Recursive decomposition tree of a Moore co-family and closure algorithm

Pierre Colomb, Alexis Irlande, Olivier Raynaud, Yoan Renaud

Noname manuscript No.
(will be inserted by the editor)

Recursive decomposition tree of a Moore co-family and closure algorithm

Pierre Colomb · Alexis Irlande · Olivier Raynaud · Yoan Renaud

Received: date / Accepted: date

Abstract A collection of sets on a ground set U_n ($U_n = \{1, 2, ..., n\}$) closed under intersection and containing U_n is known as a Moore family. The set of Moore families for a fixed n is in bijection with the set of Moore co-families (union-closed families containing the empty set) denoted $\mathbb{M}_n$. This paper follows the work initiated in [9] in which we have given an inductive definition of the lattice of Moore co-families. In the present paper we use this definition to define a recursive decomposition tree of any Moore co-family and we design an original algorithm to compute the closure under union of any family. Then we compare performance of this algorithm to performance of Ganter's algorithm and Norris' algorithm.

Keywords Closure systems · Enumeration algorithm · Lattice theory

Pierre Colomb
Clermont University, University Blaise Pascal, Campus des Cézeaux 63173 Clermont-Ferrand, France
E-mail: pierre@colomb.me

Alexis Irlande
Universidad Nacional de Colombia Bogotá, Colombia
E-mail: irlande@lirmm.fr

Olivier Raynaud
Clermont University, University Blaise Pascal, Campus des Cézeaux 63173 Clermont-Ferrand, France
E-mail: raynaud@isima.fr

Yoan Renaud
INSA de Lyon, Bâtiment Blaise Pascal, Campus de la Doua, 69621 Villeurbanne, France
E-mail: yoan.renaud@insa-lyon.fr

1 Introduction

The concept of a collection of sets closed under intersection appears with different names depending on the scientific fields. The name *Moore family* was first used by Birkhoff in [4] referring to E.H. Moore's research. But, very frequently, such a collection on a ground set U_n ($U_n = \{1, 2, ..., n\}$) is called closure system. This concept is applied to numerous fields in pure or applied mathematics and computer science. For instance Cohn, Sierksma and van de Vel have used it in the framework of algebra and topology ([10,23,24]) while Birkhoff, Davey and Priestley focused on order and lattice theory ([3,14]). Formally a closure operator is an extensive, isotone and idempotent function on 2^{U_n} (the set of all subsets of U_n), and a closure system is then the set of its fixed points. In particular, it is well-known that any closure system is a complete lattice. In 1937, Birkhoff ([3]) gave a compact representation of *quasi-ordinal spaces* (in other words, collections of sets closed under intersection and union and so which are distributive lattices). More recently the notion of closure system appears as a significant concept in computer science with research in relational databases ([12]), data analysis and formal concept analysis ([16,1,17]). More precisely, Ganter and Wille defined a mathematical framework for classification, and Barbut defined and used Galois lattices about questions raised in Guttman scales analysis ([1]). Meanwhile, in 1985, equivalent collections of sets were called *knowledge spaces* by Doignon and Falmagne ([11]). In that context, a set of the collection is a possible state of knowledge of a student following a specific discipline in an educational setting and such that each element of the state is some elementary fact or question known by the student. The hypothesis is that the union of any set of states is itself a knowledge state.

An important fact is that the collection of Moore families on U_n is itself a closure system. Indeed, the system composed of Moore families on U_n contains a maximum element (2^{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 the survey [7]. There exists a one-to-one mapping[1] between the set of Moore families and the set of Moore co-families defined dually as the set of union-closed collections of subsets of U_n containing the empty set. The latter is denoted by $\mathbb{M}_n$.

Some researches focus on quantitative properties of this lattice of Moore co-families. As an example, Demetrovics *et al.* in [15] noticed that the problem of counting Moore co-families on n elements is a complex issue for which there is no known formula. Even the lack of such a formula has not been proved. In [6], Burosh *et al.* consider the issue of counting Moore families as natural, so they provide an upper bound for that number (see also [13]). An often supported approach to try to obtain such a formula involves counting the number of

[1] Basically, for a given Moore family, one only has to complement every set to obtain a Moore co-family. For example, the Moore family $\{\{1\}, \{1,2\}, \{1,3\}, \{1,2,3\}\}$ on U_3 corresponds to the Moore co-family $\{\emptyset, \{2\}, \{3\}, \{2,3\}\}$ and vice versa.

objects for the first values of n using a systematic procedure. In [20], Habib and Nourine computed this number for $n = 6$ and in [8], Colomb *et al.* found 10^{19} families for $n = 7$. We can find such integer sequences on the well-known On-line Encyclopedia of Integer Sequences[2].

From this quantitative study of the lattice, the same authors were able to state an inductive definition of the lattice (see [9]). In the present paper we define a recursive decomposition tree of a Moore co-family and then we design an original algorithm to compute the closure of any given family. Then we compare the performance of this algorithm to performance of Ganter's algorithm and Norris' algorithm. The paper proceeds as follows. Section 2 introduces the inductive definition of the lattice of Moore co-families. Section 3 describes how each Moore co-family can be represented by a decomposition tree relying on the inductive definition (see Theorem 1). In section 4 we introduce the new algorithm and we give practical performance in section 5.

In the rest of the paper, we denote elements by numbers $(1, 2, 3, \dots)$. Sets are denoted by capital letters $(A, B, C, \dots)$. Families of sets are denoted by calligraphic letters $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \dots)$. Finally, we denote the sets of families of sets by black board letters $(\mathbb{A}, \mathbb{B}, \mathbb{C}, \dots)$.

2 Inductive definition of the lattice of Moore co-families

As explained in the introduction, the set of Moore co-families $\mathbb{M}_n$ on a ground set $U_n = \{1, 2, ..., n\}$ has a lattice structure. We give an example of such a lattice with the set of Moore co-families on U_2 in Figure 1. In this section we will show how to define the lattice $\mathbb{M}_{n+1}$ from the lattice $\mathbb{M}_n$ (see [9] and [2] for a complete study).

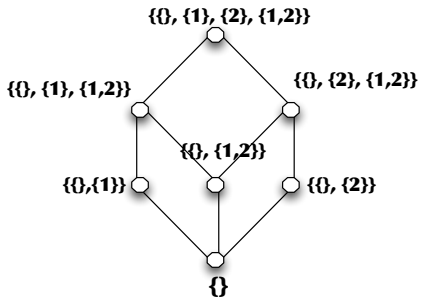

Fig. 1 The Hasse diagram of the complete set $\mathbb{M}_2$ of Moore co-families on the ground set U_2, ordered by inclusion. There are exactly 7 Moore co-families on U_2. This ordered set $(\mathbb{M}_2, \subseteq)$ (or $\mathbb{M}_2$ for short) has a maximal element 2^{U_2} (all subsets of U_2) and the intersection of two Moore co-families is a Moore co-family. Thus, $\mathbb{M}_2$ admits a complete lattice structure.

[2] http://oeis.org/A102896

2.1 Definitions and notations

For any integer $n \geq 1$, let U_n denote the set $\{1, \ldots, n\}$. Let $\mathcal{M}$ be a family on U_n, we denote $(\mathcal{M}, \subseteq)$ the corresponding ordered set. Two sets M, M' in $\mathcal{M}$ such that neither $M \subseteq M'$ nor $M' \subseteq M$ are said *incomparable* in $(\mathcal{M}, \subseteq)$. A family where every pair of sets is incomparable is called an antichain. We say that M is covered by M' in $(\mathcal{M}, \subseteq)$, denoted $M \prec M'$, if $M \subset M'$ and there is no $M'' \in \mathcal{M}$ such that $M \subset M''$ and $M'' \subset M'$. In the following, depending of the context, we will denote $\mathcal{M}$ the ordered set $(\mathcal{M}, \subseteq)$ when no confusion is possible.
Given a family $\mathcal{M}$, a subfamily $\mathcal{I}$ of $\mathcal{M}$ is an *ideal* of $\mathcal{M}$ if it satisfies the following implication for any pair M, M' in $\mathcal{M}$,

$$M \subseteq M' \text{ and } M' \in \mathcal{I} \Rightarrow M \in \mathcal{I}.$$

In other words, an ideal of $\mathcal{M}$ is some antichain of $\mathcal{M}$ and everything below it. We shall use $\mathbb{I}_{\mathcal{M}}$ to denote the sets of ideals on $\mathcal{M}$. Given a set X in a family $\mathcal{M}$, there exists a unique ideal $\mathcal{I}$ of $\mathcal{M}$ with X as a maximum set (also called the *principal ideal* generated by X in $\mathcal{M}$). Let $\mathcal{I}_{\mathcal{M}}(X)$ denote this ideal. By extension, we denote $\mathbb{I}_{\mathbb{M}_n}(\mathcal{M})$ the principal ideal generated by $\mathcal{M}$ in the lattice $\mathbb{M}_n$. This way, $\mathbb{I}_{\mathbb{M}_n}(\mathcal{M})$ corresponds to the set of Moore co-families included in $\mathcal{M}$.

2.2 Compatible families

A Moore co-family $\mathcal{M}$ on U_{n+1} can be decomposed into 2 parts. The part consisting of the sets of $\mathcal{M}$ containing the element $n+1$ (denoted by $\mathcal{M}_{up}$ for the upper part), and the complementary part (denoted by $\mathcal{M}_{low}$ for the lower part). The empty set is duplicated to be present in the two parts. Naturally, $\mathcal{M} = \mathcal{M}_{up} \cup \mathcal{M}_{low}$. On the one hand, the family $\mathcal{M}_{low}$ is clearly a family of $\mathbb{M}_n$. On the other hand, the family $\mathcal{M}_{up}$ is a Moore co-family on U_{n+1} with the peculiarity that all its sets contain the element $n+1$ (we will denote $\mathbb{M}^{up}_{n+1}$ as the set of Moore co-families having this property). Thus, any Moore co-family in $\mathbb{M}^{up}_{n+1}$ can be seen as a Moore co-family in $\mathbb{M}_n$ for which the element $n+1$ has been added to each set.

Example: Let $\mathcal{M}$ be a Moore co-family on U_3:

$$\mathcal{M} = \{\emptyset, \{1\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$$

We can decompose this family into two sub Moore co-families:

$$\mathcal{M}_{low} = \{\emptyset, \{1\}, \{1,2\}\} \text{ and } \mathcal{M}_{up} = \{\emptyset, \{1,3\}, \{2,3\}, \{1,2,3\}\}$$

To study the matching conditions between a family in $\mathbb{M}_n$ and a family in $\mathbb{M}^{up}_{n+1}$ in order for their union to be a family in $\mathbb{M}_{n+1}$, we define the notion of *compatible* family. Thus, we will say that a family in $\mathbb{M}_n$ is *compatible* with a family in $\mathbb{M}^{up}_{n+1}$ if the union of both families is a Moore co-family in $\mathbb{M}_{n+1}$. The

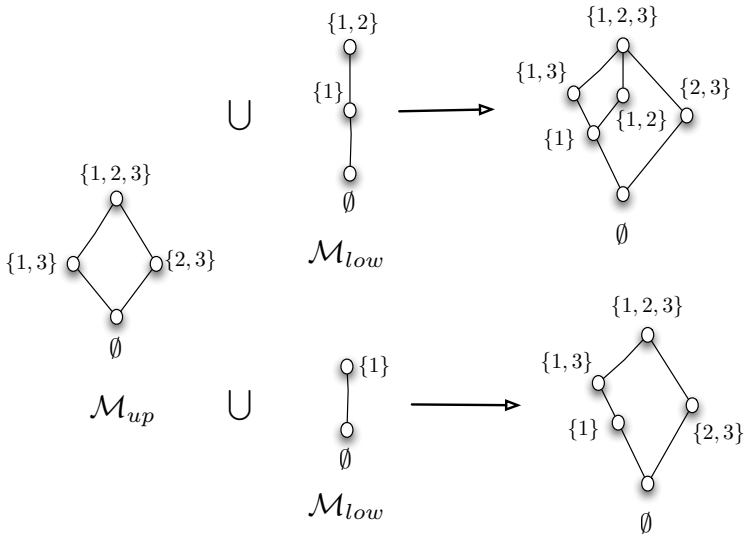

Fig. 2 On the left, a Moore co-family in $\mathbb{M}_3^{up}$ (all its non empty sets contain the element 3) and two different compatible Moore co-families on U_2. In both cases, the obtained family, on the right, is itself a Moore co-family.

example in Figure 2 illustrates that for a fixed upper part, there are several *compatible* lower parts.

In [8] we showed that for a given upper family $\mathcal{M}_{up}$ in $\mathbb{M}_{n+1}^{up}$, there exists a unique compatible *maximal family* and that the set of compatible families with $\mathcal{M}_{up}$ corresponds to the set of the Moore co-families contained in this maximal family. Function f_n defined hereafter is a characterization of the maximal compatible family of a given Moore co-family in $\mathbb{M}_{n+1}^{up}$.

Definition 1 Let f_n be a function from $\mathbb{M}_{n+1}^{up}$ to $\mathbb{M}_n$ such that:
$$f_n(\mathcal{M}) = \{X \in 2^{U_n} \mid \forall M \in \mathcal{M} \setminus \emptyset,\ M \cup X \in \mathcal{M}\}$$

We will write f instead of f_n when no confusion is possible.

Proposition 1 *[8]*
Let $\mathcal{M}_{up}$ be a Moore co-family in $\mathbb{M}_{n+1}^{up}$, for any Moore co-family $\mathcal{M}_{low}$ in $\mathbb{M}_n$, the two following assertions are equivalent:

(i) $\mathcal{M}_{low}$ is compatible with $\mathcal{M}_{up}$;
(ii) $\mathcal{M}_{low} \subseteq f_n(\mathcal{M}_{up})$.

For example, the maximal family associated with $\mathcal{M}_{up}$ (cf. Figure 2) is the family $\{\emptyset, \{1\}, \{2\}, \{1,2\}\}$. Basically, it can be verified that the two compatible families called $\mathcal{M}_{low}$ and given in Figure 2 are sub-families of this family.

Corollary 1 *Let $\mathcal{M}_{up}$ be a Moore co-family in $\mathbb{M}^{up}_{n+1}$, the set of Moore co-families compatible with $\mathcal{M}_{up}$ is $\mathbb{I}_{\mathbb{M}_n}(f(\mathcal{M}_{up}))$.*

2.3 Recursive decomposition theorem

Basically, the set $\mathbb{M}^{up}_{n+1}$ can be partitioned into 2 sets: families that do not contain the singleton $\{n+1\}$ and families that contain it. These two sets are in natural bijection with $\mathbb{M}_n$.

Definition 2 For any integer n such that $n \geq 1$, we define

$$\begin{aligned} g_{1,n+1} : \mathbb{M}_n &\longrightarrow \mathbb{M}^{up}_{n+1} \\ \mathcal{M} &\longmapsto \{X \in 2^{U_{n+1}} \mid \exists M \in \mathcal{M} \setminus \{\emptyset\} \text{ such that } X = M \cup \{n+1\}\} \cup \{\emptyset\}, \\ g_{2,n+1} : \mathbb{M}_n &\longrightarrow \mathbb{M}^{up}_{n+1} \\ \mathcal{M} &\longmapsto \{X \in 2^{U_{n+1}} \mid \exists M \in \mathcal{M} \text{ such that } X = M \cup \{n+1\}\} \cup \{\emptyset\}, \end{aligned}$$

We give an illustration in Figure 3.

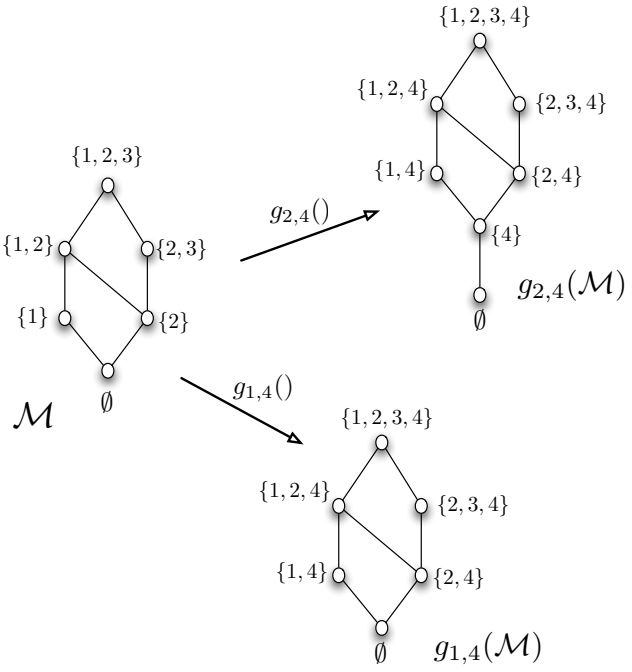

Fig. 3 On the left, a Moore co-family on U_3. At the bottom-right, its image in $\mathbb{M}^{up}_4$ by $g_{1,4}$. One can check that all its sets contain the element 4. Singleton $\{4\}$ does not belong to the family. At the top, its image by $g_{2,4}$ is in $\mathbb{M}^{up}_4$ too and contains the singleton $\{4\}$.

Now, from a given Moore co-family $\mathcal{M}$ in $\mathbb{M}_n$ we can generate two sets of Moore co-families in $\mathbb{M}_{n+1}$.

1. The set of Moore co-families which do not contain the singleton $\{n+1\}$. These families will be under the form $g_{1,n+1}(\mathcal{M}) \cup \mathcal{M}'$ with $\mathcal{M}'$ a compatible Moore co-family of $g_{1,n+1}(\mathcal{M})$. In other words $\mathcal{M}'$ has to be a sub Moore co-family of $f_n(g_{1,n+1}(\mathcal{M}))$. For the convenience of the study, we denote h_n the function from $\mathbb{M}_n$ to $\mathbb{M}_n$ such that $h_n(\mathcal{M}) = f_n(g_{1,n+1}(\mathcal{M}))$. By this way the set of compatible Moore co-families with $g_{1,n+1}(\mathcal{M})$ is exactly $\mathbb{I}_{\mathbb{M}_n}(h_n(\mathcal{M}))$.

2. The set of Moore co-families which contain the singleton $\{n+1\}$. These families will be under the form $g_{2,n+1}(\mathcal{M}) \cup \mathcal{M}'$ with $\mathcal{M}'$ a compatible Moore co-family of $g_{2,n+1}(\mathcal{M})$. In other words $\mathcal{M}'$ has to be a sub Moore co-family of $f_n(g_{2,n+1}(\mathcal{M}))$. But one can check that $f_n(g_{2,n+1}(\mathcal{M}))$ is equal to $\mathcal{M}$. For this reason, the set of compatible families with $g_{2,n+1}(\mathcal{M})$ is exactly $\mathbb{I}_{\mathbb{M}_n}(\mathcal{M})$.

An illustration is given in figure 4. $\mathbb{I}_{\mathbb{M}_n}(h_n(\mathcal{M}))$ (respectively $\mathbb{I}_{\mathbb{M}_n}(\mathcal{M})$) is the principal ideal generated by $h_n(\mathcal{M})$ (respectively $\mathcal{M}$) in the lattice $\mathbb{M}_n$.

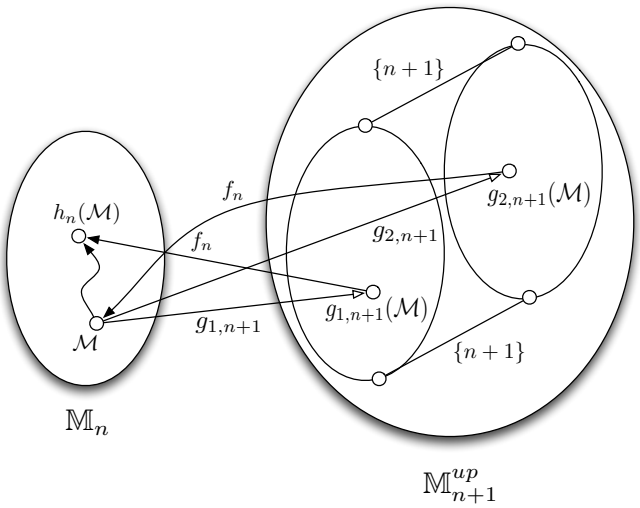

Fig. 4 On the left, the lattice $\mathbb{M}_n$ of the Moore co-families on U_n. On the right, both isomorphic images of this lattice obtained by functions $g_{1,n+1}$ and $g_{2,n+1}$. The gathering of these two images corresponds to the set of Moore co-families $\mathbb{M}_{n+1}^{up}$.

Theorem 1 states that the set of Moore co-families on U_{n+1} can be obtained from the set of Moore co-families on U_n. In the recursive decomposition formula given below for $\mathbb{M}_{n+1}$, we use the functions $g_{1,n+1}$, $g_{2,n+1}$ and h_n defined above.

Theorem 1 (Colomb et *al.*[9]) *For any integer n such that $n \geq 1$,*

$$\begin{array}{rl} \mathbb{M}_{n+1} = & \bigcup_{\mathcal{M} \in \mathbb{M}_n} \{g_{1,n+1}(\mathcal{M}) \cup \mathcal{M}' \mid \mathcal{M}' \in \mathbb{I}_{\mathbb{M}_n}(h_n(\mathcal{M}))\} \quad \cup \\ & \bigcup_{\mathcal{M} \in \mathbb{M}_n} \{g_{2,n+1}(\mathcal{M}) \cup \mathcal{M}'' \mid \mathcal{M}'' \in \mathbb{I}_{\mathbb{M}_n}(\mathcal{M})\} \end{array}$$

In other words, there exists a natural bi-partition of $\mathbb{M}_{n+1}$: the families not containing the singleton $\{n+1\}$, under the form $g_{1,n+1}(\mathcal{M}) \cup \mathcal{M}'$ with $\mathcal{M}'$ as a sub Moore co-family of $h_n(\mathcal{M})$ and the families containing $\{n+1\}$, under the form $g_2(\mathcal{M}) \cup \mathcal{M}''$ with $\mathcal{M}''$ as a sub Moore co-family of $\mathcal{M}$.

3 Recursive decomposition tree of a Moore co-family

3.1 Definition

Let $\mathcal{M}$ be a Moore co-family in $\mathbb{M}_n$ and x be in U_n, we denote $\mathcal{M}_x$ (resp. $\mathcal{M}_{\bar{x}}$) the restriction of $\mathcal{M}$ to its sets containing x (resp. to its sets not containing x). The empty set is added to $\mathcal{M}_x$.

3.2 Theoretical results

Another interpretation of the inductive definition of $\mathbb{M}_n$ stated by Theorem 1 is that any Moore co-family can be decomposed by a recursive process described by Corollary 2.

Corollary 2 *Let $\mathcal{M}$ be a Moore co-family in $\mathbb{M}_n$, for any element x in U_n, $\mathcal{M}$ can be written $\mathcal{M}_{\bar{x}} \cup g_{i,x}(\mathcal{M}')$ with $\mathcal{M}'$ such that $g_{i,x}(\mathcal{M}')$ corresponds to $\mathcal{M}_x$. This way, $\mathcal{M}$ is decomposed in two Moore co-families $\mathcal{M}_{\bar{x}}$ and $\mathcal{M}'$. Applying recursively this decomposition principle we obtain a decomposition binary tree of $\mathcal{M}$ (each leaf is an empty set which cannot be further decomposed). The left branch of a node $\mathcal{M}$, labelled $\bar{x}$, leads to $\mathcal{M}_{\bar{x}}$ and the right one, labelled $g_{i,x}$, leads to $\mathcal{M}'$ such that $g_{i,x}(\mathcal{M}')$ is equal to $\mathcal{M}_x$ (if the singleton $\{x\} \in \mathcal{M}$ we have $i = 2$, if not $i = 1$).*

An example of decomposition of a Moore co-family is given in Figure 5.

Basically each leaf of the tree shall relate to one set, and only one, of the initial family $\mathcal{M}$. To obtain the set corresponding to a leaf, one just has to iteratively apply the different functions $g_{1,x}$ or $g_{2,x}$ that can be found in the path from the chosen leaf to the root of the tree.

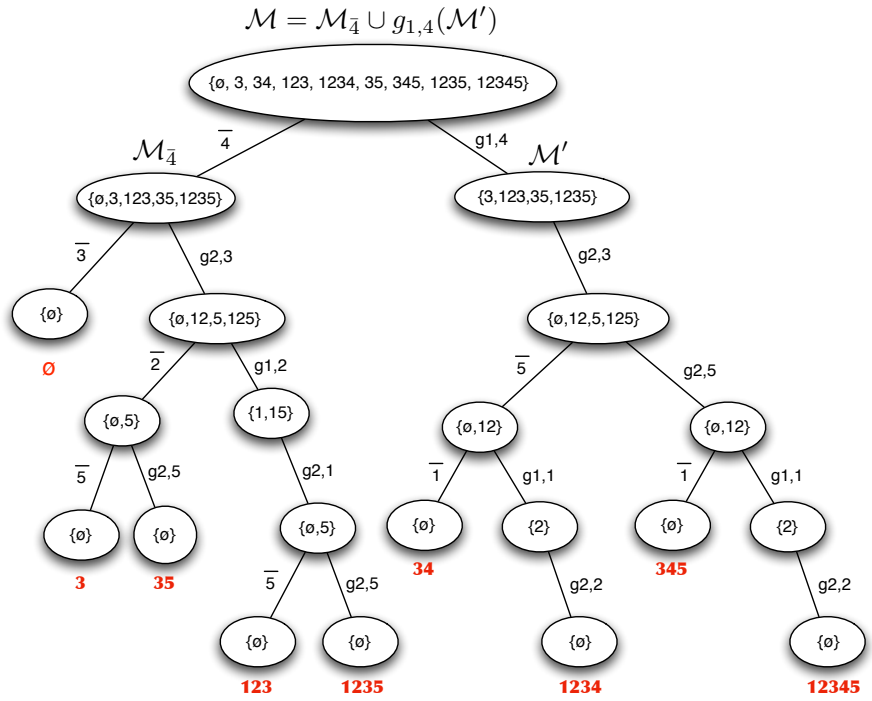

Fig. 5 Decomposition of a Moore co-family. The family $\mathcal{M}'$ only owns a right branch.

3.3 Proper decomposition tree

As we can see on the example given in Figure 5, some nodes only own a right branch. This may occur in two well-defined cases: either when the chosen element x belongs to each sets of the family, or when the family is reduced to only one set to be further decomposed.

However, it is always possible to obtain a *proper binary tree* (i.e. a tree with the property that any internal node owns exactly two sons, also called *full binary tree*). Indeed, in the first case, for any given family with at least two sets, there exists always an element x which belongs to a set and does not belong to another one. By choosing this element wisely we are ensured that the node will have a left branch and a right one. The second case can be detected and the recursive process easily adapted. In Figure 6, we give a proper decomposition tree for another family.

Remark: It is well known that in a proper binary tree the number of leaves is equal to the number of internal nodes plus one. This point will be further addressed to state the time complexity of the closure algorithm.

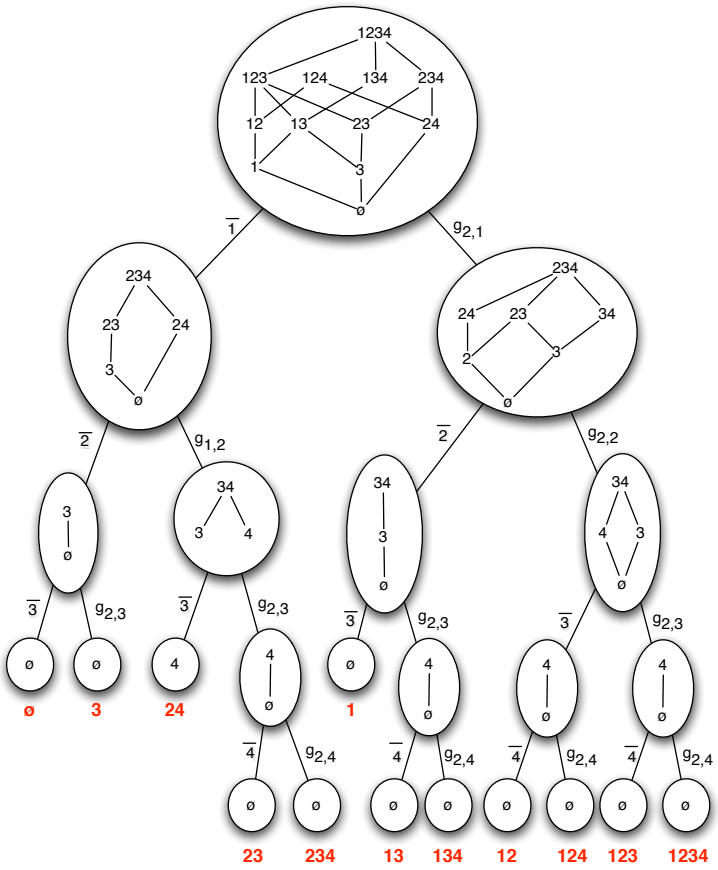

Fig. 6 Proper decomposition tree of a Moore co-family. Each internal node of the tree owns two sons.

4 Algorithm to compute the union closure of a family

Problem of generation of closed-sets from a given family has been well-studied these last years. Most of existing algorithms are based on a decomposition strategy [18,5,21,19]. As previously stated, generation of closed-sets and co-closed sets can be treated in the same way (i.e. one only have to complement every sets of input and output families to obtain both closed and co-closed families). In this section, we present an algorithm to generate a Moore co-family $\mathcal{M}$ from its set of join-irreducible elements denoted $\mathcal{J}_{\mathcal{M}}$. This way, this algorithm can be used to compute the closure by union of any given family. This algorithm is based on Theorem 1.

4.1 Definition

A set $J \in \mathcal{M}$ is called a join-irreducible of $\mathcal{M}$ if J covers only one set in $\mathcal{M}$. The set of all join-irreducible sets of $\mathcal{M}$ is denoted $\mathcal{J}_\mathcal{M}$. By extension, we denote by $(\mathcal{J}_\mathcal{M})_x$ (resp. $(\mathcal{J}_\mathcal{M})_{\bar{x}}$) the set of join-irreducible elements of $\mathcal{M}$ which contains the element x (resp. which do not contain the element x). We call union product between two families $\mathcal{M}_1$ and $\mathcal{M}_2$, denoted $\mathcal{M}_1 \otimes \mathcal{M}_2$, the set $\{M_1 \cup M_2 \mid M_1 \in \mathcal{M}_1, M_2 \in \mathcal{M}_2\}$.

4.2 Theoretical results

We propose an algorithm based on the decomposition of a Moore co-family that only uses families of join-irreducible sets. In other words, for each step of the recursive decomposition, given a local set $\mathcal{J}_\mathcal{M}$, we compute families of sets $\mathcal{J}_{\mathcal{M}_{\bar{x}}}$ and $\mathcal{J}_{\mathcal{M}'}$ corresponding to the join-irreducible sets of $\mathcal{M}_{\bar{x}}$ and $\mathcal{M}'$ such that $\mathcal{M} = \mathcal{M}_{\bar{x}} \cup g_i(\mathcal{M}')$ (see Figure 7). Leaves of the decomposition tree obtained after the whole recursive decomposition correspond to the closure of the family $\mathcal{M}$ (see Figure 8).

Proposition 2 *Let $\mathcal{M}$, $\mathcal{M}' \in \mathbb{M}_n$, let $x \in U_n$ and let $i \in \{1, 2\}$ such that $\mathcal{M} = \mathcal{M}_{\bar{x}} \cup g_{i,x}(\mathcal{M}')$. Then*

- $\mathcal{J}_{\mathcal{M}_{\bar{x}}} = (\mathcal{J}_\mathcal{M})_{\bar{x}}$;
- $\mathcal{J}_{\mathcal{M}'} \subseteq \{J \backslash \{x\} \mid J \in (\mathcal{J}_\mathcal{M})_x\} \cup ((\mathcal{J}_\mathcal{M})_{\bar{x}} \otimes \{J \backslash \{x\} \mid J \in (\mathcal{J}_\mathcal{M})_x\})$.

An illustration is given in Figure 7.

Proof

- $\mathcal{J}_{\mathcal{M}_{\bar{x}}} = (\mathcal{J}_\mathcal{M})_{\bar{x}}$
  Any join-irreducible element of $\mathcal{M}$ not containing x is also join-irreducible of $\mathcal{M}_{\bar{x}}$. Indeed, all the predecessors of each set in $\mathcal{M}$ not containing x, don't contain x itself. Similarly, there is no new join-irreducible element in $\mathcal{J}_{\mathcal{M}_{\bar{x}}}$.
- $\mathcal{J}_{\mathcal{M}'} \subseteq \{J \backslash \{x\} \mid J \in (\mathcal{J}_\mathcal{M})_x\} \cup ((\mathcal{J}_\mathcal{M})_{\bar{x}} \otimes \{J \backslash \{x\} \mid J \in (\mathcal{J}_\mathcal{M})_x\})$
  Let us show that $g_{i,x}(\mathcal{J}_{\mathcal{M}'}) \subseteq (\mathcal{J}_\mathcal{M})_x \cup ((\mathcal{J}_\mathcal{M})_x \otimes (\mathcal{J}_\mathcal{M})_{\bar{x}})$. Let $J \in g_{i,x}(\mathcal{J}_{\mathcal{M}'})$, then $J \in \mathcal{J}_{\mathcal{M}_x}$. Two cases occur:
  - $J \in \mathcal{J}_\mathcal{M}$ and with $x \in J$ we obtain $J \in (\mathcal{J}_\mathcal{M})_x$;
  - J owns one, and only one, immediate predecessor in $\mathcal{M}_x$ and another one, and only one, in $\mathcal{M}_{\bar{x}}$. By contradiction, let us suppose by hypothesis that $J \notin (\mathcal{J}_\mathcal{M})_x \otimes (\mathcal{J}_\mathcal{M})_{\bar{x}}$. Let $S = \{s_1, s_2, ..., s_n\}$ (resp. $S' = \{s'_1, s'_2, ..., s'_m\}$) be the family of maximal join-irreducible sets of $\mathcal{M}_{\bar{x}}$ (resp. of $\mathcal{M}_x$) such that $\forall i,\ s_i \subset J$ (resp. $\forall j,\ s'_j \subset J$). Let $Max = max(S \otimes S')$, we have $Max \subseteq \mathcal{M}_x$ and by hypothesis we have $\forall m \in Max, m \neq J$ and so $|Max| > 1$. With $J = (\bigcup_{m \in Max} m)$ we conclude that $J \notin \mathcal{J}_{\mathcal{M}_x}$. $J \setminus \{x\} \notin \mathcal{J}_{\mathcal{M}'}$ and then $J \notin g_{i,x}(\mathcal{J}_{\mathcal{M}'})$. Contradiction.

□

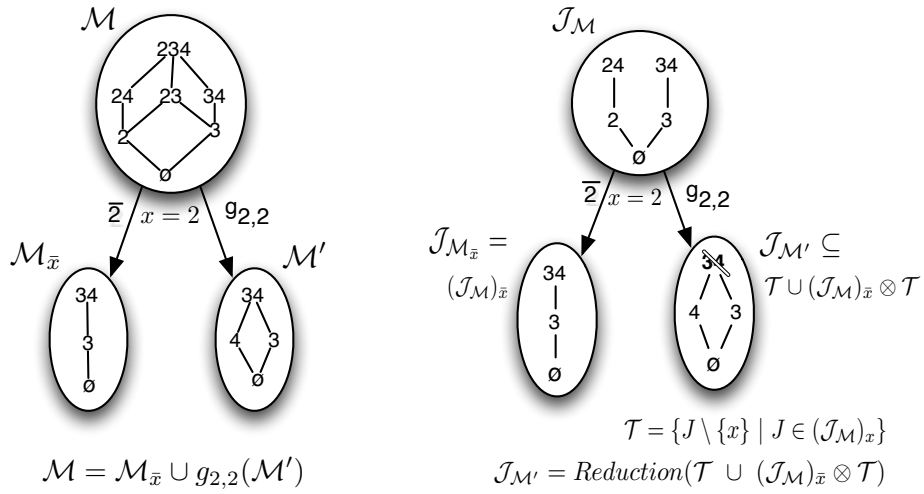

Fig. 7 On the left, $\mathcal{M}$ is splitted in two parts according to theorem 1 with $x = 2$. Singleton $\{2\}$ belongs to $\mathcal{M}$ and so $i = 2$. We have $\mathcal{M} = \mathcal{M}_{\bar{x}} \cup g_{2,2}(\mathcal{M}')$. To compute the closure of a given family, we will simulate this step by the process illustrated on the right of the figure. From the set of join irreducible elements of $\mathcal{M}$ denoted $\mathcal{J}_{\mathcal{M}}$, thanks to Proposition 2 we compute the set $\mathcal{J}_{\mathcal{M}_{\bar{x}}}$ of join-irreducible elements of $\mathcal{M}_{\bar{x}}$ (on the left branch), and the set $\mathcal{J}_{\mathcal{M}'}$ of join-irreducible elements of $\mathcal{M}'$ (on the right branch). The set $\{3, 4\}$, in bold, is not a join-irreducible element and it will be deleted by the reduction step.

Proposition 2 gives an exact characterization of the set $\mathcal{J}_{\mathcal{M}_{\bar{x}}}$ and merely states a superset of $\mathcal{J}_{\mathcal{M}'}$. But sets $\mathcal{J}_{\mathcal{M}'}$ and $\{J \backslash \{x\} \mid J \in (\mathcal{J}_{\mathcal{M}})_x\} \cup (\mathcal{J}_{\mathcal{M}})_{\bar{x}} \otimes \{J \backslash \{x\} \mid J \in (\mathcal{J}_{\mathcal{M}})_x\}$ share exactly the same set of join-irreducible elements. Thus, it will be enough to proceed to a step of reduction to obtain the exact value of $\mathcal{J}_{\mathcal{M}'}$. We call **process of reduction** of a family, the process consisting of deleting all non join-irreducible sets of the family.

4.3 Algorithm

The given Algorithm 1 is recursive and the stop case arises when the local family $\mathcal{J}_{\mathcal{F}}$ is reduced to one set.

In the general case, an element x is wisely chosen (see Section 4.5) and the algorithm computes the set of join-irreducible elements of $\mathcal{F}_{\bar{x}}$ and of $\mathcal{F}'$ such that $\mathcal{F} = \mathcal{F}_{\bar{x}} \cup g_{i,x}(\mathcal{F}')$. Since $\mathcal{J}_{\mathcal{F}_{\bar{x}}} = (\mathcal{J}_{\mathcal{F}})_{\bar{x}}$ the recursive process is

launched on the left branch with $(\mathcal{J}_\mathcal{F})_{\bar{x}}$ as input family. After that, the set $\mathcal{J}_{\mathcal{F}'}$ is computed thanks to a reduction process from the set $\{J \backslash \{x\} \mid J \in (\mathcal{J}_\mathcal{F})_x\} \cup (\mathcal{J}_\mathcal{F})_{\bar{x}} \otimes \{J \backslash \{x\} \mid J \in (\mathcal{J}_\mathcal{F})_x\}$. The recursive process is then launched again, corresponding to the right branch, and the element x is added to the current solution. See Figure 8 for an example.

Note: It is important to understand that the algorithm operates well even if the given family is not reduced (the call to the reduction process in Line 4, Algorithm 1, is optional). Although the process of reduction is the opposite process to the closure process, our experimental results have shown that it was much more efficient to reduce the given family before proceeding with the closure process.

Algorithm 1: Co-closure($\mathcal{J}_\mathcal{F}$, S)

Input: $\mathcal{J}_\mathcal{F}$ a family of sets;
S the initial part of the current solution;

begin
  if $\mathcal{J}_\mathcal{F} = \{E\}$ **then**
    print$(E \cup S)$;
  else
    1 Choose an element x in $\bigcup \mathcal{J}_\mathcal{F} \setminus \bigcap \mathcal{J}_\mathcal{F}$;
    2 $\mathcal{J}_{\mathcal{F}_{\bar{x}}} \leftarrow \{E \mid x \notin E, E \in \mathcal{J}_\mathcal{F}\}$;
    3 $\mathcal{T} \leftarrow \{E \setminus \{x\} \mid x \in E, E \in \mathcal{J}_\mathcal{F}\}$;
    Co-Closure$(\mathcal{J}_{\mathcal{F}_{\bar{x}}}, S)$;
    4 $\mathcal{J}_{\mathcal{F}'} \leftarrow \mathit{Reduction}(\mathcal{T} \cup \mathcal{J}_{\mathcal{F}_{\bar{x}}} \otimes \mathcal{T})$;
    Co-Closure$(\mathcal{J}_{\mathcal{F}'}, S \cup \{x\})$;

4.4 Discussion about the time complexity of Algorithm 1

We denote $|Co - closure(\mathcal{F})|$ the size of the closure under union of the family $\mathcal{F}$. Since the decomposition tree obtained at the end of the whole process of Algorithm 1 is a proper binary tree, we know that this tree owns exactly $|Co - closure(\mathcal{F})| - 1$ internal nodes[3]. This means that the time complexity needed to compute **one** set of the closure corresponds exactly to the time complexity needed to treat **one** internal node. Moreover in the following we will suppose that the size of the ground set U_n is constant. This way, the operations on sets like intersection and union will be considered feasible in constant time.

Let $\mathcal{F}$ be the local family of an internal node. Lines 1, 2 and 3 of Algorithm 1 can be treated in one pass through the family $\mathcal{F}$. All these operations can

[3] For any proper binary tree, the number of leaves is equal to the number of internal nodes plus one.

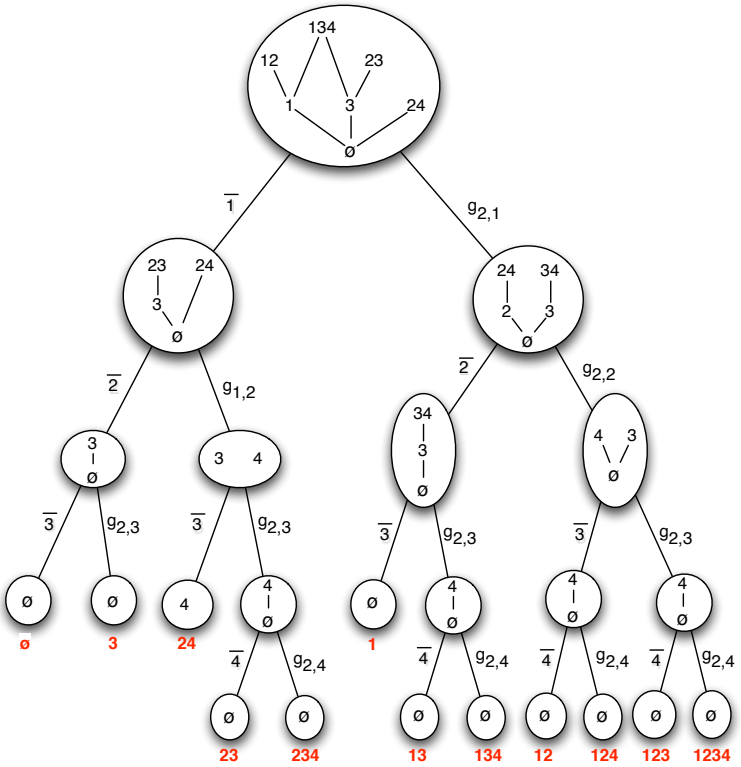

Fig. 8 Proper decomposition tree obtained by the execution of Algorithm 1 on the set of join irreducible elements of the family given in Figure 6. One can check that leaves of the trees are strictly identical.

be done in $\mathcal{O}(|\mathcal{F}|)$. Basically, the whole complexity of the computing lies in line 4 with two processes: a process of union product between two sets and then a process of reduction of the obtained family. These two processes are quadratic in the size of the involved families. The most noteworthy fact is that sizes of this families are not always polynomially correlated with the size of the initial family (we mean the family of the root of the tree). But one has to know that the size of the family resulting of the product is never greater than to the number of leaves of the sub-tree rooted in this internal node. This means that the local time complexity is polynomially linked with the size of the sub-result rooted in this node.

4.5 Discussion about the choice of the element x

The choice of the element x leads to a bi-partition of the input family $\mathcal{J}_{\mathcal{F}}$ into $\mathcal{J}_{\mathcal{F}_{\bar{x}}}$ and $\mathcal{T}$ (see Lines 2 and 3, Algorithm 1). Thus the sum $|\mathcal{J}_{\mathcal{F}_{\bar{x}}}| + |\mathcal{T}|$ is constant. As already mentioned, complexity of Algorithm 1 lies in line 4 with

a process of union product between $\mathcal{J}_{\mathcal{F}_{\bar{x}}}$ and $\mathcal{T}$ and a process of reduction of the result. These two operations have a time complexity in $\mathcal{O}(|\mathcal{J}_{\mathcal{F}_{\bar{x}}}| * |\mathcal{T}|)$.

- Regarding the union product, the biggest complexity comes when sets have approximately the same size. By this way, we must choose x to ensure that $|\mathcal{J}_{\mathcal{F}_{\bar{x}}}|$ or $|\mathcal{T}|$ is minimum.
- Our experimental studies show that the majority of time is spent on the process of reduction. Along the left branch of the recursive process, the time consuming process steps such as the reduction is the sum of the times spent on the reduction of each set $\mathcal{T}$. The complexity of this reduction being quadratic, to achieve short response times, the smallest possible sub-families which do not contain x should always be used.

For these two reasons, the best choice for x is the element which occurs the least often. But it is, in any case, essential to state here that, in spite of all these general rules, the computation time to find x will increase significantly calculation times of the whole process.

5 Experimental results

5.1 Experimental design

In order to assess performance, the approach described previously has been implemented in C. The executable file was generated with compiler GCC 4.4. The experiments were performed on a AMD Opteron 8356 processor with 2.3 GHz of speed and 256 Go of memory. Several well-known benchmark real-world data-sets were chosen from UCI[4]. The following table summarizes the data-sets characteristics.

Table 1 Data-sets characteristics of several well-known benchmark data-sets chosen from UCI.

Data-sets	# Attributes	# Sets	# Attributes per set	# Closed set
Mushroom	119	8124	23	238 709
Chess	75	3196	37	930 851 336
Connect	129	67557	43	1 415 737 994

To compare performance of Co-closure and previous approaches with respect to the size of family, we implemented a version of Norris' algorithm (see [22]) as well as a version of Ganter's algorithm called Next-closure (see [18]). It can be noticed that we have used the natural one to one mapping between Moore and co-Moore families to compute closed sets using Co-Closure algorithm.

[4] Uci machine learning repository

5.2 Result

We have compared execution time of these different algorithms on parts of benchmark data-sets. In that way, we have run algorithms on the first m sets of each benchmark data-sets by varying m. On Tables 2, 3, and 4 we give numbers of co-closed sets we obtained by execution of the three algorithms for each size of the considered data-sets partition. In the last three rows we give the execution time for each algorithm. ms stands for milliseconds, m for minutes, h for hours and d for days.

Table 2 Chess

#sets	# closed	Ganter	Norris	Co-Closure
100	10 867	60ms	60ms	140ms
200	73 033	160ms	190ms	600ms
500	1 051 399	1.6s	6s	5s
800	10 145 237	21s	1m19	33s
1000	38 558 373	1m50s	5m20	3m
1500	183 655 113	12m	50m	51m
2000	292 107 880	23m	2h02	4h15
2500	386 235 477	40m	>5h	>5h
3196	930 851 336	2h15	>5h	>5h

Table 3 Mushroom

#sets	# closed	Ganter	Norris	Co-Closure
50	1208	75ms	75ms	90ms
100	3459	80ms	85ms	150ms
200	7086	80ms	95ms	310ms
500	17 781	120ms	180ms	570ms
1000	32 513	205ms	380ms	900ms
1500	48 414	340ms	680ms	1.5s
2000	58 982	430ms	1s	1.7s
2500	72 008	490ms	2s	2.5s
3000	80901	700ms	2.1s	3s
4000	104 104	1s	3.5s	4.5s
5000	136 401	1.5s	5s	7s
6000	156 573	2s	7.5s	8.5s
7000	214 950	3.5s	11.7s	13s
8000	237 874	4s	17s	15s
8124	238 709	4.1s	17.4s	15.5s

Table 4 Connect

#sets	# closed	Ganter	Norris	Co-Closure
100	13 406	730ms	760ms	850ms
200	63 360	840ms	900ms	1.3s
300	149 393	875ms	1.2s	2s
500	445 676	1.3s	3.1s	7.3s
800	753 371	2s	8s	14s
1000	1 069 569	3s	17s	24s
2000	4 732 622	20s	2m	3m40s
3000	9 742 932	1m	5m35s	20m
5000	22 543 073	3m25s	26m	3h15m
10000	69 916 189	20m	>4h	>5h
20000	242 644 240	2h30	>4h	>5h
67557	1 415 737 994	4d	>4h	>5h

5.3 Analysis of the results

The first point to note here is that Ganter's algorithm shows the best performance in each context. Basically, the performance of the proposed algorithm appears to be equivalent to the performance of Norris' algorithm. We should add that in some cases this algorithm exceeds Norris'. Moreover, let us recall that Norris' algorithm consumes a lot of memory. Indeed the family resulting of the computing is kept in memory during all the process. Note that it is not the case of the algorithmic process described in this paper.

6 Conclusion

In [8] authors have computed the exact size of $|\mathbb{M}_7|$. From this first technical challenge has derived a recursive decomposition theorem which has been formalized in [9]. In the present article we have shown that each Moore co-family can be represented by a proper decomposition tree and we use this principle to design an original algorithm to close by union any family. The process has been implemented and the results on well-known benchmarks are given in the last section. Experimentations state the correctness of the process but show that time complexity is not satisfying. However, we think this straightforward process is interesting by itself. Moreover we have determined two major ways of improvement :

- Since most of the time is spent on the reduction of the set of join-irreducible elements on the right branch, a question could be to early detect sets, produced by a union product, which will not be join-irreducible elements and so will be deleted by the reduction process.
- A second interesting question will be the design of some kind of dynamic programming algorithm to take into account the redundancy of families which appear in the bottom part of the tree. Maybe we should use another closure algorithm in this part of the process.

References

1. Barbut, M., Monjardet, B., Ordre et classification. Hachette (1970)
2. Beaudou, L., Colomb, P., Raynaud, O., Structural properties and algorithms on the lattice of Moore co-families. In proceedings of ICFCA (Supplementary actes), p.41–52 (2012)
3. Birkhoff, G., Rings of sets. Duke Mathematical Journal **3**, p.443–454 (1937)
4. Birkhoff, G., Lattice Theory. Third edn. American Mathematical Society (1967)
5. Bordat, J.P., Calcul pratique du treillis de galois d'une correspondance. Math. Sci. Hum. **96**, p.31–47 (1986)
6. Burosh, G., Demetrovics, J., Katona, G., Kleitman, D., Sapozhenko, A., On the number of databases and closure operations. Theoretical computer science **78**, p.377–381 (1991)
7. Caspard, N., Monjardet, B., The lattices of closure systems, closure operators, and implicational systems on a finite set: a survey. Discrete Applied Mathematics **127**, p.241–269 (2003)
8. Colomb, P., Irlande, A., Raynaud, O., Counting of Moore families on n=7. In proceedings of ICFCA, LNAI 5986, p.72–87 (2010)
9. Colomb P., Irlande A., Raynaud O., Renaud R., Recursive Decomposition and bounds of the Lattice of Moore Co-Families. Ann Math Artif Intell, AMAI Springer, vol.67, issue 2, p. 109-122 (2013).
10. Cohn, P., Universal Algebra. Harper and Row, New York (1965)
11. Doignon, J.P., Falmagne, J.C., Knowledge Spaces. Springer, Berlin (1999)
12. Demetrovics, J., Libkin, L., Muchnik, I., Functional dependencies in relational databases: A lattice point of view. Discrete Applied Mathematics **40**(2), p.155–185 (1992)
13. Demetrovics, J., Libkin, L., Muchnik, I., Functional dependencies and the semilattice of closed classes. In: MFDBS, LNCS 364, p.136–147 (1989)
14. Davey, B.A., Priestley, H.A., Introduction to lattices and orders. Cambridge University Press (1991)
15. Demetrovics, J., Molnar, A., Thalheim, B., Reasoning methods for designing and surveying relationships described by sets of functional constraints. Serdica J. Computing **3**, p.179–204 (2009)
16. Duquenne, V., Latticial structure in data analysis. Theoretical Computer Science (217), p.407–436 (1999)
17. Ganter, B., Wille, R., Formal concept analysis, mathematical foundation. Berlin-Heidelberg-NewYork et al., Springer (1999)
18. Ganter, B., Two basic algorithms in concept analysis., Preprint 831, Technische Hochschule Darmstadt (1984)
19. Gely, A., A generic algorithm for generating closed sets of a binary relation. In proceedings of ICFCA. p.223-234 (2005)
20. Habib, M., Nourine, L., The number of Moore families on n=6. Discrete Mathematics **294**, p.291–296 (2005)
21. Lindig, C., Fast concept analysis. In proceedings of ICCS Shaker Verlag, p. 152–161 (2000)
22. Norris, E.M., An algorithm for computing the maximal rectangles in a binary relation. Revue Roumaine de Mathématiques Pures et Appliquées, **23**(2), p.243–250 (1978)
23. Sierksma, G, Convexity on union of sets. Compositio Mathematica, **42**, p.391-400 (1980),
24. van de Ven, L. M. J., Theory of convex structures. North-Holland, Amsterdam (1993)