A distributed version of the Ganter algorithm for general Galois Lattices

Gérard Lévy and Fatma Baklouti

CERIA Laboratory, Paris Dauphine University
Place du Maréchal de Lattre de Tassigny, 75775 Paris cedex 16, France
{gerard.levy,fatma.baklouti}@dauphine.fr

Abstract. Standard Galois Lattices are effective tools for data analysis and knowledge discovery. Several algorithms were proposed to generate concepts of lattices, among which the ScalingNextClosure algorithm. In order to share the production workload between several processors when the number of closed itemsets to determine is very large, this algorithm leans on the sequential character of the closed itemsets determination of a Galois Lattice by the Ganter algorithm. In this paper, we prove that the parallelised version of the ScalingNextClosure can be extended to more general contexts (even some complex data) than usual binary contexts and that the partition of the workload between processors can be made with all the wished precision.

1 Introduction

Standard Galois Lattices are very useful tools in data mining. They allow us to structure data sets, by extracting concepts and rules to deduce concepts from other concepts. Concept lattice studies how objects can be hierarchically grouped together according to their common attributes. Since each set of objects possesses some attributes, we can classify objects and attributes according to the relation between objects set and attributes set. Formal concept or closed itemset represents stronger association between itemsets and the set of their common objects.
Several algorithms are well known and used to determine the Galois lattice GL(C) associated to a given context C, when the size of C is not too large. Nevertheless, we need nowadays to treat contexts which are large and not necessarily binary.
Several algorithms were proposed to generate concepts of lattices, among which the ScalingNextClosure algorithm proposed in [11]. In order to share the workload between several processors when the number of closed itemsets to determine is too large, this algorithm leans on the sequential character of the closed itemsets determination of a Galois Lattice by the Ganter algorithm. One of the essential points of this distributed version is the work distribution between various processors. (The qualifier "large” concerns here not only the size of the context but also the number of closed itemsets of the lattice since a small table can generate a big lattice).

Radim Bělohlávek, Václav Snášel (Eds.): CLA 2005, pp. 207–221, ISBN 80–248–0863–3.

In this paper, we show that the parallelized version of the ScalingNextClosure can be extended to more general contexts, than usual binary contexts and that the partition of the work between processors can be made with all the required precision.

The rest of the paper is organized as follows. In section 2, we define all the concepts necessary to present clearly the general contexts to which we shall apply our algorithms, and to develop the tools of partition. In section 3, we define the lexicographical order. We introduce the first version of the Ganter algorithm in section 4 and its segmented version in section 5. Section 6 is devoted to the choice of partitions. In section 7, we define some mathematical tools in order to help the construction of good partitions proposed in section 8. Section 9 concludes this paper by listing future research directions.

2 Notations, Definitions

Let m and n be two finite positive integers. The set $I = \{1...m\} = [1...m]$ designates a set of m individuals or objects i, and the set $J = \{1... n\} = [1... n]$ designates a set of n properties or variables j.

Every variable j takes its values in a set $B_j = \{0...b_j - 1\} = [0...b_j - 1]$, where b_j is a finite integer > 1.

In that case, the sequence $(b_1...b_n)$ will be called a *multibased* or a *generalized base*, or *a heterogeneous base*.

We suppose that every B_j is provided with the total order $\leq$ of the natural integers.

We note $B^{[n]}$ the *Cartesian product* B_1 x B_2 x… x B_n which is the set of the *elements* $X_n = (x_1... x_n)$ such as $x_j \in B_j$, for every j of J.

We provide $B^{[n]}$ with the relation of order $\leq$ of the Cartesian product, namely: $X_n \leq Y_n$ iff $x_j \leq y_j$ for every j of J. Provided with this relation $< B^{[n]}, \leq >$ is a lattice for the operations "sup" (supremum) noted $\vee$ and "inf" (infimum) noted $\wedge$. These operations are defined by:

$Z_n = X_n \vee Y_n$ iff for every j of J $z_j = x_j \vee y_j = sup\ (x_j, y_j);$

$Z_n = X_n \wedge Y_n$ iff for every j of J $z_j = x_j \wedge y_j = inf\ (x_j, y_j).$

Furthermore, the extreme elements of this lattice are O_F and 1_F. O_F and 1_F are defined by $O_F = (0… 0… 0)$, and $1_F = (b_1 - 1... b_j - 1... b_n - 1)$.

Every X_n *of* $B^{[n]}$ *verifies the constraint* $0_F \leq X_n \leq 1_F$.

(In the rest of this paper, in order to avoid using $b_j - 1$, we shall put $1_F = (c_1... c_j... c_n)$, with $c_j = b_j - 1$, for every j *of* J)

Let $E = 2^I = P(I)$ be the set of subsets of I.

Let d: $I \rightarrow B^{[n]}$ be a mapping which associates to every individual i of I its *description* $d\ (i) = (d_1\ (i)...\ d_j\ (i)...\ d_n\ (i))$, where $d_j\ (i)$ *is the* value of the variable j for the individual i.

Let f: $E \rightarrow B^{[n]}$ be the mapping which associates to every subset X of I the element $f\ (X)$ *of* $B^{[n]}$ defined by $f\ (X) = 1_F$; $X = \emptyset$, *and* $f\ (X) = \wedge_{i \in X}\ d\ (i) = d\ (i_1) \wedge d\ (i_2) \wedge. ... \wedge d\ (i_k)$ if $X = \{i_1, i_2… i_k\} \subseteq I$. ($f\ (X)$ *is called* the *intent* of X).

We define the *mapping* g: $B^{[n]} \rightarrow E$ by $g\ (Z_n) = \{i \in I: Z_n \leq d\ (i)\}$, for every Z_n *de* $B^{[n]}$. ($g\ (Z_n)$ is called the *extent* of Z_n).

We note that f and g are decreasing mappings and their composites $h = g \circ f$ and $k = f \circ g$ are closure operators.
In the literature of « data mining », the triplet $C = < I, F, d >$ is called a *context*. It is often materialized by a table, which has m lines and n columns, every element in position (i, j) being equal to $d_j(i)$.
The pair (f, g) is called the *Galois connection* associated to the context C.
The elements X of E such as $h(X) = X$ are called the closed elements of E, and Z_n *of* $B^{[n]}$ such as $k(Z_n) = Z_n$ are called the closed elements of $B^{[n]}$.
Let us note $Inv(E) = \{X \in E: h(X) = X\}$ and $Inv(F) = \{Z_n \in F: k(Z_n) = Z_n\}$.
We can prove that these sets are in bijection [4].
The set of pairs (X, Z_n) of $E \times B^{[n]}$ such as ($X \in Inv(E)$ and $Z_n = f(X)$) is equal to the set of all the pairs (X, Z_n) such as ($Z_n \in Inv(F)$ and $X = g(Z_n)$). (Each of these pairs is called a *concept*). This set is called the *Galois Lattice* associated to the context C, and noted $TG(C)$.
$TG(C) = \{(X, Z_n): X \in Inv(E) \text{ and } Z_n = f(X)\} = \{(X, Z_n): Z_n \in Inv(F) \text{ and } X = g(Z_n)\}$.
We can prove that this set is a lattice for the relation of *order* $\leq$ defined by $(X, Z_n) \leq (X', Z_n')$ iff $X \subseteq X'$ *and* $Z_n' \leq Z_n$.
Many problems of data mining are related to the determination of the Galois Lattice associated to a context C. Depending on the nature of the data mining problem, this determination may consist in determining simply all the concepts (X, Z_n), or to determine this set and its order relation.

3 Lexicographical Order on $B^{[n]}$

Up to here, $B^{[n]}$ is provided with the order associated with $B_1 \times B_2 \times \ldots \times B_n$.
If all the b_j's are equal to 2 (binary case), the Ganter algorithm allows to build all the closed itemsets of the Galois lattice following the lexicographical order of its elements.
In order to generalize the Ganter algorithm, we define, in this section, this order on $B^{[n]}$ when the b_j's are greater than 1.

3.1 The lexicographical order definition

Being given two elements X_n and Y_n of $B^{[n]}$, our objective is to determine their greatest common prefix. Two cases are possible: either $X_n = Y_n$, or they are different. If $j = \inf\{k / x_l \neq y_l\}$ of and if $x_j < y_j$ then X_n precedes Y_n in lexicographical order (L.O), and otherwise Y_n precedes X_n. We note $X_n \angle Y_n$ the fact that X_n precedes Y_n in L.O.
In pseudo-Pascal, the following function takes the value True iff $X_n \angle Y_n$.

```
Function inflex (Xn, Yn: elements of B[n]): boolean;
  Var j: integer;
  Begin
```

```
        j = 1; while ((j < = n) and (x [j] = y [j])) do  j = j+1;
        Inflex= (j > n) or (x [j] < y[j]);
  End;
```

3.2 The next in lexicographical order of X_n

While $1_F = (c_1\dots\ c_j\dots\ c_n)$, we define the transition subscript of every X_n of $B^{[n]}$ as follows:
$j = n$; While (($j > 0$) and ($x\ [j] = c\ [j]$)) do $j = j - 1$; Subscript of transition= j;
In other words, if $X_n = (x_1, x_2\dots\ x_{i-1}, x_i, c_{i+1}, c_{i+2}\dots c_n)$, and $x_i < c_i$, then its transition subscript is i.
This subscript is designated by $i^+(X_n)$. Therefore, $i^+(X_n) = 0$, iff $X_n = 1_F$.
Moreover, if $X_n \neq 1_F$, and if its transition subscript is i, we define its following Y_n in lexicographical order by $Y_n=(x_1, x_2\dots\ x_{i-1}, 1 + x_i, 0, 0\dots\ 0)$., and we note it X_n^+. (It is easy to verify that Y_n follows of X_n in L.O).On the other hand, if $X_n = 1_F$, the following of X_n in L.O is not defined.
We can also define, the next of X_n in L.O, from the subscript i (of J), as follows:

```
Procedure next (X: element of B [n]; i: element of J; the Var Y: element of B [n];
Var i+: integer)

Var j, k: integer;
begin
j = i;
        While ((j > 0) and (x [j] = c [j])) do j = j-1;
        i+ = j;
        If (i+ > 0) then
        begin
                For j = 1 to i+ - 1 do y [j] = x [j];
                y [i+] = 1 + x [i+];
                For j = 1 + i+ to n do y [j] = 0;
        end;
end;
```

Remark: when i is the transition subscript of X, this procedure output is $Y = X^+$, the next of X in L.O, which may be generally provided by starting with $i = n$.
For every $X = X_n = (x_1, x_2\dots\ x_{i-1}, x_i, c_{i+1}, c_{i+2}\dots\ c_n)$, other than 1_F, we define the element X^* of $B^{[n]}$ by: $X^* = (x_1, x_2\dots\ x_{i-1}, c_i, c_{i+1}, c_{i+2}, \dots c_n)$, if $i^+(X) = i$. Thus, we remark that $X \lneq X^+ \leq X^*$.
for every X of $B^{[n]}$, other than 1_F,
In the next sub-section, we establish several relations between product-order and lexicographical order.

3.3 Product-order and lexicographical order on $B^{[n]}$

Proposition 1: $\forall X$ and Y of $B^{[n]}$, if $X \leq Y$ then $X \underline{\angle} Y$

Proof: For every j of J $x_j \leq y_j$. If $Y < X$, $\exists\, j$ of J such as $y_j < x_j$. This is in contradiction with the fact that $X \leq Y$.

Proposition 2: whatever are X and Y of $B^{[n]}$, if X is different of 1_F, then: $X^+ \leq Y \leq X^*$ iff $X^+ \underline{\angle} Y \underline{\angle} X^*$.

Proof: It is obvious that $\forall X$, $X^+ \leq X^*$ and $X^+ \underline{\angle} X^*$. The implication from left to right results from 2.3.1.

Let us prove the second implication. We note $i = i^+(X)$ the transition subscript of X. Since $X^+ \underline{\angle} Y$,

- Either $X^+[j] = Y[j]$ for every j of J .i.e. $X^+ = Y$. Therefore, we conclude in this case that $X^+ \leq Y$ and $Y \leq X^*$;
- Or there is a subscript k of J such as $X^+[k] < Y[k]$. If $k < i = i^+[X]$, we have every $j < k$, $X^+[j] = Y[j]$, and $X^+[k] < Y[k]$. But for every $j < i$, we have $X^+[j] = X^*[j] = X[j]$. It follows that $X^* \angle Y$, in contradiction with the hypotheses. In that way, $i = i^+[X] \leq k$.

Consequently, for every $j < i$:

$X^+[j] = X^*[j] = X[j] = Y[j]$. Furthermore, for every $j > i$, we have $X^+[j] = 0 \leq Y[j] \leq c_j = X^*[j]$.

If $X^+ \underline{\angle} Y \underline{\angle} X^*$, we have thus for $j = i$, $X^+[i] = 1 + X[i] \leq Y[i] \leq c_i = X^*[j]$.

This proves that for every j of J, $X^+[j] \leq Y[j] \leq X^*[j]$, and thus that $X^+ \leq Y \leq X^*$.

3.4 Next closed itemset according to lexicographical order

For a given context $C = < I, F, d >$, with $F = B^{[n]}$, and an element a of F, we search the first element y of F which is a closed itemset of the lattice TG (C) and such as $a \angle y$.

The following result generalizes the Ganter's one.

Proposition 1: Let a^+ be the following of a in L.O. We pose $X = g(a^+) \subseteq I$, and $y = k(a^+) = f(X) \in F$.

If $a^+ \leq y \leq a^*$, then y is the first closed itemset, of TG (C), following a in L.O;

Otherwise, there is no closed item z of TG (C) between a^+ and a^*, and the following closed itemset of a in L.O is to be searched from the next of a^*.

Proof:

Since k is extensive, $a^+ \leq k(a^+) = y$.

- We suppose that we have furthermore $y \leq a^*$. Then $a^+ \leq y \leq a^*$.

According to 2.3.2, $a^+ \leq y \leq a^*$. y is a closed item which follows a in L.O. Let us prove that it is the first one. If there was one closed item z of TG (C), such as $a^+ \underline{\angle} z \underline{\angle} a^*$, then $a^+ \leq z \leq a^*$. Since k is increasing, we have $y = k(a^+) \leq k(z) = z$. Therefore $y \leq z$, and according to 2.3.1, $y \underline{\angle} z$.

Consequently, y is the first closed item of TG (C) following a in L.O.

- On the other hand, if $a^* < y$, let us prove that there is no closed item z of TG (C) such as $a^+ \underline{\angle} z \underline{\angle} a^*$.

If a such z existed, then $a^+ \leq z \leq a^*$. Since k is increasing, we have $y = k\,(a^+) \lneq k\,(z) = z$. Finally, since $z \leq a^*$, we have $y \leq a^*$. This result is in contradiction with the hypothesis.

4 Ganter algorithm

The properties established previously lead to the first version of the Ganter algorithm for contexts C = <*I*, *F*, *d* > such as $F = B^{[n]}$ provided with the product-order.

<u>Procedure **GANTER1** (C: context)</u>
Var a, a^+, a^*, y: elements of F; X element of E; nf: integer; {nf = number of closed itemsets of the lattice}
Begin
    $nf = 0$; $a = O_F$;
    $X = g\,(a)$; $y = f\,(X)$;
    If ($y = a$) then begin $nf = 1 + nf$; show (X, y); **end;**
    While ($a < 1_F$) **do**
    Begin
        $a = a^+$;
        $X = g\,(a) = f\,(X)$;
        If ($y \leq a^*$) **then**
          begin
            $nf = 1 + nf$; show (X, y); $a = y$; **end;**
        else $a = a^*$;
    End;
End;

We can reduce the number of operations of this algorithm by taking into account the properties established above: If i denotes the transition subscript of a, it obvious that that

- By assigning a to O_F, we have $i = n$;
- The condition ($a < 1_F$) is equivalent to ($i > 0$);
- *The expression "$a = a^+$"* may be replaced by the expression "**next** (a, i, a^+, i^+); $i = i^+$", which provides a^+ as well as its transition subscript;
- The expression "If ($y \leq a^*$) "may be replaced by the expression "(If $y_j = a_j$ for j =1… i-1)".
- *The expression "$a = a^*$* "may be replaced by the expression "$i = i$-1".

That leads to a faster version of the Ganter (?) algorithm.

<u>Procedure **GANTER2** (*C*: context)</u>
Var a, a^+, a^*: elements of F; X element of E; i, i$^+$, nf: integer; {nf = number of closed itemsets of the lattice}
Begin

```
        nf = 0; a=O_F; i=n; X= g (a);y = f (X);
        If (y = a) then
            begin nf=1+nf; show (X, y); end;
        While (i > 0) then
        begin
                next (a, i, a+, i+); a = a+; i = i+; X = g (a); y = f (X);
                If (for every j < i we have y_j = a_j) then
                    begin nf =1+nf; show (X, y); a = y; i = n; end;
                else i = i-1;
        end;
End;
```

<u>Example:</u>

$F = F_1 \times F_2 \times F_3$

- F_1 : Size: short (1), medium (2), high (3)
  F_1 will be presented by $1 < 2 < 3$ (ordered set by the relationship ≤)
- F_2 : Weight: thin (0), fat (1)
  F_2 will be presented by $0 < 1$ (ordered set by the relationship ≤)
- F_3: Age: child (1), adolescent (2), adult (3)
  F_3will be presented by $1 < 2 < 3$ (ordered set by the relationship ≤)

	Size	Weight	Age
Marc	2	0	1
Cédric	2	0	3
Céline	1	1	2
Carine	3	1	2

Total number of closed pairs (X, z) of T = GL(C) = 8.
C_1 : X = { Marc, Céline, Carine }, z = (2,0,1)
C_2 : X = { Marc, Cédric, Céline, Carine }, z = (1,0,1)
C_3 : X = {Cédric, Carine }, z = (1,1,2)
C_4 : X = {Cédric, Céline, Carine }, z = (1,0,2)
C_5 : X = {Céline }, z = (2,0,3)
C_6 : X = {Céline, Carine }, z = (2,0,2)
C_7 : X = {Carine }, z = (3,1,2)
C_8 : X = {}, z = (3,1,3)

5 The segmented procedure of Ganter

In this section, we use the sequential character of the construction of TG (C) by the Ganter algorithm to split this construction process into many parts.

5.1 Construction of an interval of TG(C)

Let u and v be two elements of $B^{[n]}$ such as $0_F \angle u \angle v \angle 1_F$. We recall that the Ganter algorithm produces the closed itemset (X, y) of TG (C) in the lexicographical order of the elements y of $B^{[n]}$. Therefore, if we note TG (C, u, v) the set of closed itemset (X, y) of TG (C) such as $u \angle y \angle v$, and if we call it the interval [u, v[of TG(C), then it is possible to obtain this interval by applying the following procedure:

Procedure **GANTER_TRUNCATED** (C: context; u, v: elements of $B^{[n]}$; Var TG: set of pairs of (X, y) of E x F)

Var nf: integer; a, a^+, a^*, y: elements of F; X: element of E;
Begin
TG= ∅; nf: 0; a=u;
X=g (a); y = f (X);
If (y = a) **then begin** nf= 1 + nf; show (X, y); TG= TG ∪{(X, y)}; **end;**
While ($a \angle v$) do
begin
$a = a^+$; $X = g$ (a); y = f (X);
If ($y \le a^*$ and $y \angle v$)
then begin nf =1+nf; show (X, y); a=y; TG = TG ∪{(X, y)}; **end**
else a= a^*;
end;
End;

We think that it is possible to provide a more efficient version of this procedure by taking into account the same remarks as for Ganter2.

5.2 Distributed construction of TG(C)

Let (u[0], u[1] … u [p-1], u[p]) be a sequence of elements of $B^{[n]}$, strictly increasing according to the lexicographical order, and such as O_F = u[0] $\angle$ u[1] $\angle$...$\angle$ u[p-1] $\angle$ u[p] = 1_F.
In the rest of the paper, such a sequence is called a partition of $B^{[n]}$.
TG (C) may be built as the union of the TG (C, u [k-1], u [k]), for k =1, 2… p.
In that way, we obtain all the closed itemset (X, y) of TG (C) such as $Y \in [0_F, u[1][\cup [u[1], u[2][\cup ... \cup [u[p-1], 1_F[$.
Let us remark that this construction excludes the pair (X, y) such as $y = 1_F$.
1_F is a closed item, since for every y of F we have $y \le k(y)$. Thus, $1_F \le k(1_F)$. Nevertheless $k(1_F)$ is an element of F and thus $k(1_F) \le 1_F$. That confirms that $y = 1_F$ is a closed item.

Furthermore $g\,(1_F) = \{i \in I\colon 1_F = d\,(i)\} = \{i \in I\colon 1_F = d\,(i)\}$. Consequently TG ($C$) = TG ($C$, $u[0]$, $u[1]$) $\cup \ldots \cup$ TG(C, $u[p\text{-}1]$, $u[p]$) $\cup$ $\{(g\,(1_F), 1_F)\}$.
Besides, we obtain a procedure to build TG (C), all the pairs of closed itemsets of the Galois lattice associated to the context C.

Procedure **GANTER_PARTITION** (C: context; u: partition of $B^{[n]}$; Var TG: ensemble of pairs (X, z) of E x F)
Var k: integer;
Begin
TG= $\emptyset$;
For k = 1 to p **do** TG=TG $\cup$ TG(C, $u\,[k\text{-}1]$, $u\,[k]$);
TG= TG $\cup$ $\{(g\,(1_F)), 1_F)\}$;
End;

The value of TG provided by this algorithm is TG (C).

6 The choice of a partition

We often use the segmentation of $B^{[n]}$ because this set is with strong cardinality. Moreover, since the algorithm of Ganter consists essentially in making a path in L.O of this set, we can intend to split this path in many separate intervals to be computed by different processors. Therefore the problem to be solved is equivalent to use partitions ($u[0]$… $u[p]$) which are as adequate as possible. A partition is called adequate if it allows sharing the workload by taking into account the capacity of the various available processors.
Let us note $b^{[n]} = |\,B^{[n]}\,|$ the cardinality of $B^{[n]} = b_1.\, b_2 \ldots b_n$. If there are p processors of the same capacity, we can, for example, intend to allocate to each processor, the exploration of an interval $[u[k\text{-}1], u[k][$of $B^{[n]}$ (with length = $b^{[n]} / p$). Consequently, the total workload between processors will be equally distributed.
It is also possible to design partitions which attribute to the processors the exploration of intervals with amplitudes proportional to their respective powers. From this point of view, the choice of partition, proposed in [11], is unbalanced.
Let us recall this choice proposed in the binary case:

- We have $b_i = 2$, for every j of $J = \{1\ldots n\}$;
- For every k of J, $\delta_{n,k}$ the vector of $B^{[n]} = B_1 \text{ x } B_2 \text{ x } \ldots \text{ x } B_n = \{0, 1\}^n$, where all the constituents are null, except the k-th which is equal to 1.

Let us denote $u\,[k] = \delta_{n,n\text{-}k+1}$, for $k = 1,\ldots, n$, $u\,[0] = O_F = (0,\ldots, 0)$, $u[n+1] = 1_F = (1,\ldots, 1)$, and $p = n+1$.
Then, the sequence ($u[0]$, $u[1]$, $u[2]$...$u[p\text{-}1]$, $u[p]$) is lexicographically increasing, and it can be used to partition $B^{[n]}$. It is easy to note that:

- The number of elements of $B^{[n]}$ following $\delta_{n,k}$ in L.O , is $2^n - 2^{n\text{-}k}$, for $k = 1,\ldots, n$;
- The number of elements of $B^{[n]}$ following $u[k] = \delta_{n,n\text{-}k+1}$ in L.O, is thus $2^n - 2^{k\text{-}1}$;

So that the number of elements of $B^{[n]}$ contained in L.O in the interval $[u[k\text{-}1], u[k][$ is:

$\Delta_{n,k} = (2^n - 2^{k-2}) - (2^n - 2^{k-1}) = 2^{k-2}$, for $k = 2, \ldots, n+1$.

Then $\Delta_{n,k} = 2.\ \Delta_{n,k-1}$. That indicates that the amplitudes of the intervals of the partition are in a geometrical progress of reason 2. Every interval being twice as large as the precedent, the distribution of the tasks is extremely unbalanced. If the first processor has to do a certain work, the second will have to do twice as much, the third four times more, etc. We also note that the last processor has to do the half of the total work, the last but one processor the quarter of the work.

In that way, any other partition which uses only a part of the $\delta_{n,k}$ would be also unbalanced. In the next section, we present some mathematical tools that will help us to build better-balanced partitions.

7 Mathematical tools

7.1 Rank of an element of B [n] in lexicographical order

In this section, we assume that all the b_j can be different. This corresponds to the general case. Our goal is to define a ranking function which permits associating to each element X_n of $B^{[n]}$ its rank in $B^{[n]}$ in lexicographical order. Then we shall prove that this function defines an isomorphism of ordered sets.

$B^{[n]}$ and X_n may be defined recursively as following:

- $B^{[n]} = B_1$, if $n = 1$, and $B^{[n]} = B^{[n-1]} \times B_n$, if $n > 1$.
- Any element X_n of $B^{[n]}$ by $X_n = (x_1)$, if $n = 1$, and $X_n = (X_{n-1}, x_n) \in B^{[n-1]} \times B_n$, if $n > 1$.

In the same way, we define recursively the mapping ρ_n: $B^{[n]} \to N$: for every X_n of $B^{[n]}$:

If $n = 1$, then $\rho_n (X_n) = x_n$, and if $n > 1$, then $\rho_n (X_n) = x_n + b_n .\ \rho_{n-1} (X_n - 1)$.

Let us prove the following property:

<u>Property 7.1:</u> ρ_n is a bijection between $B^{[n]}$ and the set $[0, b^{[n]} - 1]$.

Proof: we proceed by induction on $n > 0$.

The property is true for $n = 1$. Let us suppose that we have established it until n-1. Let us show that it is true for n.

- Let us begin by showing that for every X_n of $B^{[n]}$, we have $0 \leq \rho_n (X_n) \leq b^{[n]} - 1$.

Indeed, by hypothesis of induction (I.H, in summary), we have $0 \leq \rho_{n-1} (X_{n-1}) \leq b^{[n-1]} - 1$. On the other hand since $0 \leq x_n \leq b_n - 1$, we have $0 + b_n . 0 = 0 \leq \rho_n (X_n) \leq b_n - 1 + b_n . (b^{[n-1]} - 1) = b_n . b^{[n-1]} - 1 = b^{[n]} - 1$.

- Let us prove that ρ_n is injective. If two elements of $B^{[n]}$ are X_n and Y_n such as $\rho_n (X_n) = \rho_n (Y_n)$

We thus have $\rho_n (X_n) = x_n + b_n . \rho_{n-1} (X_{n-1}) = \rho_n (Y_n) = y_n + b_n . \rho_{n-1} (Y_{n-1})$.

If $\rho_{n-1} (X_{n-1}) < \rho_{n-1} (Y_{n-1})$, we have:

$$x_n - y_n = b_n . (\rho_{n-1} (Y_{n-1})) - \rho_{n-1} (X_{n-1})) \geq b_n - 1 = b_n.$$

This is impossible, because $x_n - y_n \lesssim x_n \lesssim b_n - 1$.
In the same way, we cannot have $\rho_{n-1}(X_{n-1}) > \rho_{n-1}(Y_{n-1})$. It follows that $\rho_{n-1}(X_{n-1})=\rho_{n-1}(Y_{n-1})$,. Thus $x_n = y_n$. Now, by I.H, ρ_{n-1} is injective. Consequently, $X_{n-1} = Y_{n-1}$, and $X_n = (X_{n-1}, x_n) = (Y_{n-1}, y_n) = Y_n$.
- Let us prove that ρ_n is surjective. It is necessary to show that for every integer a, such as $0 \le a \lesssim b^{[n]} - 1$, there is an element X_n of $B^{[n]}$ such as $\rho_n(X_n) = a$. According to the Euclidian division of r by b_n, there is a unique pair of naturel integers (q, r) such as $a = b_n . q + r$, with $r < b_n$. Furthermore $0 \le q \le (b^{[n]} - 1) / b_n$, $0 \le q \lesssim b^{[n-1]} - 1$. Let us define $x_n = r$. By I.H, ρ_{n-1} is surjective. Thus, there is an element X_{n-1} of $B^{[n-1]}$ such as $\rho_{n-1}(X_{n-1}) = q$,. Furthermore, we have $0 \le x_n \lesssim b_n - 1$. So $X_n = (X_{n-1}, x_n)$ is an element of $B^{[n]}$, and we have $\rho_n(X_n) = x_n + b_n . \rho_{n-1}(X_{n-1}) = r + b_n. q = a$. Q.E.D.
The following procedures written in pseudo Pascal allow computing the function **rank**: $B^{[n]} \to N$ as well as its inverse Expansion: $N \to B^{[n]}$. Knowing $n, b_1 \dots b_n$, and X_n de $B^{[n]}$, the first one computes $\rho_n(X_n)$. For every integer a such as $0 \le a \lesssim b^{[n]} - 1$, the second procedure, determines the element X_n of $B^{[n]}$ such as $\rho_n(X_n) = a$.

```
Function RANK (n, b1… bn: integer; Xn: element of B [n]): long integer;
Var i: integer; a: long integer;
Begin
        a= 0;
        For i= 1 to n, do a= x [i] + a* b [i];
        Rank=a;
End;
```

```
Procedure EXPANSION (n, b1… bn: integer; a: long integer; var X: element of
B[n]);
Var s: long integer; i: integer;
Begin
        s =a;
        For i= n down to 1 do
         begin
                X [i]= s mod b [i];
                s=s div b [i];
         end;
End;
```

Remark: The inverse function of **rank** is called **expansion**, because it provides the expression of the integer a in the multiple or heterogeneous base $(b_1, b_2 \dots b_n)$. This writing is $X_n = (x_1, x_2 \dots x_n)$.
Reciprocally, every X_n of $B^{[n]}$ can be viewed as the writing in base $(b_1 \dots b_n)$ of the integer a which is equal to $\rho_n(X_n)$.
Remark 2: the function rank was defined in a recursive way. We can need an explicit formulation which can be obtained by using the following formula:

$\rho_n (X_n) = x_n + b_n . x_{n-1} + b_n. b_{n-1}. x_{n-2} + \dots + b_n. b_{n-1} \dots b_2. x_1.$

7.2 Isomorphism of ordered sets

<u>Proposition :</u> ρ_n is an isomorphism between $B^{[n]}$ provided with lexicographical order $\underline{\angle}$, and $[0, b^{[n]} - 1]$ provided with the order $\leq$ of the naturel integers.
Proof: By induction on n. The property is obvious for $n = 1$. Let us suppose we have proved it until n-1. Let us prove that it is true for n. We have $X_n = (X_{n-1}, x_n)$, and $Y_n = (Y_{n-1}, y_n)$. If we suppose that $X_n \underline{\angle} Y_n$. then we have $\rho_n(X_n) \leq \rho_n(Y_n)$. Indeed we have $X_n \underline{\angle} Y_n$ if $(X_{n-1} \angle Y_{n-1})$, or $(X_{n-1} = Y_{n-1}$ and $x_n < y_n)$.
- in the second case, we have $\rho_n (X_n) = x_n + b_n. \rho_{n-1} (X_{n-1}) < y_n + \rho_{n-1} (X_{n-1}) = \rho_n (Y_n)$.
- In the first case, by I.H, we have $\rho_{n-1} (X_{n-1}) < \rho_{n-1} (Y_{n-1})$, and therefore $\rho_n (Y_n) - \rho_n (X_n) = b_n. (\rho_{n-1} (Y_{n-1}) - \rho_{n-1} (X_{n-1})) + y_n - x_n \geq b_n . 1 + y_n - x_n$. Since $0 \leq x_n,\ y_n \leq b_n - 1$, we can conclude that $\rho_n (Y_n) - \rho_n (X_n) > 0$. So $X_n \underline{\angle} Y_n$. $\rho_n (X_n) \leq \rho_n(Y_n)$. Reciprocally, let X_n and Y_n of $B^{[n]}$ such as $\rho_n (X_n) \leq \rho_n (Y_n)$. Let us show that $X_n \underline{\angle} Y_n$.
- If $Y_{n-1} \angle X_{n-1}$, then, by I.H, $\rho_{n-1} (Y_n) < \rho_{n-1} (X_{n-1})$. In that way, we have $\rho_n (X_n) - \rho_n (Y_n) = b_n. (\rho_{n-1} (X_{n-1}) - \rho_{n-1}(Y_{n-1})) + x_n - y_n \geq b_n - 1 + x_n - y_n > 0$. This is in contradiction with the hypothesis.
- If $X_{n-1} = Y_{n-1}$ and $y_n < x_n$, we still have then for every integer a such as $0 \leq a \leq b^{[n]} - 1$ for every integer a such as $0 \leq a \leq b^{[n]} - 1$ $\rho_n (X_n) - \rho_n (Y_n) > 0$, in contradiction with the hypothesis CQFD.

7.3 Addition of two numbers written in a multiple base

Let $(b_1, b_2 \dots b_n)$ be the multiple base and x, y two natural integers such as $0 \leq x, y \leq b^{[n]} - 1$. The respective expressions of these two numbers in base $(b_1, b_2 \dots b_n)$ are X_n and Y_n. What is the expression of $z = x + y$ in this base?
In a different way: if we know the writings X_n and Y_n of two numbers x, y in base $(b_1 \dots b_n)$, without knowing x and y, is-it possible de compute the expression Z_n of z, without having to compute $z=x+y$?
The solution consists in making the addition of X_n and Y_n in the base. The following procedure implements this operation. It uses an auxiliary sequence of nature integers $r_0, r_1 \dots r_n$ which play the role of the "carries".

```
Procedure ADDITION (n, b1,…, bn: integer; X, Y: elements of B[n]; var Z: element of B[n]; var r: integer);
Var r[0..n]: integer; i: integer; u:entier;
Begin
        r[n]= 0;
        For i = n down to 1 do
        begin
                u = x[i] + y i] + r [i];  If (u < b [i]) Then
         begin z [i]= u; r [i-1]= 0; end
           else begin z [i]= u - b [i]; r [i-1]= 1; end;
```

```
        end;
        r = r [0];
End;
```

In this procedure, r plays the role of the final carry. The addition of X_n and Y_n in base ($b_1 \ldots b_n$) will be noted $X_n \oplus Y_n$.
Let us show that the procedure **ADDITION** provides the expected result.

- For $i = n$, n-1,...1, we add x_i and y_i and the carry r_i. Since for every i we have $0 \le x_i, y_i \le b_i - 1$, and $r_i < 2$, we obtain that $0 \le u = x_i + y_i + r_i \le 2b_i - 1$. If $0 \le u \le b_i - 1$, we put
- $z_i = u$, and $r_{i-1} = 0$; and else $z_i = u - b_i$, and $r_{i-1} = 1$. Therefore, for every i of 1...n we have
- $0 \le z_i \le b_i - 1$, and $0 \le r_i \le 1$. The final carry $r = r_0$, is also equal to 0 or 1.
- Since for every i of {1... n} $0 \le z_i \le b_i - 1$, $Z = (z_1 \ldots z_n)$ belongs to $B^{[n]}$.
- $X_n \oplus Y_n$. is equal to the output Z, of the procedure of addition

Now we establish the following proposition:
<u>Proposition 7.3:</u> Whatever are the elements X_n and Y_n of $B^{[n]}$, we have $\rho_n (X_n \oplus Y_n.) = \rho_n (X_n) + \rho_n (Y_n). - r_0 . b^{[n]}$, with $r_0 = 0$ or 1.
Proof: At every step i of the procedure **ADDITION**, we have noted that:
If $u < b_i$, then
$z_i = u$ and $r_{i-1} = 0$;
else $z_i = u\text{-}b_i$, and $r_{i-1} = 1$;
end;
Therefore for every i, we can write that $z_i = u - b_i . r_{i-1}$, $z_i = x_i + y_i + r_i - b_i . r_{i-1}$.

8 Construction of good partitions

The tools, which we introduced above, will help us to build any partition of $B^{[n]}$.
Let us suppose that we have p processors to determine TG (C), knowing that $F = B^{[n]}$.
Suppose that these processors are of respective capacities $c_1, c_2 \ldots c_p$, (c_k = number of closed that the processor k can build in a given lapse of time), and that $c_1 + c_2 \ldots + c_p \geq b^{[n]} = b_1.b_2 \ldots b_n$. We can build a sequence of integers $a_0, a_1 \ldots a_l$, in the following way:
l: 0; $a_l = 0$;
while ($a_l < b^{[n]} - 1$) do begin $l = l + 1$; $a_l = a_{l-1} + c_l$; end;
$a_l = b^{[n]} - 1$.
In that way, we obtain a strictly increasing sequence ($a_0, a_1 \ldots a_l$) such as $a_0 = 0$, and $a_l = b^{[n]} - 1$. Furthermore for $k = 1\ldots$ l, we have $c_k = a_k - a_{k-1}$.
Then, for $k = 0, 1\ldots$ l, we build the elements u_k of $B^{[n]}$, by computing **EXPANSION** ($n, b_1 \ldots b_n, a_k, u_k$), which provides the writing u_k of a_k in base ($b_1 \ldots b_n$).
This sequence ($u_0, u_1 \ldots u_l$) constitutes a good partition of $B^{[n]}$, since it is lexicographically strictly increasing one, with $u_0 = O_F$, $u_l = 1_F$, and since it distributes the workload by taking into account the capacity of all the processors.

However, although mathematically correct, this method is difficult to implement. Indeed, when n is large, (even for n = 30), the number $b^{[n]} = b_1.b_2 \ldots b_n$ is equal at least to 2^n, and it can reach very large values, which are hard to compute with the current processors. Therefore, the procedure **EXPANSION** (n, b… a_k, u_k) is difficult to execute with big values of the parameter a_k.
To overcome this difficulty we can use additions in base (b_1… b_n) as in the following case of equi partition:
Let us suppose that all the processors have the same capacity c. We determine the smallest integer p equal or bigger than $b^{[n]} / c$.
Then we determine the expression uc of c in base (b_1…b_n), by executing **EXPANSION** (n, b_1… b_n, c, uc). Finally, we build the sequence of elements u_k of $B^{[n]}$ by the mean of the following instructions:

$k = 0$; $u[k]= 0_F$; $r = 0$;
while (r = 0) **do**
begin
    $k=k+1$;
    ADDITION (n, b_1, …, b_n, $u[k\text{-}1]$, uc, $u[k]$, r);
end;
$p = k$; $u[p]= 1_F$;

This procedure returns $u_0 = 0_F$; $u_1 = u_0 \oplus uc = uc$, $u_2 = u_1 \oplus uc$… until the carry r becomes equal to 1.
While $r = 0$, by the property 5.3, we have $\rho_n (X_n \oplus Y_n) = \rho_n (X_n) + \rho_n (Y_n)$. Thus the addition of writings in base (b_1… b_n) is equivalent to the addition of the corresponding numbers.
By this way we built the sequence (u_0, u_1… u_p) of the partition, without computing (a_0, a_1… a_p) ranks $a_k = \rho_n (u_k)$. We stop when r takes the value 1. One then take $p = k$, and we set $u_p = 1_F$.

9 Conclusion

We have proposed a generalization of the Ganter algorithm, as well as a distributed version of this algorithm.
On the one hand, this generalization allows us to determine the Galois lattices associated to rather general contexts, without needing to re-code the data into binary values. On the other hand, when one analyze the relationship between product-order and lexicographical order on the Cartesian product $B^{[n]} = B_1 \times B_2 \times \ldots \times B_n$, with $B_j=\{0, b_j -1\}$, for j=1… n, this generalization seems to be natural. Moreover, viewing the elements of $B^{[n]}$ as expressions of integers in base (b_1… b_n) allows us to obtain good partitions of $B^{[n]}$ and to simplify the calculations. From a practical point of view, we can intend to apply the procedure of segmentation of $B^{[n]}$ to very large contexts.

This approach seems more efficient than the context-based approaches proposed in [1] and [14] which need to compute the Cartesian product of two Galois lattices.

Finally, let us note that the algorithm proposed in this paper is based on a lexical search through the space $B^{[n]}$ of the attributes. While being more general than [11], it cannot determine the most general Galois lattice. To reach this target, we need algorithms which search throw the set P(*I*) of all subsets of individuals. This could be obtained either by searching this space in lexicographical order, or by obtaining a partition of this space and to share the workload of solving the corresponding sub-lattices between several processors. This is a possible future research direction.

References

1. Baklouti F, Lévy G. Parallel algorithms for general Galois lattices building. Proc. WDAS 2003.
2. Birkhoff. G: Lattice Theory. American Mathematical Society, Providence, RI, 3rd edition. (1967)
3. Emilion. R. Lambert. G, Lévy. G, Algorithmes pour les treillis de Galois, Indo-French Workshop., University
Paris IX-Dauphine. 1997.
4. Diday. E, Emilion. R : Treillis de Galois maximaux et capacités de Choquet. C.R.Acad.Sci. Paris, t.325, (1997) Série I. p.261-266.

5. Ganter, B: Two basic algorithms in conceptual analysis. Technical report, Darmstadt University (1984).
6. G. Lambert, R. Emilion, G. Lévy : Algorithmes pour construire les treillis de Galois généraux. To appear.
7. Ganter. B: Two basic algorithms in concept analysis. Preprint 831, Technische Hochschule Darmstadt (1984).

8. Ganter. B, Wille. R: Formal Concept Analysis. Mathematical Foundations, Springer. (1999).

9. Ganter. B : Formal Concept Analysis: algorithmic aspects. TU Dersden. (2002).

10. Godin. R : Complexité de structures de treillis. Ann. SC. Math., (1989) Quebec, 13 (1): 19-38.

11. Huaiguo Fu, Engelbert Mephu Nguifo: A fast scalable algorithm to build closed item sets for large data. Third IASTED International Conference on Artificial Intelligence and Applications (2003).
12. Nourine. L., Raynaud. O: A fast algorithm for building Lattices. Information Processing Letters 71, (1999) 199.
13. Njiwoua. P, Mephu Nguifo. E: A parallel algorithm to build concept lattice. In Proceedings of 4th Groningen Intl. Information Tech. Conf. For Students, (1997) pp. 103-107.
14. Valtchev P., Missaoui R., Lebrun P., A partition based approach toward construction Galois (concept) lattices. Discrete Mathematics 256(2002) 801-829.
15. Wille. R: Concept Lattices and Knowledge Systems. Computer Mathematic Applied, 23 (6-9), (1992) 493-515.