_3)$ is equivalent to $g(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_1 \wedge x_3)$ via the permutation $\sigma = (12)$.

Let $R(n)$ be the number of equivalence classes defined by "$\sim$" among monotone Boolean functions on n variables. As with $D(n)$, no closed form is known for $R(n)$, and in fact only the values up to $n = 6$ have been computed. These appear to have been obtained by the straightforward method of listing all monotone Boolean functions on n variables and then sorting them into equivalence classes. The known values are shown in Table 3. We will denote by $\mathcal{D}(n)$ and $\mathcal{R}(n)$ the *sets* of monotone Boolean functions and their equivalence classes, respectively. So $D(n) = |\mathcal{D}(n)|$ and $R(n) = |\mathcal{R}(n)|$.

Example 1.1. The inequivalent MBFs in two variables are: $f = 0$, $f = 1$, $f = x_1$, $f = x_1 \vee x_2$, and $f = x_1 x_2$; $\mathcal{R}(2)$ is the set comprised by these functions (equivalence classes) and $R(2) = 5$. Then $\mathcal{D}(2)$ consists of these functions along with $f = x_2$, which is equivalent to $f = x_1$; $D(2) = 6$.

n	0	1	2	3	4	5	6
$R(n)$	2	3	5	10	30	210	16 353

TABLE 3. Known Values of $R(n)$ (A003182, [Slo11])

1.2. **Terminology and Elementary Facts.** Define a *minimal term* of an MBF f to be an input $x \in \{0,1\}^n$ such that $f(x) = 1$ and $f(y) = 0$ if $y < x$. The minimal terms of a monotone Boolean function are the minimal vectors where the function equals 1 – any input below a minimal term in the lattice of 0-1 vectors evaluates to 0, and everything above evaluates to 1 by virtue of monotonicity. For example, the function $f(x_1, x_2, x_3) = x_1 \vee x_2x_3$ evaluates to 1 at $(1, 0, 0)$ and $(0, 1, 1)$, as well as at all vectors above $(1, 0, 0)$ and $(0, 1, 1)$ in the lattice, and evaluates to 0 at all other vectors. Indeed, each MBF can be written as a disjunction of clauses, each representing one of its minimal terms. This is known as its minimal *disjunctive normal form*, or DNF [CH11]. For the rest of this paper we will represent 0-1 vectors by the sets of indices of their non-zero entries – e.g. $(1, 0, 0)$ by $\{1\}$ and $(0, 1, 1)$ by $\{2, 3\}$.

MBFs can be classified according to the number of minimal terms they have. Call $D_k(n)$ the number of monotone Boolean functions on n variables with k minimal terms. Kilibarda and Jovovic [KJ03] derive closed form expressions for $D_k(n)$ for fixed $k = 4, 5, \ldots, 10$ (A051112 to A051118, [Slo11]).

A *truth table* for a Boolean function is a row of zeros and ones which encodes the outputs of the function corresponding to every possible input state. To illustrate, the function $f(x_1, x_2, x_3) = x_1 \vee x_2x_3$ from above has minimal terms $\{1\}$ and $\{2, 3\}$, and so it has the following truth table:

variables set to 1	x_1, x_2, x_3	x_2, x_3	x_1, x_3	x_3	x_1, x_2	x_2	x_1	none
output states	1	1	1	0	1	0	1	0

TABLE 4. Truth table for the function $f(x_1, x_2, x_3) = x_1 \vee x_2x_3$

Note that the input states on the top row are arranged in a reverse colexicographic (or *colex*) order on $\{0,1\}^3$, defined as $x < y$ if $x \neq y$ and $x_k < y_k$ where $k = \max\{i : x_i \neq y_i\}$. Fixing this order, we write f as 11101010. In general, any Boolean function on n variables can be written as a 0-1 string of length 2^n where each entry corresponds to an input state; we use this convention throughout this paper. The ordering has the nice property that the first 2^{n-1} of its entries have $x_n = 1$, and the second half have $x_n = 0$.

The truth table form is the most compact way to represent general Boolean functions. For monotone Boolean functions, both the truth table and the minimal terms representation are useful for our purposes.

Example 1.2. The colexicographic order for two variables is $\{1,2\} > \{2\} > \{1\} > \{\}$. The functions in $\mathcal{D}(2)$, written in truth table form are {1111, 1110, 1100, 1010, 1000, 0000}.

1.3. **Background.** Monotone Boolean functions are ubiquitous objects in mathematics, occurring in many guises. For instance, there is a one-to-one correspondence between MBFs in $\mathcal{D}(n)$ and antichains in the set $2^{[n]}$, that is, pairwise incomparable subsets of the power set of $\{1, 2, \ldots, n\}$. Specifically, the set of minimal terms of an MBF in $\mathcal{D}(n)$ is an antichain in $2^{[n]}$. Since by Sperner's Theorem, any antichain on the n-set can have at most $\binom{n}{\lfloor n/2 \rfloor}$ elements, we have that any n-variable MBF can have at most $\binom{n}{\lfloor n/2 \rfloor}$ minimal terms [Eng97].

The one-to-one correspondence between n-variable MBFs and Sperner hypergraphs is also well-known. In particular, each minimal term of an MBF maps to a hyperedge in the corresponding hypergraph, and because of pairwise incomparability, the hypergraph thus exhibits the Sperner property, that is, no hyperedge contains another.

Some other fields in which monotone Boolean functions appear include lattice theory [Grä71], nonlinear signal processing [SSGC95], coding theory [IKN07], computational learning theory [Shm], game theory [RP11], and computational biology [KG04],[HKS08].

For a comprehensive discussion of Boolean functions, see the recently-published book by Crama and Hammer [CH11].

2. Computational Strategies

2.1. **Profiles of MBFs.** It is natural to refine the classification of monotone Boolean functions by number of minimal terms, and consider how many elements are contained in each of these terms. We define the notion of a profile formally as given in Engel [Eng97], and introduce some notation:

Definition 2.1 (Profile of an MBF)**.** Given an n-variable MBF f where $f \not\equiv 1$, the **profile** of f is a vector of length n, $(a_1, a_2, \ldots, a_n)$, where the ith entry is equal to the number of minimal terms of f which are i-sets.

Example 2.2. The MBF 11111100 has minimal terms $\{2\}, \{3\}$, and profile $(2, 0, 0)$, while the MBF 11111000 has minimal terms $\{1, 2\}$, $\{3\}$, and profile $(1, 1, 0)$.

Definition 2.3. Given profile vector $(a_1, a_2, \ldots, a_n)$, define $(a_1, a_2, \ldots, a_n)_D$ to be the number of monotone Boolean functions on n variables with profile vector $(a_1, a_2, \ldots, a_n)$. Similarly define $(a_1, a_2, \ldots, a_n)_R$ for inequivalent monotone Boolean functions on n variables.

Note that the number of variables n is implicit in the profile vector – it is the length of the vector. Some relations between the profiles are described and proven in Lemma 2.4.

Lemma 2.4. *Assume that all profile vectors pertain to MBFs on n variables, unless otherwise stated.*

(A): $(0,\ldots,0,a_i,0,\ldots,0)_R = (0,\ldots,0,\binom{n}{i}-a_i,0,\ldots,0)_R$.
(B): *If* $a_1 > 0$, *then* $a_n = 0$ *and* $(a_1,a_2,\ldots,a_{n-1},a_n)_R = (a_1-1,a_2,\ldots,a_{n-1})_R$.
(C): $(a_1,a_2,\ldots,a_{n-2},a_{n-1},a_n)_R = (a_{n-1},a_{n-2},\ldots,a_2,a_1,a_n)_R$.

*Statements **(A)** and **(C)** hold true when R is replaced by D.*

Proof. The proof of each claim rests on the fact that there is a one-to-one correspondence between functions of the first type and functions of the second type, for the purposes of counting both $D(n)$ and $R(n)$. For statements **(A)** and **(C)**, we will prove the claim for $D(n)$ first, from which the results for $R(n)$ follow.

(A): Given an MBF with exactly a_i i-sets as minimal terms, we can derive another MBF with minimal terms exactly the $\binom{n}{i}-a_i$ i-sets which were not taken in the first MBF. For example, the 3-variable MBF with minimal terms $\{1,2\},\{2,3\}$ corresponds to the MBF with minimal term $\{1,3\}$. So, we must have $(0,\ldots,0,a_i,0,\ldots,0)_D = (0,\ldots,0,\binom{n}{i}-a_i,0,\ldots,0)_D$.

The statement then follows for $R(n)$ by observing that the images of any two equivalent functions under this correspondence will also be equivalent, via the same permutation.

(B): Suppose that f is an MBF with profile $(a_1,a_2,\ldots,a_{n-1},a_n)$, $a_1 > 0$. Without loss of generality assume that $\{n\}$ is a minimal term of f. (We can assume this because we are only considering $R(n)$.)

This implies that f cannot have minimal terms containing the element n, and hence $a_n = 0$. In addition, removing the term $\{n\}$, we are left with an $(n-1)$-variable MBF, with the profile $(a_1-1,a_2,\ldots,a_{n-1})$.

(C): Let $f \in \mathcal{D}(n)$. If $a_n = 1$, then $[n] := \{1,2,\ldots,n\}$ is the only minimal term. This implies that all the other a_i's are zero, and hence the claim follows trivially.

If $a_n = 0$, assume that the minimal terms of an MBF f with the given profile are $A_1,A_2,\ldots,A_k$. We know that none of the A_i's are comparable, so it follows that none of the sets $[n]-A_1,[n]-A_2,\ldots,[n]-A_k$ must be comparable to one another as well. Hence the collection $\{[n]-A_j\}_{1\le j\le k}$ is the set of minimal terms of an MBF g where the number of i-sets is equal to the number of $(n-i)$-sets in f. This proves that the profile of g is $(a_{n-1},a_{n-2},\ldots,a_2,a_1,a_n)$.

As an example, the 5-variable MBF with minimal terms $\{1,2\}$, $\{1,3,4\}$, $\{2,4,5\}$ is mapped to the MBF with minimal terms $\{3,4,5\}$, $\{2,5\}$, $\{1,3\}$ under this correspondence. As in **(A)**, if we have $(a_1,a_2,\ldots,a_{n-2},a_{n-1},a_n)_D = (a_{n-1},a_{n-2},\ldots,a_2,a_1,a_n)_D$ then the statement is also true for $R(n)$.

□

Lemma 2.4 is very useful in reducing the amount of computation that needs to be done to compute $D(n)$ or $R(n)$. For instance, when generating

$\mathcal{R}(7)$, we build the profiles $(0,0,k,0,0,0,0)$ for $k = 1$ to 17 using an expensive algorithm, and then profiles for $k = 18$ to $k = \binom{7}{3} = 35$ are cheaply computed via complements, using the idea from the proof of part **(A)**. Part **(B)** enables us to refer back to $R(6)$ when considering profiles with a nonzero entry in the first position. The most useful is **(C)**, which effectively cuts all computation time in half.

2.1.1. *Generating Profiles.* To generate all n-variable MBFs using these profiles, we first need to determine which vectors of length n are the profile vectors for some MBF. It is not immediately apparent how to do so. The *number* of profile vectors for any n is sequence `A007695` [Slo11]. The entry for `A007695` also includes undocumented code that can be used to compute this number. This code does not explicitly generate these profiles.

n	Number of profiles
0	1
1	2
2	4
3	9
4	25

n	Number of profiles
5	95
6	552
7	5460
8	100708
9	3718353

TABLE 5. Number of profiles for each n, from $n = 0$ to $n = 9$, (`A007695`, [Slo11]).

In Section 4 we present an algorithm, based on ideas from the code, to output the profiles of n-variable MBFs. Here we briefly outline the algorithm. The main idea is to use the incomparability condition on the minimal terms of MBFs to generate new profiles from vectors we already know are profiles of MBFs.

The algorithm starts by generating all profiles consisting only of 1-sets, then 1-sets and 2-sets and so on. Consider, by way of example, the profile $P = (1,3,0,0,0)$ – this is the profile vector of several 5-variable MBFs. Observing that any 3-set is comparable to 3 2-sets, from P we construct the profile $(1,0,1,0,0)$ by subtracting 3 from the second entry and adding 1 to the third entry.

In general, the algorithm proceeds from $(r-1)$-sets to r-sets by determining the number of $(r-1)$-sets that must be comparable to at least one set from a collection of k r-sets for some fixed k. Since the largest rth entry that a profile vector can have is $\binom{n}{r}$, we precompute the answer to this question for collections of sizes $1, 2, \ldots, \binom{n}{r}$. Then, given the list of profiles with $a_{r-1} > 0$ and $a_k = 0$ for $k \geq r$, we repeat the technique in the previous paragraph to build up all profiles with $a_r > 0$ and $a_k = 0$ for $k > r$.

2.1.2. *Using Profiles to Generate Functions.* A monotone Boolean function can be written uniquely as the disjunction of its minimal terms. Thus we can generate all MBFs inductively beginning with profiles that have a single

non-zero entry. Let $\mathcal{Q}_k^i(n)$ be the collection of MBFs on n variables with exactly k minimal terms, all of size i. We will take the liberty of omitting the argument n, which, for the purposes of this section discussion, is fixed. The $\mathcal{Q}_k^i$ are exactly the MBF's with profile $(\underbrace{0,\ldots,0}_{i-1\ times},k,\underbrace{0,\ldots,0}_{n-i\ times})$.

Fixing n and i, we generate $\mathcal{Q}_1^i, \mathcal{Q}_2^i, \ldots, \mathcal{Q}_{\lfloor\binom{n}{2}/2\rfloor}^i$ in turn, with the remaining non-zero $\mathcal{Q}_k^i$ obtained by complements via Lemma 2.4 part (A). Since we are counting inequivalent MBFs, without loss of generality $\mathcal{Q}_1^i$ contains only the function whose single minimal true clause is $\{1,2,\ldots,i\}$. To build $\mathcal{Q}_{k+1}^i$ from $\mathcal{Q}_k^i$, we take each MBF from $\mathcal{Q}_k^i$ and, for each function, take all i-sets that are not already minimal true clauses. Each pairing of a function and an i-set generates a candidate MBF with the appropriate profile. However, many of these MBFs will be duplicates.

To handle duplicates, we convert each MBF to a canonical MBF (for its equivalence class under permutations of the variables) before storing it. Recall that we are using truth tables to represent MBFs; the disjunction of two functions represented by truth tables is computed via a bitwise OR. We precompute the truth table for the MBF generated by each possible single i-set, as these are used to produce the new functions via disjunction. Once the function f has been identified, we generate the full set of $n!$ functions equivalent to f by permuting the variables. These are obtained by permuting the 2^n coordinates of the truth table; the exact permutation corresponding to each of the $n!$ permutations is precomputed.

We choose the canonical MBF to be the lexicographically smallest of the $n!$ truth tables in the set. These canonical representatives are stored in an ordered list of truth tables for $\mathcal{Q}_{k+1}^i$. Duplicates have identical canonical truth tables and thus are only counted once; they are dropped rather than inserted in the list.

Consider now profiles with multiple non-zero entries. The simplest situation is where the profile differs from some $\mathcal{Q}_k^i$ by having a single second non-zero entry with index $j \neq i$ and the entry is 1. Then we can proceed as before, generating the profile beginning with the functions that comprise $\mathcal{Q}_k^i$ and taking disjunctions with all possible j-sets. Again, since the functions are represented as truth tables, a lookup of the relevant j-set allows us to see if the disjunction is non-trivial and hence has the appropriate profile. When this occurs, we convert to a canonical MBF and continue until all possibilities are exhausted.

In the same manner, we can build all possible profiles with two non-zero entries, and then on to three, four and beyond. Note that profiles with multiple non-zero entries can be computed from one of several adjacent profiles. In Section 3 we discuss briefly our strategies for moving through the profiles. We remark that several profiles were computed more than once from different adjacent profiles, and in all cases the same result was obtained.

Finally, we note that one MBF, the function which is true for all inputs, does not correspond to any profile as per Definition 2.1, and is accounted for separately. This function has the empty set as its lone minimal true clause: if Definition 2.1 were modified to include an extra a_0 term then it would be the lone function with $a_0 > 0$. We did not do this as this increases clutter without removing the need to handle this case separately.

Remark 2.5. We are in particular interested in the case $n = 7$, where the techniques described in this section are comfortably within the range of a modern computer. This is not true for $n = 8$, and the techniques certainly don't make sense asymptotically. One issue is finding the canonical function by brute force: each prospective function is expanded into a list of $7! = 5040$ equivalent functions and this is then searched for the lexicographic minimum. At first glance one might hope to do better, but since the list can be generated quickly by multiplying by a precomputed matrix, investing in a complicated algorithm here would offer at most a mild return. For $n = 8$, brute force is possible, but alternatives would become more appealing.

We frequently generate all j-sets of n for use in disjunctions. For $n = 7$ this is no more than 35 sets. For $n = 8$ this doubles to a still-manageable $\binom{8}{4} = 70$, though it becomes prohibitive for large n.

A more serious issue is the size of the profiles. The largest profile in $\mathcal{R}(7)$ is that of $(0, 0, 7, 7, 0, 0, 0)$, with 5443511 MBFs, slightly more than 1% of the total. A back of the envelope calculation suggests that the largest profile in $\mathcal{R}(8)$ would have around 10^{16} MBFs. The number of profiles also increases rapidly with n, referring back to Table 5, we see there are 100708 for $n = 8$. To give the reader an idea of the distribution of MBFs by profile, we include in Table 6 the number of inequivalent MBFs by profile for five-variable functions.

As a byproduct of the calculations done for $R(7)$, we also extended the known values for the sequences $R_k(n)$ included in the OEIS. The corresponding sequences for $D_k(n)$ are `A051112` to `A051118` [Slo11]. The new values that we have computed are included in Table 7.

2.1.3. *Computing Bounds on $R(n)$ and $D(n)$.* Since each MBF can have at most $7! - 1 = 5039$ other functions equivalent to it, we know that $D(7)/7! \sim 479$ million is a lower bound for $R(7)$. In fact, we can increase the lower bound by looking for highly symmetric functions. For instance, the MBF with minimal term $\{1\}$ is equivalent to only six other MBFs, all with one singleton set as the only minimal term. Hence this equivalence class only has 7 functions, so by considering this class of functions we could increase the lower bound to $(D(7) + 5040 - 7)/7!$. Doing this for all equivalence classes of functions with at most two minimal terms, and some simple equivalence classes with three and four minimal terms increases the lower bound of $D(7)/7!$ by only about 500. This suggests that $R(7)/(D(7)/7!)$ will be fairly close to 1.

Profile	#	Profile	#	Profile	#	Profile	#
(0,0,0,0,0)	1	(0,9,0,0,0)	1	(1,0,3,0,0)	1	(0,2,0,1,0)	1
(1,0,0,0,0)	1	(0,10,0,0,0)	1	(0,1,3,0,0)	6	(0,3,0,1,0)	1
(2,0,0,0,0)	1	(0,0,1,0,0)	1	(0,2,3,0,0)	6	(0,4,0,1,0)	1
(3,0,0,0,0)	1	(1,0,1,0,0)	1	(0,3,3,0,0)	4	(0,0,1,1,0)	1
(4,0,0,0,0)	1	(2,0,1,0,0)	1	(0,4,3,0,0)	1	(0,1,1,1,0)	1
(5,0,0,0,0)	1	(0,1,1,0,0)	2	(0,0,4,0,0)	6	(0,2,1,1,0)	1
(0,1,0,0,0)	1	(1,1,1,0,0)	1	(1,0,4,0,0)	1	(0,0,2,1,0)	2
(1,1,0,0,0)	1	(0,2,1,0,0)	4	(0,1,4,0,0)	6	(0,1,2,1,0)	1
(2,1,0,0,0)	1	(1,2,1,0,0)	1	(0,2,4,0,0)	4	(0,0,3,1,0)	3
(3,1,0,0,0)	1	(0,3,1,0,0)	6	(0,3,4,0,0)	1	(0,1,3,1,0)	1
(0,2,0,0,0)	2	(1,3,1,0,0)	1	(0,4,4,0,0)	1	(0,0,4,1,0)	2
(1,2,0,0,0)	2	(0,4,1,0,0)	6	(0,0,5,0,0)	6	(0,0,5,1,0)	1
(2,2,0,0,0)	1	(0,5,1,0,0)	4	(0,1,5,0,0)	4	(0,0,6,1,0)	1
(0,3,0,0,0)	4	(0,6,1,0,0)	2	(0,2,5,0,0)	1	(0,0,0,2,0)	1
(1,3,0,0,0)	3	(0,7,1,0,0)	1	(0,0,6,0,0)	6	(0,1,0,2,0)	1
(2,3,0,0,0)	1	(0,0,2,0,0)	2	(0,1,6,0,0)	2	(0,0,1,2,0)	1
(0,4,0,0,0)	6	(1,0,2,0,0)	1	(0,0,7,0,0)	4	(0,0,2,2,0)	1
(1,4,0,0,0)	2	(0,1,2,0,0)	4	(0,1,7,0,0)	1	(0,0,3,2,0)	1
(0,5,0,0,0)	6	(1,1,2,0,0)	1	(0,0,8,0,0)	2	(0,0,0,3,0)	1
(1,5,0,0,0)	1	(0,2,2,0,0)	7	(0,0,9,0,0)	1	(0,0,1,3,0)	1
(0,6,0,0,0)	6	(0,3,2,0,0)	6	(0,0,10,0,0)	1	(0,0,0,4,0)	1
(1,6,0,0,0)	1	(0,4,2,0,0)	4	(0,0,0,1,0)	1	(0,0,0,5,0)	1
(0,7,0,0,0)	4	(0,5,2,0,0)	1	(1,0,0,1,0)	1	(0,0,0,0,1)	1
(0,8,0,0,0)	2	(0,0,3,0,0)	4	(0,1,0,1,0)	1	**TOTAL**	**209**

TABLE 6. Number of inequivalent five-variable MBFs by profile.

k	$R_k(5)$	$R_k(6)$	$R_k(7)$
2	13	22	34
3	30	84	202
4	49	287	**1321**
5	48	787	**8626**
6	34	1661	**50961**
7	18	2630	**253104**
8	7	3164	**1025322**
9	2	2890	**3365328**
10	2	2159	**9005678**
11	0	1327	**19850932**

TABLE 7. Partial list of values $R_k(n)$, values in boldface were not known to us.

This raises the interesting question of what the functions in high-cardinality equivalence classes look like, that is, functions which have few or no symmetries. For $n \leq 5$ functions with no symmetries are quite rare, however it

appears that they already are the overwhelming majority when $n = 7$. We computed the number of inequivalent n-variable MBFs that have no symmetries starting from $n = 1$ by enumerating $\mathcal{R}(n)$. This gives the sequence 0, 1, 0, 0, 7, 7281; we haven't computed the exact number for $n = 7$.

3. Implementation Details

All computations were done on MATLAB, a high-level scientific computing language [MAT]. We used three computational clusters: the Optima cluster at SFU Surrey, the IRMACS computational cluster, and the bugaboo cluster of WestGrid under Compute Canada. We used MATLAB for building a prototype because it is easy to get started, and it is built for handling large vectors and matrices. It has many built-in functions that work well with the types of lists we are generating, and if desired, further work can be transported over to other programming languages.

In MATLAB, we represented MBFs as their truth table forms, 1×2^n row vectors. This representation also lends itself well to using 32-bit integers instead of long 0-1 strings. Given an MBF of length 2^n, we partitioned the zeros and ones into blocks of length 32, which we considered as a binary number $(a_{31}a_{30}\dots a_2a_1a_0)_2$, representing $\sum_{k=0}^{31} a_k 2^k$. If n is smaller than 5, the truth table form has less than 32 entries, and we pad with zeros on the right. For $n \geq 5$, an n-variable MBF can be written as 2^{n-5} 32-bit integers.

Example 3.1. The six-variable MBF

$$f = \underset{\uparrow\,\uparrow \qquad \uparrow\;\uparrow \qquad\qquad \uparrow \qquad\qquad \uparrow \quad \uparrow\uparrow}{\texttt{1111111011111110111111100100000001111101011101010111110000000000}}$$

has 64 entries, so it is divided into two blocks of 32:

$$\texttt{11111110111111101111110010000000} \to 20938623$$
$$\texttt{11111010111010101111100000000000} \to 2053983$$

hence the 32-bit integer representation of f is $(20938623, 2053983)$. Its minimal terms are $\{1, 2, 4\}$, $\{3, 4\}$, $\{1, 5\}$, $\{2, 3, 5\}$, $\{1, 2, 3, 6\}$, $\{2, 4, 6\}$, $\{2, 5, 6\}$, and $\{3, 5, 6\}$, represented by the numbers 11, 12, 17, 28, 39, 42, 50 and 52 respectively. The indices of these minimal terms are identified by arrows above.

The algorithms we used involve building and frequently referencing a very long list of functions. To do this efficiently we used a hash table and a nonlinear hashing function on the 32-bit representations to perform checks and lookups quickly. In particular, we used a polynomial hash function, which acts on the four integers (say b_1, b_2, b_3, and b_4) modulo a prime p by repeatedly adding a number $\alpha > 2$, and multiplying by the next component, i.e. we take $b_1(\alpha + b_2(\alpha + b_3(\alpha + b_4)))$ $(\texttt{mod}\ p)$. For the profiles where hashing was used, we used a hash table of size 20 million; the largest profile has size 5443511. We believe the collision rate from this scheme on the generated MBFs is not much higher than for a random hashing function.

Each profile P was generated from a profile P' which differs from P by one in a single coordinate. This allowed some freedom in traversing this lattice of profiles. We kept the following observations in mind. All profiles with a nonzero 1st entry can be obtained from the corresponding profile in $R(6)$, via Lemma 2.4 part (B); further profiles with a nonzero 6th entry can be handled by using this in combination with Lemma 2.4 part (C). Remaining profiles with a nonzero 2nd or 5th entry were computed by first adding the necessary 2-sets and 5-sets as any such set is comparable to many 3-sets and 4-sets, and thus substantially reduce the search space and profile size. The largest profiles we encountered were those solely containing 3-sets and 4-sets. In fact, the functions in these profiles account for 366689638 out of the total of 490013148 for $R(7)$, or 74.8%.

We observed that when moving from profile P to profile P' by adding j-sets that any MBF in P' will be generated, since from an MBF with profile P' (which has $j \geq 1$), removing any single j-set that is a generator gives an MBF with profile P. On the other hand, the use of canonical representatives ensured that only a single function per equivalence class was included in the computation of each P'.

In the computation, we generated the list of functions for profiles with exactly one nonzero entry first, then proceeded through the list of profiles with the above considerations as a general guide. Note that the branches of computation were independent of each other and so multiple calculations can be made to run concurrently. Also many profiles were computed more than once, either as an intermediate step with the goal of computing a larger profile, or as a redundancy check. For example, the profile $(0, 0, 3, 4, 0, 0, 0)_R$ was computed both by a process generating profiles of the form $(0, 0, 3, x, 0, 0, 0)_R$ and another one generating profiles of the form $(0, 0, x, 4, 0, 0, 0)_R$. We saved lists of functions to disk for the larger profiles generated to have various points to start further computations or recover (as some jobs lasted several weeks and were susceptible to system shutdowns, etc.).

All results of computations were saved by the script files as text files, which include the numbers obtained and the computation time (Fig. 1). To keep track of the data, we stored all the results in a database together with the computation times, with a running total (Fig. 2). At the end we obtained the number $R(7) = \mathbf{490013148}$.

Our MATLAB code and the database shown in Figure 2 are available on the website [Yus12]. To ensure accuracy, we checked the minimal terms of around 10,000 functions from each profile in a random sample: ensuring that the functions in the input file indeed correspond to the profile we start with. We also tested our algorithm for $n \leq 6$ and obtained the correct number for $R(n)$. We also observe that the number obtained is slightly higher than the lower bound of 479 million discussed in Section 2.1.3, as expected.

```
>> >> >> >> >> >> >> >> >> >> >> Initial profile [0 0 15 2 0 0 0] has 2931257 functions.

Profile [0 0 15 3 0 0 0] done after 422847.33 seconds.
Number of functions: 2294434

Profile [0 0 15 4 0 0 0] done after 701392.33 seconds.
Number of functions: 1141753

Profile [0 0 15 5 0 0 0] done after 820305.51 seconds.
Number of functions: 390945
```

FIGURE 1. Sample output log of the computation

Profile #	# minterms	a1	a2	a3	a4	a5	a6	a7	
1932	17	0	4	1	12	0	0	0	28
1933	18	0	5	1	12	0	0	0	2
1934	19	0	6	1	12	0	0	0	1
1935	14	0	0	2	12	0	0	0	2635180
1936	15	0	1	2	12	0	0	0	89339
1937	16	0	2	2	12	0	0	0	1110
1938	17	0	3	2	12	0	0	0	71
1939	18	0	4	2	12	0	0	0	3
1940	15	0	0	3	12	0	0	0	4126847
1941	16	0	1	3	12	0	0	0	64838
1942	17	0	2	3	12	0	0	0	577
1943	18	0	3	3	12	0	0	0	36
1944	16	0	0	4	12	0	0	0	4291110
1945	17	0	1	4	12	0	0	0	31174
1946	18	0	2	4	12	0	0	0	263

FIGURE 2. Portion of database where results are stored

We tracked computation times using MATLAB's somewhat unreliable `cputime` function. The migration between systems[1], the introduction of hashing after some profiles had been computed, and overflow in the internal representation of the `cputime` function[2], make it difficult to assign a meaningful total time to the computation.

To give an idea of the total CPU time that would be required to rerun the entire calculation on the WestGrid cluster, the largest profile, $(0, 0, 7, 7, 0, 0, 0)$ had a MATLAB `cputime` reading of just under a month (29.3 days). This contains more than 1% of the profiles, and should be slower than average to generate (small profiles can be generated quickly). Thus we expect that it would take less than 8 years of CPU time to run the entire computation

[1]The computation began on the Optima cluster at SFU Surrey, which consists of 9 dual core blade machines shared among several users. Since the computation parallelizes very naturally, we started some as well on the IRMACS computational cluster at SFU, which includes another 10 dual core blades, and was not being heavily used. This continued until the IRMACS machines were disabled for a time. Then we moved the largest jobs to the much larger Bugaboo cluster on WestGrid, which consists of 446 blades with 8 to 12 fast cores each. It did work noticeably faster than the smaller clusters.

[2]An advertised bug in MATLAB.

in this configuration. We remark again that the computation is highly parallelizable. Our computations took about a year in real time across several machines.

4. The Profile-Generating Algorithm

Algorithm 1: Generating all profiles of MBFs on n variables.

Input: n
Output: $\mathcal{P}(n)$, all profiles of n-variable MBFs
Step 1: Precompute $K(r,s)$, lower bounds for how many $(r-1)$-sets are comparable to s number of r-sets
Initialize $K(0,s) := 0$ for $0 \leq s \leq \binom{n}{\lfloor n/2 \rfloor}$
for $r = 1$ **to** n **do**
  Set $k := r$
  Set $K(r,0) := 0$
  for $s = 1$ **to** $\binom{n}{r}$ **do**
    if $s \geq \binom{k}{r}$ **then**
      $k \leftarrow k+1$
    end
    $K(r,s) = K(r-1, s - \binom{k-1}{r}) + \binom{k-1}{r-1}$
  end
end
Step 2: Generate all profiles using $K(r,s)$
Initialize $\mathcal{P}(n) = \{(0,\ldots,0)\} \cup \{(i,0,\ldots,0) : i = 1,2,\ldots,n\}$
for $r = 2$ **to** n **do**
  for $s = 1$ **to** $s = \binom{n}{r}$ **do**
    Let $\mathcal{P}_{r,s} = \{(a_1,a_2,\ldots,a_n) : a_{r-1} \geq K(r,s), a_i = 0, i \geq r\}$.
    Update $\mathcal{P}(n) \leftarrow \mathcal{P}(n) \cup \{(a_1,a_2,\ldots,a_{r-1} - K(r,s), s, 0,\ldots,0) : (a_1,a_2,\ldots,a_n) \in \mathcal{P}_{r,s}\}$.
  end
end
Output $\mathcal{P}(n)$

We prove the correctness of Algorithm 1 by first proving that the values $K(r,s)$ generated in the first step are lower bounds for the number of $(r-1)$-sets comparable to a collection of s sets of cardinality r.

Lemma 4.1. *Given $2 \leq r \leq n$ and $0 < s \leq \binom{n}{r}$, $K(r,s)$ in Step 1 of Algorithm 1 is the smallest number of $(r-1)$-sets comparable to any collection of s r-sets.*

Proof. First, observe that because we increase k by 1 if $s \geq \binom{k}{r}$, whenever we recursively calculate $K(r,s) = K(r-1, s - \binom{k-1}{r}) + \binom{k-1}{r-1}$, it will always be true that $\binom{k-1}{r} \leq s < \binom{k}{r}$.

We are looking for a best possible collection in that it contains the smallest number of $(r-1)$-sets. So, since $s < \binom{k}{r}$, such a collection has to contain only k distinct elements. (Any collection with more than k elements will be comparable to a larger number of $(r-1)$-sets.) Without loss of generality assume that this best collection contains the elements $\{1, 2, \ldots, k\}$.

Because $s \geq \binom{k-1}{r}$, this collection also has to contain *all* r-subsets of $\{1, 2, \ldots, k-1\}$ (again, without loss of generality). This is in turn comparable to all $(r-1)$-subsets of $\{1, 2, \ldots, k-1\}$.

This leaves us with $s - \binom{k-1}{r}$ sets remaining in the collection, all containing the element k. Since all $(r-1)$-subsets of $\{1, 2, \ldots, k-1\}$ have been accounted for, we only have to look at the number of $(r-1)$-sets that also contain k, comparable to these remaining sets. By removing the element k from each of them, we see that this number is bounded below by $K(r-1, s-\binom{k-1}{r})$.

Therefore by the two previous observations, $K(r,s) = K(r-1, s-\binom{k-1}{r}) + \binom{k-1}{r-1}$, completing the proof. □

Theorem 4.2. *The list $\mathcal{P}(n)$ in Algorithm 1 contains all profiles of monotone Boolean functions on n variables.*

Proof. We perform induction on the rightmost nonzero entry in a profile vector.

When $\mathcal{P}(n)$ is initialized, the empty profile and all profiles with a single nonzero entry in the first position are included. This is the collection of all MBFs with only 1-sets as minimal terms, and there are n such profiles as there are n such sets in $2^{[n]}$.

Now assume that $\mathcal{P}(n)$ contains all profiles with the rightmost nonzero entry in the $(r-1)$-th position.

Since $K(r,s)$ is the smallest number of $(r-1)$-sets comparable to any s number of r-sets, profiles with $a_r = s$ and $a_i = 0$ for $i > r$ can be obtained by taking all profiles with at least $K(r,s)$ in the $(r-1)$-th position, subtracting $K(r,s)$ from a_{r-1}, and setting a_r to s.

Note that as we are only looking at profile vectors, this operation of removing $(r-1)$-sets and adding r-sets will not always work for particular MBFs. However because the $K(r,s)$ values represent lower bounds that are achieved, doing this operation will always result in a profile that is also achieved by some MBF.

Repeating this for all $s = 1, 2, \ldots, \binom{n}{r}$ will result in all profile vectors of MBFs with $a_r > 0$ and $a_i = 0$ for $i > r$. This completes the induction, and so the output $\mathcal{P}(n)$ contains all profiles of n-variable MBFs. □

5. Conclusions and Discussion

In this paper we propose a strategy for counting inequivalent monotone Boolean functions (MBFs), which is a challenging enumeration problem on a fundamental combinatorial object. The strategy is to break the computation

into smaller parts based on profiles of MBFs. We describe and implement a non-trivial algorithm to generate the profiles. Using profiles, we are able to generate the full set of inequivalent 7-variable MBFs in manageable pieces, which in particular allows us to find that the number of such functions is $R(7) = \mathbf{490013148}$.

At present it appears difficult to extend this technique to computing $R(8)$ because it requires generating, rather than merely counting, the profiles. Actual enumeration of the MBFs is needed to identify equivalent functions, in contrast to the situation with $D(n)$, and it is difficult to see how to get around this. The set $\mathcal{R}(8)$ contains more than $D(8)/8! \approx 1.4 \times 10^{16}$ MBFs distributed across 100708 profiles, and thus would require substantial computational resources to compute under any strategy that enumerates the functions. For $n \geq 9$ enumeration strategies appear hopeless.

It is appealing to try to use $R(7)$ to compute $D(9)$, which is presently not known. Wiedemann in 1991 computed $D(8)$ using $D(6)$ and $R(6)$, by going through all pairs of functions in $D(6) \times R(6)$, and using a lookup function to calculate how many functions in $D(8)$ can be formed by fixing two "middle functions". See [Wie91] for details.

To apply this technique to the computation of $D(9)$, we would need to generate $D(7)$ from $R(7)$, store the number of functions in each equivalence class, and then calculate the number of MBFs each function contains as a preprocessing step. The difficulty lies in the sheer amount of computation needed, but as the strategy is simple to parallelize, there is some hope. To make the calculation more manageable we would like to understand the symmetries of monotone Boolean functions better. We might start by trying to count functions by their symmetry group, or by extending the sequence of inequivalent non-symmetric MBFs that we consider in Section 2.1.3.

6. Acknowledgments

Parts of this work were included in the M.Sc. thesis of T. Yusun [Yus11]. This research was partially supported by an NSERC Discovery Grant, and by a grant from the SFU office of the Vice-President, Research. We would like to thank the SFU Math Department, the IRMACS Centre at SFU, and WestGrid and Compute Canada for the access to the computational resources needed to perform the calculations in this paper.

Also, we are grateful to Michael Monagan and Utz-Uwe Haus for discussion, in particular we thank MM for pointing out that we needed to use hash tables for the large lists. We appreciate the detailed and thoughtful comments of the anonymous referees, which improved the presentation of the paper.

References

[BK76] J. Berman and P. Köhler, *Cardinalities of finite distributive lattices*, Mitt. Math. Sem. Giessen (1976), no. Heft 121, 103–124.

[CH11] Y. Crama and P.L. Hammer, *Boolean functions: Theory, algorithms, and applications*, Encyclopedia of Mathematics and Its Applications, Cambridge University Press, 2011.

[Chu40] R. Church, *Numerical analysis of certain free distributive structures*, Duke Mathematical Journal **6** (1940), no. 3, 732–734.

[Chu65] ———, *Enumeration by rank of the free distributive lattice with 7 generators*, Notices of the American Mathematical Society (1965), no. 11, 724.

[Ded97] R. Dedekind, *Über Zerlegungen von Zahlen durch ihre größten gemeinsamen Teiler*, Ges. Werke, Vol. 2, 1897, pp. 103–148.

[Eng97] K. Engel, *Sperner theory*, Cambridge University Press, 1997.

[FMSS01] R. Fidytek, A. Mostowski, R. Somla, and A. Szepietowski, *Algorithms counting monotone Boolean functions*, Information Processing Letters **79** (2001), 203–209.

[Grä71] George Grätzer, *Lattice theory. First concepts and distributive lattices*, W. H. Freeman and Co., San Francisco, Calif., 1971.

[HKS08] U. Haus, S. Klamt, and T. Stephen, *Computing knock-out strategies in metabolic networks*, J. Comput. Biol. **15** (2008), no. 3, 259–268.

[IKN07] H. Ito, M. Kobayashi, and G. Nakamura, *Semi-distance codes and Steiner systems*, Graph. Comb. **23** (2007), 283–290.

[KG04] S. Klamt and E.D. Gilles, *Minimal cut sets in biochemical reaction networks*, Bioinformatics **20** (2004), no. 2, 226–234.

[Kis88] A. Kisielewicz, *A solution of Dedekind's problem on the number of isotone Boolean functions*, J. Reine Angew. Math. **386** (1988), 139–144.

[KJ03] G. Kilibarda and V. Jovovic, *On the number of monotone Boolean functions with fixed number of lower units (in Russian)*, Intellektualnye sistemy **7** (2003), no. 1-4, 193–217.

[Kle69] D. Kleitman, *On Dedekind's problem: The number of monotone Boolean functions*, Proc. Amer. Math. Soc. **21** (1969), 677–682.

[Kor03] A. Korshunov, *Monotone Boolean functions*, Russian Mathematical Surveys **58** (2003), no. 5, 929–1001.

[MAT] MATLAB, The MathWorks Inc., Natick, Massachusetts.

[RP11] F. Riquelme and A. Polyméris, *On the complexity of the decisive problem in simple and weighted games*, Electronic Notes in Discrete Mathematics **37** (2011), 21–26.

[Shm] I. Shmulevich, *Computational learning theory*, `http://personal.systemsbiology.net/ilya/LEARN.htm`.

[Slo11] N. J. A. Sloane, *The online encyclopedia of integer sequences*, `http://oeis.org`, 2011.

[SSGC95] I. Shmulevich, T. M. Sellke, M. Gabbouj, and E. J. Coyle, *Stack filters and free distributive lattices*, Proceedings of the 1995 IEEE Workshop on Nonlinear Signal and Image Processing (Halkidiki, Greece), 1995, pp. 927–930.

[War46] M. Ward, *Note on the order of free distributive lattices*, Bulletin of the American Mathematical Society (1946), no. 52, 423.

[Wie91] D. Wiedemann, *A computation of the eighth Dedekind number*, Order **8** (1991), no. 1, 5–6.

[Yus11] T. Yusun, *Dedekind numbers and related sequences*, M.Sc. thesis, Department of Mathematics, Simon FraserUniversity, 2011.

[Yus12] ———, *Counting inequivalent monotone Boolean functions*, `http://sfu.ca/~tyusun/inequivalentMBF.html`, 2012.

Department of Mathematics, Simon Fraser University, 8888 University Drive, Burnaby, B.C. V5A 1S6, Canada
E-mail address: tamon@sfu.ca
E-mail address: tyusun@sfu.ca