Counting of Moore Families for n=7

Pierre Colomb[1], Alexis Irlande[2], and Olivier Raynaud[1]

[1] Université Blaise Pascal, Campus Universitaire des Cézeaux, 63173 Aubière, France
[2] Universidad Nacional de Colombia, Bogota, Colombia

Abstract. Given a set $U_n = \{0, 1, ..., n-1\}$, a collection $\mathcal{M}$ of subsets of U_n that is closed under intersection and contains U_n is known as a Moore family. The set of Moore families for a given n, denoted by $\mathtt{M}_n$, increases very quickly with n, thus $|\mathtt{M}_3|$ is 61 and $|\mathtt{M}_4|$ is 2480. In [1] the authors determined the number for $n = 6$ and stated a 24h- computation-time. Thus, the number for $n = 7$ can be considered as an extremely difficult technical challenge. In this paper, we introduce a counting strategy for determining the number of Moore families for $n = 7$ and we give the exact value : 14 087 648 235 707 352 472. Our calculation is particularly based on the enumeration of Moore families up to an isomorphism for n ranging from 1 to 6.

1 Introduction

The counting (and/or enumeration) of a large set of mathematical objects is a pleasant challenge. This kind of exercise requires original algorithmic processes, which involves a thorough knowledge of the properties of the objects to be counted, efficient search data structures, but also, and above all, state-of-the-art programming techniques. In this paper, our efforts have been concentrated on the counting of Moore families generated by a given set $U_n = \{0, 1, ..., n-1\}$. The concept of Moore family, or of closure operator (extensive, isotone and idempotent function of 2^{U_n} in 2^{U_n}), or of implicational system, is applied in numerous fields. For example, let's consider mathematics research such as [2] for algebra, computer science such as [3] for the theory of orders and lattices, [4] for relational databases and finally [5] and [6] for data analysis. The name 'Moore family' was first used by Birkhoff in [7] referring to E.H. Moore 's early century research in [8]. Technically, a Moore family on U_n, denoted by $\mathcal{M}$, is a collection of sets (or family) closed under intersection and containing U_n (cf. figure 1).

The set of Moore families on U_n, denoted $\mathtt{M}_n$, is itself a closure system (a closure system being the set of the fixed points of a closure operator). Thus, the system composed of Moore families contains one maximum element (2^{U_n} : all subsets of U_n) and the intersection of two Moore families is a Moore family itself. To get an overall view of the properties of this closure system, see [9]. Ordering the set of elements of a closure system by inclusion, we get a lattice structure. Indeed, an inclusion ordering of the set of elements of a closure system can generate a lattice structure.

In [10], Burosch considers the issue of counting Moore families as natural, so he suggests an upper bound for that number. In this paper, we will complete

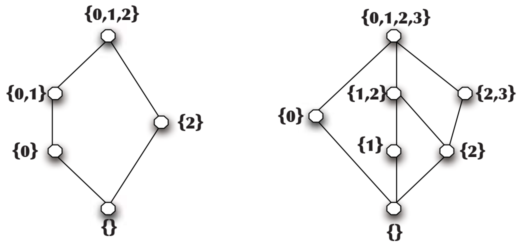

Fig. 1. Two moore families : On the left, family $\{\emptyset, \{0\}, \{0,1\}, \{2\}, \{0,1,2\}\}$ on the set $\{0,1,2\}$. On the right, family $\{\emptyset, \{0\}, \{1\}, \{2\}, \{1,2\}, \{2,3\}, \{0,1,2,3\}\}$ on the set $\{0,1,2,3\}$.

his survey using a new $|\mathtt{M}_n|$-bound based on $|\mathtt{M}_{n-1}|$. In 1998, Higuchi in [11] calculated $|\mathtt{M}_5|$ by a depth first search study of the covering graph of a Moore family lattice structure. More recently, Habib and Nourine have evaluated in [1] the size of $\mathtt{M}_6$. Their method is based on the existence of a bijection between the set of Moore families and the ideals colorset of a colored poset composed of boolean lattices. Thanks to an efficient algorithm of enumeration of order ideal sets (cf. [12]), the authors managed to count $\mathtt{M}_6$ and stated that the process would take 24h on a pentium III 600 MegaHertz (we note that the method presented here compute this number $\mathtt{M}_6$ in around 120ms on a Core2quad Q9300 2,5 GigaHertz). The whole set of values of this counting is given in table 1. Considering the exponential development of the results, the evaluation of the number for $n = 7$ turned out to be a particularly difficult challenge.

Table 1. Known values of $|\mathcal{M}_n|$ on $n \leq 7$

n	$\lvert\mathtt{M}_n\rvert$	Référence
0	1	
1	2	
2	7	
3	61	
4	2 480	
5	1 385 552	[11]
6	75 973 751 474	[1]
7	14 087 648 235 707 352 472	This paper

The rest of the paper is composed as follows. The second part is devoted to the data structure and key points on which our calculation strategy was based: **symmetry concept**, **canonical form** and **maximal family**. In the third part, we will present the main algorithmic principles we have implemented. Then, in the fourth part, we will deal with the technological aspects of the calculation process (type of machine, performance, reliability index). As a conclusion, we will put things into perspective.

2 Strategy Elements

This part defines the essential elements of our calculation strategy. Firstly, we will describe the coding used to store and process the Moore families. Then, we will explain the various concepts used as the basis of this calculation strategy: The symmetries between the families, the concept of canonical form as the identifier of an equivalence class as well as the concept of maximal family based on the recursive structure of the objects to be counted. The proofs of the propositions are given in the appendix.

In the introduction, we have defined a Moore family on U_n as a collection of sets containing U_n and closed by intersection. However for the reasons of convenience that the reader will find in the course of reading, our entire algorithmic process will concentrate on enumeration of families closed by union and containing the empty set. All the Moore families are in fact in bijection with this set. Actually, for a family closed by union containing the empty set, one only has to complement every set to obtain a Moore family (and vice-versa).

For example, the U_3 family $\{\{0\},\{0,1\},\{0,2\},\{0,1,2\}\}$ corresponds to the Moore family $\{\emptyset,\{1\},\{2\},\{1,2\}\}$ and vice versa.

2.1 Encoding

Let us consider a $U_n = \{0,...,n-1\}$ universe with n elements and a set $E \subseteq U_n$. E can be naturally encoded by its characteristic vector (for example, refer to figure 2).

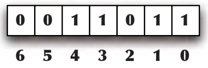

Fig. 2. For the universe $U_7 = \{0,...,6\}$, the set $\{0,1,3,4\}$ is encoded by a 7 bit vector $b[]$ such that $b[i] = 1$ if and only if $i \in \{0,1,3,4\}$

We can associate a decimal value to each subset of E by interpreting the characteristic vector as a binary number. To be more precise, an integer between 0 and 2^n-1 corresponds to each sub-set of $\{0,...,n-1\}$. In the previous example, the integer associated to the set $\{0,1,3,4\}$ is equal to the sum of $2^0+2^1+2^3+2^4$ i.e. 27. By using these decimal values as set identifiers, we can also encode a set family on a universe with n elements by a characteristic vector of 2^n bits (for example, refer to figure 3).

Finally, we associate an integer between 0 and 2^{2^n} with each family by interpreting the new vector as a binary number. For example, 131 is the identifier of the family $\{\emptyset,\{0\},\{0,1,2\}\}$. As we will see, using such an encoding enables carrying out normal operations on the sets by simple logic or arithmetic operations.

Two operations are essential in our counting process:

Testing whether a family contains the empty set and testing whether after adding an E set to an $\mathcal{F}$ family closed by union, $\mathcal{F}$ remains closed by union. The

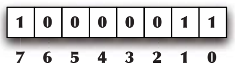

Fig. 3. Let be a universe $U_3 = \{0, 1, 2\}$, the family $\mathcal{F} = \{\emptyset, \{0\}, \{0, 1, 2\}\}$ is encoded by the $2^3 = 8$ bits vector $b[]$ where $b[i] = 1$ if and only if the set E belongs to $\mathcal{F}$ (i being the integer associated with the set E)

first of the two tests becomes trivial when the proposed coding is used. In fact, since the empty set is borne by the weakest bit, a family contains the empty set if and only if the identifier of this family is an odd number. The second, more difficult test requires verifying whether each element of $\mathcal{F}$, the union of this element with E is in $\mathcal{F}$. Nevertheless, this test can be carried out by extracting the following code:

```
for (i = 1 ; i < N ; i++ )
      if ( ( F \& (1U<<i)) \&\& ! ( F \& (1U << (E | i) ) ) )
       printf("F Union E is not union closed");
```

The for loop scans all the characteristic integers i of a set. The first part of the test if $(\mathcal{F}\&(1U << i))$ verifies whether the set corresponding to the integer i belongs to the $\mathcal{F}$ family. For this test, we create a vector made up of 0 except of 1 at the position i using the $(1U << i)$ expression and we carry out an *and* logic with the $\mathcal{F}$ family. This test naturally gives "true" as result if and only if the i set belongs to $\mathcal{F}$. The second part of the test verifies whether $i \cup E$ belongs to the $\mathcal{F}$ family. Since the identifier of the $E \cup i$ set corresponds to the decimal value of $E|i$, it is enough to carry out an *and* logic between $\mathcal{F}$ and a vector made up of 0 except 1 at the $E|i$ position.

2.2 Symmetry and Canonical Form

A permutation Φ on a finite set $U_n = \{0, ..., n-1\}$ is a bijective function from U_n to U_n. For convenience Φ is often represented by the sequence of its images $\Phi(0), \Phi(1), ..., \Phi(n-1)$. All the permutations on a U_n set are marked by Sym_n. For a set $E \subseteq U_n$ and $\Phi \in Sym_n$, we use $\Phi(E)$ to mark the image of E by Φ defined by :$\Phi(E) = \{\Phi(x)|x \in E\}$. Similarly, we simply use $\Phi(\mathcal{F}) = \{\Phi(E)|E \in \mathcal{F}\}$ for all $\mathcal{F} \subseteq 2^{U_n}$.

Example: Let $\mathcal{F}$ be the family $\{\emptyset, \{2\}, \{1, 2\}, \{0, 1, 2\}\}$ and Φ the permutation $1, 2, 0$ then $\Phi(\mathcal{F}) = \{\emptyset, \{0\}, \{0, 2\}, \{0, 1, 2\}\}$.

Using the concept of permutation we can divide all the families on U_n into equivalence classes. Thus we can say that two families belong to the same class if they are the images of one another by a Sym_n permutation. The reference family of each class is called the **canonical form**. We have used an identification of the canonical form based on the properties of our encoding. Hence, the canonical

Table 2. For $\mathcal{F} = \{\emptyset, \{2\}, \{1,2\}, \{0,1,2\}\}$ and each permutation Φ of Sym_3, the image of $\mathcal{F}$ by Φ and its identifier

$Permutation \Phi$	$\Phi(\mathcal{F})$	Identifier
012	$\{\emptyset, \{2\}, \{12\}, \{012\}\}$	209
021	$\{\emptyset, \{1\}, \{12\}, \{012\}\}$	197
102	$\{\emptyset, \{2\}, \{02\}, \{012\}\}$	177
120	$\{\emptyset, \{0\}, \{02\}, \{012\}\}$	163
210	$\{\emptyset, \{0\}, \{01\}, \{012\}\}$	139
201	$\{\emptyset, \{1\}, \{01\}, \{012\}\}$	141

Table 3. The number of Moore families up to an isomorphism and the average size of classes for each value of n. The average size of classes increases drastically with n (and tends to be close to $n!$) which proves that the set of all the Moore families on a universe contains a large number of isomorphic objects.

n	Number of Moore families up to isomorphism	Average size of classes
1	2	1
2	5	$1,40$
3	19	$3,21$
4	184	$13,48$
5	14664	$94,49$
6	108 295 846	$701,54$

form of a family $\mathcal{F} \subseteq 2^{U_n}$ is defined as the image by one of the permutations of Sym_n having the smallest identifier.

Example: Let us consider $\mathcal{F} = \{\emptyset, \{2\}, \{1,2\}, \{0,1,2\}\}$ family on U_3 as well as the Sym_3 set containing 6 permutations. As per table 2, the canonical form of $\mathcal{F}$ is the $\{\emptyset, \{0\}, \{01\}, \{012\}\}$ family whose identifier is equal to 139.

Proposition 1. *Let $\mathcal{M}$ be a Moore family on U_n and $\Phi \in Sym_n$ be a permutation, then $\Phi(\mathcal{M})$ is a Moore family on U_n.*

The partitioning that exists on the set of families on U_n also pertains to the set of Moore families. Actually, as per the previous property, the image of a Moore family by a permutation remains a Moore family. Therefore, it is possible to enumerate the Moore families by enumerating the representative of each equivalence class and then by counting its images with the help of different permutations. In general, the enumeration of a single representative per equivalence class is called as "enumeration of a set of combination objects up to an isomorphism". This strategy is often based on the observation that a large part of the combinatorial explosion related to all the objects studied can be explained by the presence of isomorphic objects. Table 3 shows that this situation is specially verified for the set of Moore families. Let's note that it's common to use symmetry to decrease combinatorial explosion (see for example [13,14]).

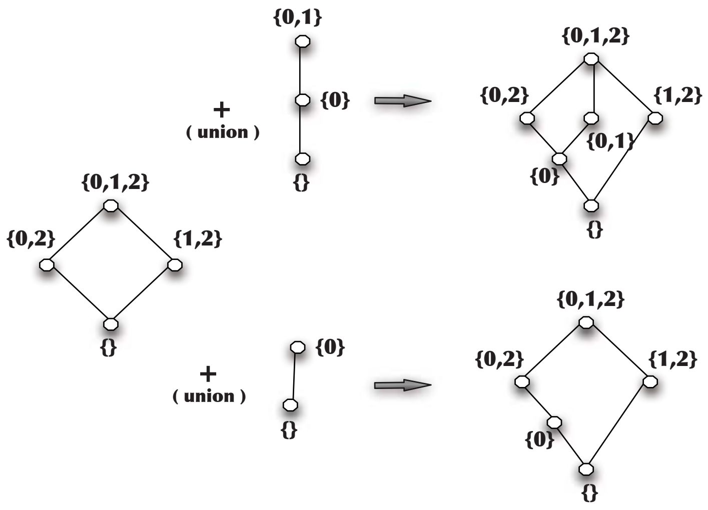

Fig. 4. On left a Moore family on U_3 (such that each element contains 2) associated to two different Moore families on U_{n-1}. In each case, the obtained family, on right, is itself a Moore family.

2.3 Maximal Family

A Moore family $\mathcal{M}$ on U_n can be divided into two parts. The part made up of $\mathcal{M}$ sets containing the $\{n-1\}$ element (marked as $\mathcal{M}_{sup}$ for the upper part) and the additional part (marked as $\mathcal{M}_{inf}$ for the lower part). The $\emptyset$ element is duplicated in order to be included in both parts. Naturally $\mathcal{M} = \mathcal{M}_{sup} \cup \mathcal{M}_{inf}$. Besides, $\mathcal{M}_{sup}$ and $\mathcal{M}_{inf}$ are Moore families. The example in figure 4 proves that there can be many compatible lower parts for a fixed upper part (i.e., their combination gives a Moore family).

There is a unique **maximal family** for a given upper $\mathcal{M}_{sup}$ family as all the families compatible with $\mathcal{M}_{sup}$ are the sub-families of the maximal family. To be more specific:

Proposition 2. *Let $\mathcal{M}_{sup}$ be a Moore family on U_n with $n-1 \in M$ for all $M \in \mathcal{M}_{sup} \setminus \{\emptyset\}$. Then there exists a* ***unique Moore family*** *$\mathcal{M}_{max}$ on U_{n-1} compatible with $\mathcal{M}_{sup}$ such that all Moore families on U_n, whose the restriction to its elements containing $n-1$ corresponds to $\mathcal{M}_{sup}$, can be written $\mathcal{M}_{sup} \cup \mathcal{M}_{inf}$, with $\mathcal{M}_{inf} \subseteq \mathcal{M}_{max}$.*

We can make two remarks : First, The maximal family can be defined as $\mathcal{M}_{max} = \{M \in 2^{U_{n-1}} \mid M \cup M' \text{ for all } M' \in \mathcal{M}\}$. Second the maximal family correspond to the set of the quasi-closed sets that don't contain $\{n-1\}$.

The maximal family associated to the $\mathcal{M}_{sup}$ family given in figure 4, for example, is a family made up of $\{\emptyset, \{0\}, \{1\}, \{0,1\}\}$ elements. We can verify whether the two compatible families given in figure 4 are the sub-families of this family.

Algorithm 1. *maximalFamily*()

```
Data   : $V[2^n]$ which corresponds to the family $\mathcal{M}_{sup}$
Result : $V[2^n]$ which corresponds to the family $\mathcal{M}_{sup} \cup \mathcal{M}_{max}$
begin
    for ($b = 2^n - 1$ to 0) do
        $V[b] \leftarrow 1$;
        if $V[]$ does not corresponds to a union closed family then
            $V[b] \leftarrow 0$;
    return $V[]$;
end
```

Remember, the encoding taken from a Moore family takes the form of a 2^n bit vector (we will use a 128 size vector to count the Moore families on U_7). Naturally the first 2^{n-1} bits encode for all the sets containing the $n-1$ element (i.e. $\mathcal{M}_{sup}$), whereas the last 2^{n-1} bits encode for $\mathcal{M}_{inf}$. Therefore, our counting strategy consists in generating only the upper parts of the vector, and determining the maximal family that is compatible with each of these upper parts (which means calculating the lower part of the vector). Algorithm 1 is a calculation process of this unique family.

Proposition 3. *Let $\mathcal{M}_{sup}$ be a Moore family on U_n (all its elements containing $n-1$), the algorithm 1 computes the maximal family $\mathcal{M}_{max}$ compatible with $\mathcal{M}_{sup}$ and returns the vector which corresponds to the family $\mathcal{M}_{sup} \cup \mathcal{M}_{max}$.*

There are multiple uses of the maximal family concept: Firstly, we have seen that the counting of Moore families of U_7 requires using a 128 bits vector. Unfortunately, integers on 128 bits are not convenient enough to handle directly. Using maximal families enables dividing the calculation of Moore families into two distinct and independent parts by using only the 64 bits vectors. Secondly, it also enables reusing the calculations made while counting the Moore families on U_{n-1} to count the Moore families on U_n. Finally, it helps to highlight a natural bound on the number of Moore families on U_n according to this number on U_{n-1} (refer to proposition 4).

Proposition 4. *Let $\mathtt{M}_n$ and $\mathtt{M}_{n+1}$ be the sets of all Moore family respectively on U_n and U_{n+1} then we have $\mid \mathtt{M}_{n+1} \mid \leq 2 * \mid \mathtt{M}_n \mid^2$.*

The following section concentrates on the implementation of these concepts in an algorithmic framework.

3 Algorithms

This section describes three different algorithms of counting the Moore families on U_n. The first algorithm is a naïve recursive algorithm that scans a tree representing the set of Moore families. The second algorithm introduces the concept

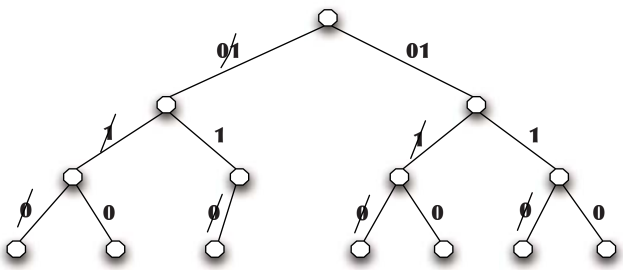

Fig. 5. Binary tree corresponding to the exploration of algorithm 2

Algorithm 2. *Moore*$_1$()

```
Data   : $V[]$: bit vector; $k$ :integer
begin
    if $k = 0$ then
        $|\mathtt{M}_n| \leftarrow |\mathtt{M}_n| + 1$;
    else
        $Moore_1(V[], k-1)$;
        $V[k] \leftarrow 1$;
        if $V[]$ corresponds to an union closed family then
            $Moore_1(V[], k-1)$
        $V[k] \leftarrow 0$;
end
```

of symmetry by storing the unique representative of each class in a hash table. A "coefficient" variable associated to each class records the size of the class. Lastly, the third algorithm integrates the concept of maximal family in the symmetry concept. Note that only the last algorithm gives reasonable calculation time for $n = 7$.

3.1 Naïve Algorithm

The algorithm 2 completes a recursive scan of the set of Moore families on U_n. The medium for this scan is a tree having Moore families as leaves. It stops when a given set is present or absent in the family. The tree structure of Moore families on U_2 is given in figure 5. Thus the algorithm generates a new set at every node and determines whether it can be added to the current family. If the answer is "No", the process continues on the left branch with the same family. Otherwise, the process is restarted on the right branch with a family integrating the new element. When a leaf is reached, the value of a global variable ($|\mathtt{M}_n|$), counting the total number of families, increases.

Technically, the current family is stored in the $V[]$ vector (initialised to 0) which is updated from left to right. k corresponds to the position of the bit of

$V[]$, i.e. to the set which is likely to be inserted in the family. k initialised to $2^n - 1$ decreases naturally till 0 which corresponds to a leave of the tree. In this way, the sets are considered in a linear order whose first element is the $\{0, 1, ..., n-1\}$ set and the last element is the empty set.

3.2 Algorithm Using the Symmetries

The algorithmic principle given here integrates the concept of symmetries and stores the canonical forms in a hash table by associating a *coefficient* to each of these forms. Inserting a canonical form coupled with a c coefficient in the table involves creating a new input if the form is not known or increasing the coefficient of the existing form by c.

To calculate the canonical form of the Moore family $\mathcal{M}$, the function $canonicalForm(\mathcal{M})$ generates $\mathcal{M}$ images by all the permutations of Sym_{U_n} and returns the image of the smallest identifier. These small values of n (at the most 7) provide reasonable calculation time for a complexity of $O(n!)$. Finally we can notice that the problem of the canonical form calculation comes down to the problem of graph isomorphism.

Our strategy can be represented using two algorithms:
The first $fillVector()$ (refer to algorithm 3) enables assigning bits to the $V[]$ vector between the k and m positions in such a way that$V[]$ becomes a family closed by union. It is an adaptation of the $Moore_1$ algorithm for which the stop bit (corresponding to a leave) can be parameterised. The second algorithm (refer to algorithm 4) uses the information stored in the hash table at each step to effectively construct the new vectors which can correspond to the Moore families. At the end of the calculation, it is enough to only add the coefficients of each canonical form present in the latest table to find the total number of families.

Let us note that the c coefficient associated to a $V[]$ vector when it is inserted in the hash table is the coefficient associated to the vector which was used when the $fillVector()$ algorithm was called in $Moore_2$ process.

Algorithm 3. $fillVector()$

```
Data   : $V[]$ : bit vector; $k$,$m$ : integers;
begin
    if $k = m - 1$ then
        $myHashTable.add(canonicalForm(V[]))$;
    else
        $fillVector(V[], k - 1)$;
        $V[k] \leftarrow 1$;
        if $V[]$ corresponds to an union closed family then
            $fillVector(V[], k - 1)$;
        $V[k] \leftarrow 0$;
end
```

Table 4. Sizes of the hashtable containing Moore families corresponding to different values of n before and after reduction

n	$\|H\|$ before reduction	$\|H\|$ after reduction
2	7	5
3	46	19
4	916	184
5	140 463	14 664
6	770 413 085	108 295 846

Algorithm 4. *moore*$_2$()

```
Data   : $n$ : integer;
Result : integer / number of Moore families on $U_n$;
begin
    $H'.add(V[] =< 0, 0, ..., 0, 0, 1 >)$;
    $H \leftarrow \emptyset$;
    $i \leftarrow 1$;
    while $i \leq n$ do
        for $V[] \in H'$ do
            $fillVector(V[], 2^i - 1, 2^{i-1})$;
        $i++$;
        $H' \leftarrow H$;
    return $\Sigma_{V[] \in H} V[].coefficient$;
end
```

Table 4 recapitulates the size of the table containing the families generated for each value of n. The size is given before and after the reduction. Naturally, we can find the number of Moore families up to an isomorphism in the last column. Finally, let us remember that the hash table before reduction is never constructed since the reduction takes place with the help of the *canonicalForm*() function as the process progresses.

3.3 Algorithm Using the Maximal Families

The algorithm shown here integrates the principle of maximal family (refer to the previous section) with the concept of symmetry that we have just implemented. At first, an adaptation of the previous algorithm is used to generate the vectors representing the $\mathcal{M}_{sup}$ families up to an isomorphism (the number of families isomorphic to each of these families is stored in the c coefficient). Secondly, we determine the associated $\mathcal{M}_{inf}$ maximal family for each $\mathcal{M}_{sup}$ family. Let us note that a $\mathcal{M}_{inf}$ family is a Moore family on U_{n-1} and its weight p corresponds to the number of Moore families included in this family. *Hence, the counting of the Moore families on U_n corresponds to the sum of the product between the weights and the coefficient of each selected family.*

Algorithm 5. $leftPart()$

```
Data   : $n$ : integer;
Result : $H$;
begin
    $H'.add(V =< 0, 0, ..., 0, 0, 1 >)$;
    $H \leftarrow \emptyset$;
    $i \leftarrow 1$;
    while $i \leq n - 1$ do
        for $V \in H'$ do
            $FillVector(V, 2^i - 1 + 2^{n-1}, 2^{i-1} + 2^{n-1})$;
        $n++$;
        $H' \leftarrow H$;
    return $H$;
end
```

The input of the $maximalFamily()$ function is a $V[]$ vector of 2^n bits that represents a Moore family on U_n as all its elements contain the $n-1$ element. Therefore, all the bits of $V[]$ included between 1 and $2^{n-1} - 1$ are positioned at 0. The function returns a vector of 2^{n-1} bits representing the Moore family on U_{n-1} that corresponds to the maximal family associated to the considered Moore family.

The $leftPart()$ algorithm (refer to algorithm 5) generates vectors up to an isomorphism corresponding to the Moore families on U_n by considering only the $2^{U_n} \setminus 2^{U_{n-1}} \setminus \{n-1\}$ elements. The $Moore2$ algorithm given earlier is adapted considering only the bits included between $2^n - 1$ and $2^{n-1} + 1$.

Algorithm 6. $moore_3()$

```
Data   : $n$ : integer; $\mathcal{M}_{n-1}$;
Result : $|\mathtt{M}_n|$ the number of Moore family on $U_n$;
begin
    $|\mathtt{M}_n| \leftarrow 0$;
    $H \leftarrow leftPart(n)$;
    for $V[] \in H$ do
        $|\mathtt{M}_n| \leftarrow |\mathtt{M}_n| + canonicalForm(maximalFamily(V[])).weight * V[].coefficient$;
        $V[2^{n-1}] \leftarrow 1$;
        $|\mathtt{M}_n| \leftarrow |\mathtt{M}_n| + canonicalForm(maximalFamily(V[])).weight * V[].coefficient$;
    retourner $|\mathtt{M}_n|$;
end
```

The $Moore3()$ algorithm (refer to algorithm 6) starts by generating the left parts of the vectors by using the $leftPart()$ algorithm. Two Moore families are associated to each left part: one family containing the $n-1$ element and another not containing this element. Then the algorithm associates a maximal family to each Moore family. The number of Moore families on U_n are calculated by accumulating the results of the coefficients of the $\mathcal{M}_{sup}$ families by the weights of corresponding $\mathcal{M}_{inf}$ families as the process progresses. The algorithm uses a $\mathcal{M}_{n-1}$ dictionary containing the weights of Moore families on U_{n-1}. The weight of a family can be calculated with the help of the $computeWeight()$ algorithm (refer to algorithm 7) which functions conversely to the $fillVector()$ algorithm.

Algorithm 7. $computeWeight()$

```
Data   : $V[]$ : vector, $k$ : integer;
Result : $p$;
begin
    if $k = 2^n$ then
        $p \leftarrow p + 1$;
    else
        $computeWeight(V[], k + 1)$;
        $V[k] \leftarrow 0$;
        if $V[]$ corresponds to a union closed family then
            $computeWeight(V[], k + 1)$
end
```

In case of the calculation of number of Moore families on U_7, the number of left parts up to an isomorphism is 108295846 (it corresponds to the number of Moore families on U_6 up to an isomorphism). We find 118540742 maximal families and 108295846 maximal families up to an isomorphism. In other words, each Moore family on U_6 is, at least one time, a maximal family of a family on U_7.

4 Technical Aspects

4.1 Implementation

This program is written in C, compiled with Gcc version 4 for Linux 64 bits. We have used the OpenMP parallel programming library. The main calculation lasted 11 hours on a 2.5 GHz Core2 Q9300. The verification lasted 9 hours on a 1.86 GHz double Xeon E5320 (University of Bogotá). The calculation was simultaneously done on a 2.3 GHz Opteron 8356 quadruplet (LIMOS).

4.2 Reliability

The question of reliability is of primordial importance for any result involving a large number of computer calculations. In this case, the authors would like to highlight the following points:

– The result complies with the previous estimations of one of the authors (between $1,2.10^{19}$ and $1,7.10^{19}$) based on an extrapolation of data for $n \leq 6$.
– There is no specific code at $n = 7$ and no special threshold between $n = 6$ and $n = 7$ (for example, size of integers used)
– The same program gives good results for n ranging from 1 to 6 (number of Moore families and the families reduced by isomorphism).
– The code was executed on numerous occasions on different architectures (Intel and AMD).
– The program was compiled with three stable compilers (Gcc v. 4.1, 4.2 and 4.3) and various options on different operating systems (Ubuntu and CentOS).
– The result remained stable at each modification of the size available for the hash tables. Let us note that the modification in the table size badly disrupts the time required for the calculation to complete."
– The result remains unchanged by replacing the canonical form based on the smallest representative of the equivalence class with a canonical form based on a larger representative.

5 Conclusion

The problem of enumerating the Moore families in the n order is a complex issue for which there is no known formula. Even the absence of such a formula has not been proved. Numerous combinatory problems fall in the same case. For example, the number of monotonous Boolean functions known as the Dedekind number. An often supported approach to comprehend such formulae involves counting the number of objects for the first values of n using a systematic procedure. We can find such integer sequences on the well-known On-line Encyclopaedia of Integer Sequences. Our work thus had a dual-objective: Not only enriching the already-known sequence for the number of Moore families, but also highlighting the new properties of the set of Moore families.

In this article, we have proved that we can calculate the search objects without enumerating all of them. Therefore, there is a smaller frame of objects (the Moore families on U_n up to an isomorphism in which all the sets include $n-1$) from which the total number can be deduced. In other words, for each object of the frame it is sufficient to calculate its associated maximal family (a Moore family on U_{n-1}) and calculate its weight (i.e. the number of Moore families included). This enumeration is carried out only once even if the maximal family is found many times. Since the result is stored in a hash table that uses the identifier of the maximal family as its input, we can also affirm that each Moore family on U_{n-1} appears at least once as a maximal family.

The possibility of predicting how many times each family on U_{n-1} appears as maximal family of a family on U_n will bring us significantly close to finding a general formula, if any such formula exists.

Acknowledgment

We are grateful to Bernhard Ganter to have pointed out the misprint on the Integer Sequence Encyclopedia, and consequently confirmed our results on the number of Moore families up to isomorphism.

References

1. Habib, M., Nourine, L.: The number of moore family on n=6. Discrete Mathematics 294, 291–296 (2005)
2. Cohn, P.: Universal Algebra. Harper and Row, New York (1965)
3. Davey, B.A., Priestley, H.A.: Introduction to lattices and orders, 2nd edn. Cambridge University Press, Cambridge (1991)
4. Demetrovics, J., Libkin, L., Muchnik, I.: Functional dependencies in relational databases: A lattice point of view. Discrete Applied Mathematics 40(2), 155–185 (1992)
5. Duquenne, V.: Latticial structure in data analysis. Theoretical Computer Science 217, 407–436 (1999)
6. Ganter, B., Wille, R.: Formal concept analysis. Mathematical Foundation. Springer, Heidelberg (1999)
7. Birkhoff, G.: Lattice Theory, 3rd edn. American Mathematical Society (1967)
8. Moore, E.: Introduction to a form of general analysis. Yale University Press, New Haven (1910)
9. Caspard, N., Monjardet, B.: The lattices of closure systems, closure operators, and implicational systems on a finite set: a survey. Discrete Applied Mathematics 127, 241–269 (2003)
10. Burosh, G., Demetrovics, J., Katona, G., Kleitman, D., Sapozhenko, A.: On the number of databases and closure operations. Theoretical Computer Science 78, 377–381 (1991)
11. Higuchi, A.: Note:lattices of closure operators. Discrete Mathematics 179, 267–272 (1998)
12. Medina, R., Nourine, L.: Algorithme efficace de génération des ideaux d'un ensemble ordonné. Compte rendu de l'Académie des sciences, Paris T.319 série I, pp. 1115–1120 (1994)
13. McKay, B.D.: Isomorph-free exhaustive generation. J. Algorithms 26(2), 306–324 (1998)
14. Ganter, B.: Finding closed sets Under Symmetry. FB4-PrePrint 1307, TH Darmstadt (1990)

Appendix

Proposition 1. *Let $\mathcal{M}$ be a Moore family on U_n and $\Phi \in Sym_n$ be a permutation, then $\Phi(\mathcal{M})$ is a Moore family on U_n.*

Proof. Let us show that $\Phi(\mathcal{M})$ is a Moore family.

- $\emptyset \in \mathcal{M}$, $\Phi(\emptyset) = \emptyset$ thus $\emptyset \in \Phi(\mathcal{M})$.
- Let M_1 and $M_2 \in \mathcal{M}$ and $\Phi(M_1)$ and $\Phi(M_2) \in \Phi(\mathcal{M})$. We have $\Phi(M_1) \cup \Phi(M_2) = \Phi(M_1 \cup M_2)$ by definition of $\Phi()$. Since $\mathcal{M}$ is closed $M_1 \cup M_2 \in \Phi(\mathcal{M})$. Then $\Phi(M_1 \cup M_2) \in \Phi(\mathcal{M})$.

So $\forall \Phi(M_1)$ and $\Phi(M_2) \in \Phi(\mathcal{M})$, $\Phi(M_1) \cup \Phi(M_2) \in \Phi(\mathcal{M})$.

$\Phi(\mathcal{M})$ contains the empty set and is closed by union, $\Phi(\mathcal{M})$ is a Moore family.

$\square$

Proposition 2. *Let $\mathcal{M}_{sup}$ be a Moore family on U_n with $n-1 \in M$ for all $M \in \mathcal{M}_{sup} \setminus \{\emptyset\}$. Then there exists a* ***unique Moore family*** *$\mathcal{M}_{max}$ on U_{n-1} compatible with $\mathcal{M}_{sup}$ such that all Moore families on U_n, whose the restriction to its elements containing $n-1$ corresponds to $\mathcal{M}_{sup}$, can be written $\mathcal{M}_{sup} \cup \mathcal{M}_{inf}$, with $\mathcal{M}_{inf} \subseteq \mathcal{M}_{max}$.*

Proof. Let $\mathcal{M}$ be a Moore family on U_n with $\mathcal{M} = \mathcal{M}_{sup} \cup \mathcal{M}_{inf}$.

A) Let us show that for any Moore family $\mathcal{M'}_{inf}$ on U_{n-1} with $\mathcal{M'}_{inf} \subseteq \mathcal{M}_{inf}$ then $\mathcal{M}_{sup} \cup \mathcal{M'}_{inf}$ is a Moore family.

- $\emptyset \in \mathcal{M'}_{inf}$, then $\emptyset \in \mathcal{M}_{sup} \cup \mathcal{M'}_{inf}$;
- Let M_1 and $M_2 \in \mathcal{M}_{sup} \cup \mathcal{M'}_{inf}$; 3 cases can occur :
  1. If M_1 and $M_2 \in \mathcal{M}_{sup}$, since $\mathcal{M}_{sup}$ is a Moore family $M_1 \cup M_2 \in \mathcal{M}_{sup}$ and then $M_1 \cup M_2 \in \mathcal{M}_{sup} \cup \mathcal{M'}_{inf}$;
  2. If M_1 and $M_2 \in \mathcal{M'}_{inf}$, since $\mathcal{M'}_{inf}$ is a Moore family $M_1 \cup M_2 \in \mathcal{M'}_{inf}$ and then $M_1 \cup M_2 \in \mathcal{M}_{sup} \cup \mathcal{M'}_{inf}$;
  3. If $M_1 \in \mathcal{M}_{sup}$ et $M_2 \in \mathcal{M'}_{inf}$. Since $\mathcal{M'}_{inf} \subseteq \mathcal{M}_{inf}$ and since for all $M \in \mathcal{M}_{inf}$, $M_1 \cup M_2 \in \mathcal{M}_{sup}$ ($M \cup M_1$ contains n) then $M_1 \cup M_2 \in \mathcal{M}_{sup}$. And so $M_1 \cup M_2 \in \mathcal{M}_{sup} \cup \mathcal{M'}_{inf}$;

B) Let us show that for all Moore families $\mathcal{M'}_{inf}$ and $\mathcal{M''}_{inf}$ on U_{n-1} such that $\mathcal{M}_{sup} \cup \mathcal{M'}_{inf}$ and $\mathcal{M}_{sup} \cup \mathcal{M''}_{inf}$ are Moore families then $C(\mathcal{M'}_{inf} \cup \mathcal{M''}_{inf}) \cup \mathcal{M}_{sup}$ is a Moore family with C an union closed operator.

- $\emptyset \in \mathcal{M'}_{inf}$, then $\emptyset \in C(\mathcal{M'}_{inf} \cup \mathcal{M''}_{inf}) \cup \mathcal{M}_{sup}$;
- Let M and $M' \in C(\mathcal{M'}_{inf} \cup \mathcal{M''}_{inf}) \cup \mathcal{M}_{sup}$; The only difficult case corresponds to the case $M \in \mathcal{M}_{sup}$ and $M' \in C(\mathcal{M'}_{inf} \cup \mathcal{M''}_{inf}) \setminus \mathcal{M'}_{inf} \setminus \mathcal{M''}_{inf}$. Since $\mathcal{M'}_{inf}$ and $\mathcal{M''}_{inf}$ are Moore families then there exists m_1 in $\mathcal{M'}_{inf}$ and m_2 in $\mathcal{M''}_{inf}$ such that $M' = M_1 \cup M_2$ and :

1. if $M_1 \in \mathcal{M}'_{inf}$ then $\forall M \in \mathcal{M}_{sup}$ we have $M \cup M_1 \in \mathcal{M}_{sup}$;
2. if $M_2 \in \mathcal{M}''_{inf}$ then $\forall M \in \mathcal{M}_{sup}$ we have $M \cup M_2 \in \mathcal{M}_{sup}$;

And since $\mathcal{M}_{sup}$ is a Moore family $M_1 \cup M_2 \cup M = M' \cup M$ belongs to $\mathcal{M}_{sup}$

We conclude that for all couple of elements M and M' in $C(\mathcal{M}'_{inf} \cup \mathcal{M}''_{inf}) \cup \mathcal{M}_{sup}$, $M \cup M'$ belongs to $\mathcal{M}_{sup}$. So $C(\mathcal{M}'_{inf} \cup \mathcal{M}''_{inf}) \cup \mathcal{M}_{sup}$ is a Moore family.

From B) we have that the set of all compatible Moore families with a given family $\mathcal{M}_{sup}$ owns only one maximal element denoted $\mathcal{M}_{max}$ such that for all families $\mathcal{M}_{inf}$ included in or equal to $\mathcal{M}_{max}$, the join of $\mathcal{M}_{sup}$ and $\mathcal{M}_{inf}$ is a Moore family. □

Proposition 3. *Let $\mathcal{M}_{sup}$ be a Moore family on U_n (all its elements containing n), the algorithm 1 computes the maximal family $\mathcal{M}_{max}$ compatible with $\mathcal{M}_{sup}$ and returns the vector which corresponds to the family $\mathcal{M}_{sup} \cup \mathcal{M}_{max}$.*

Proof. First of all let us say the the order in which element are considered follows the property : Let $M \in 2^{U_n}$, for all $M' \in 2^{U_n}$ the element $M \cup M'$ is processed before M. If $M' \subset M$ we have $M = M \cup M'$.

Let $\mathcal{M}_{max}$ be a Moore family produced by the algorithm. Let us show that $\mathcal{M}_{max}$ is a Moore family and is maximal. By construction $\mathcal{M}_{max}$ is a Moore family. Consider the following assertion: to each step of the algorithm, if a set $M \in 2^{U_n}$ is not added to $\mathcal{M}_{max}$ then $\mathcal{M}_{sup} \cup \mathcal{M}_{max} \cup M$ is not a Moore family. Let us demonstrate the assertion reductio ad absurdum : Suppose that there exists a set $M \notin \mathcal{M}_{max}$ such that $\mathcal{M}_{sup} \cup \mathcal{M}_{max} \cup M$ is a Moore family. Then there exists a set M' processed before M, and a set M'' processed after M with $M'' = M \cup M'$. Which is in contradiction with the process' computational order. Absurd. □

Proposition 4. *Let $\mathtt{M}_n$ and $\mathtt{M}_{n+1}$ be the sets of all Moore family respectively on U_n and U_{n+1} then we have $\mid \mathtt{M}_{n+1} \mid \leq 2 * \mid \mathtt{M}_n \mid^2$.*

Proof. Each family $\mathcal{M} \in \mathtt{M}_n$ is going to produce two superior families $\mathcal{M}_{sup_1} = \{M \cup n \mid M \in \mathcal{M}\} \cup \emptyset$ and $\mathcal{M}_{sup_2} = \{M \cup n \mid m \in \mathcal{M}\} \cup \emptyset \setminus n$. Therfore, the set $\mathtt{M}_{sup_{n+1}}$ of all families $\mathcal{M}_{sup}$ contains $2* \mid \mathtt{M}_n \mid$ elements. In the worst case the maximal family associated to each family $\mathcal{M}_{sup} \in \mathtt{M}_{sup_{n+1}}$ is the Moore family 2^{U_n} of weight $\mid \mathtt{M}_n \mid$. So $\mid \mathtt{M}_{n+1} \mid \leq 2 * \mid \mathtt{M}_n \mid^2$. □