

# A distributed version of the Ganter algorithm for general Galois Lattices

Gérard Lévy, Fatma Baklouti  
[Gerard.Levy@dauphine.fr](mailto:Gerard.Levy@dauphine.fr), [fatma.baklouti@dauphine.fr](mailto:fatma.baklouti@dauphine.fr)  
CERIA, Paris 9 Dauphine University

The paper [1]: *A fast scalable algorithm to build closed itemsets for large data*, of Huaiguo Fu and Engelbert Mephu Nguifo, leans on the sequential character of the determination of the closed itemsets of a Galois Lattice by the Ganter algorithm, to share the work of production between several processors when the number of closed itemsets to determine is very big. (The qualifier "large" concerns, in this situation, the number of closed itemsets of the lattice and the grate size of the Galois connection, because a small table can very well engender a big lattice). One of the essential points of this distributed version is that of the partition of the work to be made, that is of the work distribution between various processors.

We suggest here showing that:

- The Ganter algorithm can apply to contexts more general than usual binary contexts;
- The parallelised version of H.Fu and E. Mephu, can extend in these new contexts;
- The partition of the work between processors can be made with all the wished precision.

To define cleanly the general contexts to which we shall apply our algorithms, and to develop the tools of partition, we are going to introduce certain number of definitions and concepts.

## 1. Notations, definitions:

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

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

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

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

We note  $B^{[n]}$  the *Cartesian product*  $B_1 \times B_2 \times \dots \times B_n$ . The set of the *elements*  $X_n = (x_1 \dots x_n)$  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 endowed with the operations of sup noted  $\vee$  and of inf noted  $\wedge$ .

These operations are defined by:

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

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

Furthermore, this lattice has for extreme elements  $O_F$  and  $I_F$ , which are defined by  $O_F = (0 \dots 0 \dots 0)$ , and  $I_F = (b_1 - 1 \dots b_j - 1 \dots b_n - 1)$ .

And we have for every  $X_n$  of  $B^{[n]}$ :  $O_F \leq X_n \leq I_F$ .

(For the rest of the paper, to avoid using  $b_j - 1$ , we shall put  $I_F = (c_1 \dots c_j \dots c_n)$ , with  $c_j = b_j - 1$ , for ever  $j$  of  $J$ )

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

Let be an application  $d: I \rightarrow B^{[n]}$ . It associates to every individual  $i$  of  $I$  its *description*  $d(i) = (d_1(i) \dots d_j(i) \dots d_n(i))$ , with  $d_j(i) =$  value of the variable  $j$  for the individual  $i$ .

We can now define the application  $f: E \rightarrow B^{[n]}$ , who associates to every subset  $X$  of  $I$  the element  $f(X)$  of  $B^{[n]}$  defined by  $f(X) = I_F$ ;  $X = \emptyset$ , and  $f(X) = \bigwedge_{i \in X} d(i) = d(i_1) \wedge d(i_2) \wedge \dots \wedge d(i_k)$  if  $X = \{i_1, i_2 \dots i_k\} \subseteq I$ . ( $f(X)$  is called the *intent* of  $X$ ).

We define the application  $g: B^{[n]} \rightarrow E$  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  $Z_n$ ).

$f$  and  $g$  are decreasing applications and their composites  $h = g \circ f$  and  $k = f \circ g$  are closure operators.

In the literature of « data mining », the triplet  $C = \langle I, F, d \rangle$  is called a *context*. It is often realized by a rectangular table, which has  $m$  lines and  $n$  columns, every element in position  $(i, j)$  being equal to  $d_j(i)$ .

And  $(f, g)$  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 show that these sets are in bijection.

The set of pairs  $(X, Z_n)$  of  $E \times B^{[n]}$  such as  $(X \in Inv(E) \text{ and } Z_n = f(X))$  is equal to the set of all the pairs  $(X, Z_n)$  such as  $(Z_n \in Inv(F) \text{ and } X = g(Z_n))$ . (Each of these pairs is called a *concept*). It is this set that we call the *Galois Lattice* associated to the context  $C$ , and that we shall note  $TG(C)$ .

$TG(C) = \{(X, Z_n): X \in Inv(E) \text{ and } Z_n = f(X)\} = \{(X, Z_n): Z_n \in Inv(F) \text{ and } X = g(Z_n)\}$ .

We can show that this set indeed has the structure of a lattice for the relation of *order*  $\leq$  defined by  $(X, Z_n) \leq (X', Z_n')$  iff  $X \subseteq X'$  and  $Z_n' \leq Z_n$ .

Number of problems of data mining resort to the determination of the Galois Lattice associated to such or such context  $C$ . And, according to the case, this determination can consist in determining simply all the concepts  $(X, Z_n)$ , or to determine this set and its order relation.

## 2. Lexicographical Order on $B^{[n]}$ :

Up to here we saw that  $B^{[n]}$  is provided with the order produced by  $B_1 \times B_2 \times \dots \times B_n$ .

When all the  $b_j$  are equal to 2, that is in the classic or standard case, the Ganter algorithm allows to build all the closed itemsets of the Galois lattice following the lexicographical order of the elements of it.

That is why we are exactly going to define this order on  $B^{[n]}$  when  $b_j$  is superior to 1, but without being inevitably equal to 2, nor equals between them, to generalize the Ganter algorithm.

### 2.1: Lexicographical order on $B^{[n]}$

Being given two elements  $X_n$  and  $Y_n$  of  $B^{[n]}$ , we look their biggest common prefix. Two cases are possible: either we have  $X_n = Y_n$ , either they are different by at least a coordinated. If it is the first coordinated by which they differ and if  $x_j < y_j$  then  $X_n$  precedes  $Y_n$  in lexicographical order (O.L), and otherwise  $Y_n$  precedes  $X_n$ . We shall note  $X_n \leq Y_n$  the fact that  $X_n$  precedes  $Y_n$  in O.L.

In pseudo-Pascal, the following function takes the value True iff  $X_n \leq Y_n$ .

Function **inflex** ( $X_n, Y_n$ : elements of  $B^{[n]}$ ): boolean;

The Var j: integer;

Begin

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

End;

### 2.2: The next in lexicographical order of $X_n$

With the covenant  $I_F = (c_1 \dots c_j \dots c_n)$ , we define the subscript of transition of every  $X_n$  de  $B^{[n]}$  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 subscript of transition is  $i$ .

We shall note it  $i^+(X_n)$ ,  $i^+(X_n) = 0$ , iff  $X_n = I_F$ .

At the moment, if  $X_n \neq I_F$ , and if its subscript of transition is  $i$ , we define its following  $Y_n$  in lexicographical order by  $Y_n = (x_1, x_2 \dots x_{i-1}, I + x_i, 0, 0 \dots 0)$ .

And we shall note it  $X_n^+$ . (We shall easily verify that  $Y_n$  is the following of  $X_n$  in O.L) On the other hand, if  $X_n = I_F$ , the following of  $X_n$  in O.L is not defined.

We can also define, the next of  $X_n$  in O.L, 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 subscript of transition of  $X$ , this procedure produces merely  $Y = X^+$ , the next of  $X$  in O.L, whom we would obtain normally by leaving of  $i = n$ .

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  $I_F$ , 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  $i^+(X) = i$ .

We thus see that for every  $X$  of  $B^{[n]}$ , other than  $I_F$ , we note  $X \leq X^+ \leq X^*$ .

We are now going to establish some relations between product-order and lexicographical order.

### 2.3: Product-order and lexicographical order on $B^{[n]}$ :

Property 2.3.1:  $\forall X$  and  $Y$  of  $B^{[n]}$ , if  $X \leq Y$  then  $X \leq Y$

**Proof:** for every  $j$  of  $J$   $x_j \leq y_j$ . If  $Y \leq X$ ,  $\exists j$  of  $J$  such as  $y_j < x_j$ . In contradiction with the fact that  $X \leq Y$ .

Property 2.3.2: whatever  $X$  and  $Y$  of  $B^{[n]}$ , if  $X$  is different of  $I_F$ , then:  $X^+ \leq Y \leq X^*$  iff  $X^+ \leq Y \leq X^*$ .

**Proof:**

We have naturally,  $\forall X, X^+ \leq X^*$  and  $X^+ \leq X^*$ . The implication from left to right results from 2.3.1.

Let us prove the second implication.  $i = i^+(X)$ , the subscript of transition of  $X$ . Because  $X^+ \leq Y$ .

- Either  $X^+ [j] = Y [j]$  for every  $j$  of  $J$ , that is  $X^+ = Y$ , and in that case we thus have  $X^+ \leq Y$  and  $Y \leq X^*$ ;

- Or there is a subscript  $k$  of  $J$  such as  $X^+ [k] < Y [k]$ . If  $k < i = i^+(X)$ , we would 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 would follow that  $X^* < Y$ , contrary to the hypotheses. We have thus inevitably  $i = i^+(X) \leq k$ . And, 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  $X^+ \leq Y \leq X^*$ , we have thus inevitably for  $j = i$ ,  $X^+ [i] = 1 + X [i] \leq Y [i] \leq c_i = X^* [i]$ .

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

### 2.4: Next closed itemset according to lexicographical order:

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

The following result generalizes that of Ganter.

#### Proposition 2.4.1:

Let  $a^+$  be the following of  $a$  in O.L. 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 of  $a$  in O.L;

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

**Proof:**  $k$  being extensive,  $a^+ \leq k(a^+) = y$ .

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

According to 2.3.2, we thus have  $a^+ \leq y \leq a^*$ .  $y$  is a closed item which follows  $a$  in O.L. Let us show that it is the first one. If there was one closed item  $z$  of  $TG(C)$ , such as  $a^+ \leq z \leq a^*$ , we would also have  $a^+ \leq z \leq a^*$ , and because  $k$  is increasing, we would have  $y = k(a^+) \leq k(z) = z$ . So that we would have  $y \leq z$ , and according to 2.3.1,  $y \leq z$ .

What proves that  $y$  is the first closed item of  $TG(C)$  following of  $a$  in O.L.

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

In fact, if a such  $z$  existed, we would also have  $a^+ \leq z \leq a^*$ , and because  $k$  is increasing, we would have  $y = k(a^+) \leq k(z) = z$ . And, because  $z \leq a^*$ , we would have  $y \leq a^*$ , in contradiction with the hypothesis.

### **3. Ganter algorithm:**

The properties which we have just established lead to the first version of the Ganter algorithm for contexts  $C = \langle I, F, d \rangle$  such as  $F = B^{[n]}$  provided with the product-order.

```

Procedure GANTER1 (C: context);
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 := OF;
  X := g (a); y := f (X);
  If (y = a) then begin nf: =1+nf; show (X, y); end;
  While (a < IF) do
  Begin
    a := a+;
    X := g (a); y := f (X);
    If (y ≤ 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 properties established above: always by indicating by  $i$  the subscript of transition of  $a$ , we see that

- By initializing  $a$  to  $O_F$ , we have  $i = n$ ;
- The condition  $(a < I_F)$  is translated by  $(i > 0)$ ;
- $a = a^+$  can be replaced by **next**  $(a, i, a^+, i^+)$ ;  $i = i^+$ ; which supplies well  $a^+$  as well as its subscript of transition;
- If  $(y \leq a^*)$  can be replaced by (If  $y_j = a_j$  for  $j = 1 \dots i-1$ ).
- $a = a^*$  can be replaced by  $i = i-1$ .

What leads to a faster version of the algorithm:

```

Procedure GANTER2 (C: context);
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 := OF; 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 yj = aj) then
      begin nf: =1+nf; show (X, y); a := y; i := n; end
    else i := i-1;
  end;
End;

```

#### 4. Procedure of Ganter segmented:

We are going to lean on the sequential character of the construction of TG (C) by the Ganter algorithm to split this work of construction in several parts.

##### 4.1: Construction of an interval of TG (C):

Let two elements of  $B^{[n]}$   $u$  and  $v$  be such as  $O_F \leq u \leq v \leq I_F$ . Because the Ganter algorithm produces the closed itemset  $(X, y)$  of TG (C) in the lexicographical order of the elements  $y$  of  $B^{[n]}$ , if we note TG (C,  $u, v$ ) the set of closed itemset  $(X, y)$  of TG (C) such as  $u \leq y \leq v$ , and if we call it the interval  $[u, v]$  [of TG (C)], we see that it is possible to obtain this interval by the following procedure:

```

Procedure GANTER_TRUNCATED (C: context; u, v: elements of  $B^{[n]}$ ; Var TG: set of pairs of (X, y) of  $E \times F$ );
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 < v) do
  begin
    a := a+; X := g (a); y := f (X);

```

If ( $y \leq a^*$  and  $y \angle v$ )  
then begin  $nf = 1 + nf$ ; show  $(X, y)$ ;  $a := y$ ;  $TG = TG \cup \{(X, y)\}$ ; end  
else  $a := a^*$ ;

end;

End;

We can write a more efficient version of this procedure by taking into account the same remarks as for Ganter2.

#### 4.2: Distributed construction of TG (C):

Let  $(u [0], u [1] \dots u [p-1], u [p])$  be a sequence of elements of  $B^{[n]}$ , strictly increasing according to the lexicographical order, and such as  $O_F = u [0] \angle u [1] \angle \dots \angle u [p-1] \angle u [p] = I_F$ .

For the rest of the paper, we shall say that such a sequence determines a partition of  $B^{[n]}$ .

We can build TG (C) as the union of the TG  $(C, u [k-1], u [k])$ , for  $k = 1, 2 \dots p$ .

It will give in fact all the closed itemset  $(X, y)$  of TG (C) such as

$Y \in [0_F, u [1]] \cup [u [1], u [2]] \cup \dots \cup [u [p-1], I_F]$ .

This construction excludes the pair  $(X, y)$  such as  $y = I_F$ .

But we know that  $I_F$  is a closed item, because for every  $y$  of  $F$  we have  $y \leq k(y)$ , so that  $I_F \leq k(I_F)$ . But  $k(I_F)$  is an element of  $F$  and thus  $k(I_F) \leq I_F$ . What confirms that  $y = I_F$  is a closed item. Furthermore  $g(I_F) = \{i \in I : I_F = d(i)\} = \{i \in I : I_F = d(i)\}$ . And consequently  $TG(C) = TG(C, u [0], u [1]) \cup \dots \cup TG(C, u [p-1], u [p]) \cup \{(g(I_F), I_F)\}$ .

We thus have 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  $E \times F$ );

Var k: integer;

Begin

TG: =  $\emptyset$ ;

For k: = 1 to p do TG: = TG  $\cup$  TG(C, u [k-1], u [k]);

TG: = TG  $\cup$   $\{(g(I_F), I_F)\}$ ;

End;

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

### 5. Choice of a partition:

Mostly we resort to the segmentation of  $B^{[n]}$  because this set is of strong cardinality, and because the algorithm of Ganter consists essentially in making a path in O.L of this set, we can intend to split this path in several separate intervals which we shall make deal by different processors. The problem thus comes down to use partitions  $(u [0] \dots u [p])$  as adequate as possible. More exactly, we shall say that a partition is adequate if it allows sharing the work by taking into account the capacity of the various available processors.

Let us note  $b^{[n]} = |B^{[n]}| = \text{cardinality of } B^{[n]} = b_1 \cdot b_2 \dots b_n$ . If there are p processors of the same capacity, we can, for example, intend to confide to each of the investigation of an interval  $[u [k-1], u [k]]$  of  $B^{[n]}$ , of length  $b^{[n]}/p$ . What will distribute fairly the total working load between processors.

We can, also, design partitions, which attribute to the processors of the intervals of proportional amplitudes in them respective powers.

The choice of partition which is proposed in [1] is unbalanced, from this point of view.

Let us call back this choice, which is proposed in the binary case:

- We have  $b_i = 2$ , for every  $j$  of  $J = \{1 \dots n\}$ ;
- Let us call, for every  $k$  of  $J$ ,  $\delta_{n,k}$  the vector of  $B^{[n]} = B_1 \times B_2 \times \dots \times B_n = \{0, 1\}^n$ , whose all the constituents are null, with the exception of the  $k$ -th which is worth 1.

Let us take  $u [k] = \delta_{n,n-k+1}$ , for  $k = 1, \dots, n$ ,  $u [0] = O_F = (0, \dots, 0)$ ,  $u [n+1] = I_F = (1, \dots, 1)$ , and  $p = n+1$ .

So we see that the sequence  $(u [0], u [1], u [2] \dots u [p-1], u [p])$  is lexicographically increasing, and it can serve to partition  $B^{[n]}$ . It is easy to see that:

- the number of elements of  $B^{[n]}$  following  $\delta_{n,k}$  in O.L, is  $2^n - 2^{n-k}$ , for  $k = 1, \dots, n$ ;

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

So that the number of elements of  $B^{[n]}$  contained in O.L in the interval  $[u [k-1], u [k]]$  is  $\Delta_{n,k} = (2^n - 2^{k-2}) - (2^n - 2^{k-1}) = 2^{k-2}$ , for  $k = 2, \dots, n+1$ .

We see that  $\Delta_{n,k} = 2 \cdot \Delta_{n,k-1}$ . What indicates that the amplitudes of the intervals of the partition are in a geometrical progress of reason 2. Every interval being twice as big as the precedent, the distribution of the tasks is extremely

unbalanced. If the first processor has to make a certain work, the second will have to make twice as much, the third four times more, .... We also see that the last processor has to make half of the total work, the last front the quarter of the work, ...

So that any other partition which would use only a part of the  $\delta_{n,k}$  would be also unbalanced. In what follows, we are going to develop some mathematical tools that will allow us to build better-balanced partitions.

## 6. Mathematical tools:

### 6.1: Rank of an element of $B^{[n]}$ in lexicographical order.

Let us return to the general case, that where all the  $b_j$  can be different. We are going to define a function of rank which allows associating to every element  $X_n$  of  $B^{[n]}$  its rank in  $B^{[n]}$  in lexicographical order. We shall show then that this function defines an isomorphism of ordered sets.

We can define recursively  $B^{[n]}$  by  $B^{[1]} = B_1$ , if  $n = 1$ , and  $B^{[n]} = B^{[n-1]} \times B_n$ , if  $n > 1$ .

And 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  $n > 1$ .

Let us define the application  $\rho_n: B^{[n]} \rightarrow N$ , of a recursive way, for every  $X_n$  de  $B^{[n]}$ :

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

Let us prove the following property:

**Property 6.1:**  $\rho_n$  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 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  $0 \leq \rho_{n-1}(X_{n-1}) \leq b^{[n-1]} - 1$ . And, because  $0 \leq x_n \leq b_n - 1$ , we have  $0 + b_n \cdot 0 = 0 \leq \rho_n(X_n) \leq b_n - 1 + b_n \cdot (b^{[n-1]} - 1) = b_n \cdot b^{[n-1]} - 1 = b^{[n]} - 1$ .

- Let us show that  $\rho_n$  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  $\rho_n(X_n) = x_n + b_n \cdot \rho_{n-1}(X_{n-1}) = \rho_n(Y_n) = y_n + b_n \cdot \rho_{n-1}(Y_{n-1})$ .

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

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

What is impossible, because  $x_n - y_n \leq x_n \leq b_n - 1$ .

In the same way, we can't have  $\rho_{n-1}(X_{n-1}) > \rho_{n-1}(Y_{n-1})$ . It follows that  $\rho_{n-1}(X_{n-1}) = \rho_{n-1}(Y_{n-1})$ , and thus that  $x_n = y_n$ .

Now, by I.H,  $\rho_{n-1}$  is injective. Consequently, we have  $X_{n-1} = Y_{n-1}$ , and  $X_n = (X_{n-1}, x_n) = (Y_{n-1}, y_n) = Y_n$ .

- Let us show that  $\rho_n$  is surjective. It is necessary to show that for every integer  $a$ , such as  $0 \leq a \leq b^{[n]} - 1$ , there is an element  $X_n$  of  $B^{[n]}$  such as  $\rho_n(X_n) = a$ . For it let us make the Euclidian division of  $r$  by  $b_n$ . There is a unique pair of nature integers  $(q, r)$  such as  $a = b_n \cdot q + r$ , with  $r < b_n$ . Furthermore  $0 \leq q \leq (b^{[n]} - 1) / b_n$ ,  $0 \leq q \leq b^{[n-1]} - 1$ . Let us pose  $x_n = r$ . By I.H,  $\rho_{n-1}$  is surjective. There is thus an element  $X_{n-1}$  of  $B^{[n-1]}$  such as  $\rho_{n-1}(X_{n-1}) = q$ , and furthermore, we have  $0 \leq x_n \leq b_n - 1$ . What makes that

$X_n = (X_{n-1}, x_n)$  is an element of  $B^{[n]}$ , and we indeed have  $\rho_n(X_n) = x_n + b_n \cdot \rho_{n-1}(X_{n-1}) = r + b_n \cdot q = a$ . CQFD.

The following procedures written in pseudo Pascal allow calculating the function **rank**:  $B^{[n]} \rightarrow N$  as well as its inverse **write**:  $N \rightarrow B^{[n]}$ . Knowing  $n, b_1 \dots b_n$ , and  $X_n$  de  $B^{[n]}$ , the first one calculates  $\rho_n(X_n)$ . The second, for every integer  $a$  such as  $0 \leq a \leq b^{[n]} - 1$ , determines the element  $X_n$  de  $B^{[n]}$  such as  $\rho_n(X_n) = a$ .

Function **RANK** ( $n, b_1 \dots b_n$ : integer;  $X_n$ : 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 **WRITE** ( $n, b_1 \dots b_n$ : 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 1:** we have called the inverse function of **rank write**, because it gives the writing of the integer  $a$  in the multiple or heterogeneous base  $(b_1, b_2 \dots b_n)$ . This writing is  $X_n = (x_1, x_2 \dots x_n)$ .

Recursively, every  $X_n$  of  $B^{[n]}$  can be seen as the writing in base  $(b_1 \dots b_n)$  of the integer  $a$  who is equal to  $\rho_n(X_n)$ .

**Remark 2:** the function rank was defined in a recursive way. We can need an explicit formulation. We see that

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

$$\rho_n(X_n) = \sum_{i=1}^n x_i \cdot b_{i+1} \dots b_n = \sum_{i=1}^n x_i \cdot \left( \prod_{j=i+1}^n b_j \right).$$

## 6.2: Isomorphism of ordered sets

**Property 6.2:**  $\rho_n$  is an isomorphism between  $B^{[n]}$  provided with lexicographical order  $\leq$ , and  $[0, b^{[n]} - 1]$  provided with the order  $\leq$  of the nature 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 show that it is true for  $n$ . We have  $X_n = (X_{n-1}, x_n)$ , and  $Y_n = (Y_{n-1}, y_n)$ . Let us suppose that  $X_n \leq Y_n$ . Let us show that we have  $\rho_n(X_n) \leq \rho_n(Y_n)$ . Indeed we have  $X_n \leq Y_n$  if  $(X_{n-1} < Y_{n-1})$ , or  $(X_{n-1} = Y_{n-1} \text{ and } x_n < y_n)$ .

- in the second case, we have  $\rho_n(X_n) = x_n + b_n \cdot \rho_{n-1}(X_{n-1}) < y_n + \rho_{n-1}(X_{n-1}) = \rho_n(Y_n)$ .

- In the first case, by I.H, we have  $\rho_{n-1}(X_{n-1}) < \rho_{n-1}(Y_{n-1})$ , and therefore  $\rho_n(Y_n) - \rho_n(X_n) = b_n \cdot (\rho_{n-1}(Y_{n-1}) - \rho_{n-1}(X_{n-1})) + y_n - x_n \geq b_n \cdot 1 + y_n - x_n$ . But we know that  $0 \leq x_n, y_n \leq b_n - 1$ . And thus  $\rho_n(Y_n) - \rho_n(X_n) > 0$ . So  $X_n < Y_n$ .  $\rho_n(X_n) < \rho_n(Y_n)$ . Recursively, let  $X_n$  and  $Y_n$  of  $B^{[n]}$  such as  $\rho_n(X_n) \leq \rho_n(Y_n)$ . Let us show that  $X_n \leq Y_n$ .

- If we had  $Y_{n-1} < X_{n-1}$ , we would have, by I.H,  $\rho_{n-1}(Y_{n-1}) < \rho_{n-1}(X_{n-1})$ .

- Then we would have  $\rho_n(X_n) - \rho_n(Y_n) = b_n \cdot (\rho_{n-1}(X_{n-1}) - \rho_{n-1}(Y_{n-1})) + x_n - y_n \geq b_n \cdot 1 + x_n - y_n > 0$ . In contradiction with the hypothesis.

- If we had  $X_{n-1} = Y_{n-1}$  and  $y_n < x_n$ , we would still have  $\rho_n(X_n) - \rho_n(Y_n) > 0$ , in contradiction with the hypothesis CQFD.

## 6.3: Addition of two numbers written in a multiple base

Let be the multiple base  $(b_1, b_2 \dots b_n)$  and two natural integers  $x, y$  such as  $0 \leq x, y \leq b^{[n]} - 1$ . The respective writings of these two numbers in base  $(b_1, b_2 \dots b_n)$  are  $X_n$  and  $Y_n$ . What is the writing 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  $y$ , can we calculate the writing  $Z_n$  of  $z$ , without having to calculate  $z = x + y$ ?

The answer is naturally yes! It consists in making the addition of  $X_n$  and  $Y_n$  in the base. The following procedure makes this operation. It uses an auxiliary sequence of nature integers  $r_0, r_1 \dots r_n$  which are going to play the role of the "carries".

Procedure **ADDITION** ( $n, b_1, \dots, b_n$ : 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;

( $r$  play the role of the final carry)

The addition of  $X_n$  and  $Y_n$  in base  $(b_1 \dots b_n)$  will be noted  $X_n \oplus Y_n$ .

Let us show that the procedure **ADDITION** returns well the expected result.

- For  $i = n$ , then for  $i = n-1 \dots i = 1$ , we add  $x_i$  and  $y_i$  and the carry  $r_i$ . Because for every  $i$  we have  $0 \leq x_i, y_i \leq b_i - 1$ , and  $r_i < 2$ , we see that  $0 \leq u = x_i + y_i + r_i \leq 2b_i - 1$ . If  $0 \leq u \leq b_i - 1$ , we take

- $z_i = u$ , and  $r_{i-1} = 0$ ; and else  $z_i = u - b_i$ , and  $r_{i-1} = 1$ . What makes that for every  $i$  of  $1 \dots n$  we have
- $0 \leq z_i \leq b_i - 1$ , and  $0 \leq r_i \leq 1$ . As for the final carry  $r = r_0$ , it is also equal to 0 or 1.
- Because for every  $i$  of  $\{1 \dots n\}$  we have  $0 \leq z_i \leq b_i - 1$ ,  $Z = (z_1 \dots z_n)$  belongs to  $B^{[n]}$ .
- It is  $Z$ , which is produced by the procedure of addition, which we put equal to  $X_n \oplus Y_n$ .

Now we shall establish the following property:

**Property 6.3:** whatever are the elements  $X_n$  and  $Y_n$  of  $B^{[n]}$ , we have  $\rho_n(X_n \oplus Y_n) = \rho_n(X_n) + \rho_n(Y_n) - r_0 \cdot b^{[n]}$ , with  $r_0 = 0$  or 1.

**Proof:** at every step  $i$  of the procedure **ADDITION**, we see that

If  $u < b_i$ , then  $z_i = u$  and  $r_{i-1} = 0$ , and else  $z_i = u - b_i$ , and  $r_{i-1} = 1$ . So, that for every  $i$ , we can write that  $z_i = u - b_i \cdot r_{i-1}$ ,  $z_i = x_i + y_i + r_i - b_i \cdot r_{i-1}$ .

$$\begin{aligned} \text{We thus have } \rho_n(Z_n) &= \sum_{i=1}^n z_i \cdot b_{i+1} \dots b_n = \sum_{i=1}^n (x_i + y_i + r_i - b_i \cdot r_{i-1}) \cdot b_{i+1} \dots b_n = \\ &= \sum_{i=1}^n x_i \cdot b_{i+1} \dots b_n + \sum_{i=1}^n y_i \cdot b_{i+1} \dots b_n + \sum_{i=1}^n r_i \cdot b_{i+1} \dots b_n - \sum_{i=1}^n r_{i-1} \cdot b_i \cdot b_{i+1} \dots b_n \\ &= \rho_n(X_n) + \rho_n(Y_n) + r_n + \left( \sum_{i=1}^{n-1} r_i \cdot b_{i+1} \dots b_n \right) - r_0 \cdot b_1 \dots b_n - \left( \sum_{i=2}^n r_{i-1} \cdot b_i \cdot b_{i+1} \dots b_n \right). \end{aligned}$$

But, we notice that the last two summations are equal. And we know that  $r_n = 0$ . What proves the property.

## 7. 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  $c_1, c_2 \dots c_p$ , ( $c_k$  = number of closed that the processor  $k$  can build in a given lapse of time), and that  $c_1 + c_2 \dots + c_p \geq b^{[n]} = b_1 \cdot b_2 \dots b_n$ . We can build a sequence of integers  $a_0, a_1 \dots a_l$ , 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;

$a_l = b^{[n]} - 1$ .

We obtain in this way a strictly increasing sequence  $(a_0, a_1 \dots a_l)$  such as  $a_0 = 0$ , and  $a_l = b^{[n]} - 1$ . And furthermore for  $k = 1 \dots l$ , we have  $c_k = a_k - a_{k-1}$ .

Then, for  $k = 0, 1 \dots l$ , we build the elements  $u_k$  of  $B^{[n]}$ , by calculating **WRITE**  $(n, b_1 \dots b_n, a_k, u_k)$ , which gives the writing  $u_k$  of  $a_k$  in base  $(b_1 \dots b_n)$ .

This sequence  $(u_0, u_1 \dots u_l)$  constitutes a good partition of  $B^{[n]}$ , since it is lexicographically strictly increasing one, with  $u_0 = 0_F, u_l = 1_F$ , and since it distributes the work by taking into account the capacity of processors.

However, this way, although mathematically correct, is difficult to implement. Indeed, when  $n$  is big, even for  $n = 30$ , the number  $b^{[n]} = b_1 \cdot b_2 \dots b_n$  is equal at least to  $2^n$ , and it can reach very big values, which are hard to calculate with the current compilers. And, therefore, the procedure **WRITE**  $(n, b_1 \dots b_n, a_k, u_k)$  is difficult to execute with big values of the parameter  $a_k$ .

To overcome this difficulty we can use additions in base  $(b_1 \dots b_n)$  as in the following case of equal partition:

Let us suppose that all the processors have the same capacity  $c$ . We determine the smallest integer  $p$  equal or bigger than  $b^{[n]} / c$ .

We determine the writing  $uc$  of  $c$  in base  $(b_1 \dots b_n)$ , by means of **WRITE**  $(n, b_1 \dots b_n, c, uc)$ . Then we build the sequence of elements  $u_k$  of  $B^{[n]}$  by the mean of the following operations:

$k := 0; u[k] := 0_F; r := 0;$

while  $(r = 0)$  do

begin

$k := k + 1;$

**ADDITION**  $(n, b_1, \dots, b_n, u[k-1], uc, u[k], r);$

end;

$p := k; u[p] := 1_F;$

This procedure returns  $u_0 = 0_F; u_1 = u_0 \oplus uc = uc, u_2 = u_1 \oplus uc \dots$  until the carry  $r$  becomes equal to 1.

While  $r = 0$ , by the property 5.3, we have  $\rho_n(X_n \oplus Y_n) = \rho_n(X_n) + \rho_n(Y_n)$ , and thus the addition of writings in base  $(b_1 \dots b_n)$  is equivalent to the addition of the corresponding numbers.

By this way we built the sequence  $(u_0, u_1 \dots u_p)$  of the partition, without computing  $(a_0, a_1 \dots a_p)$  ranks  $a_k = \rho_n(u_k)$ . We stop when  $r$  takes the value 1. One then take  $p = k$ , and we set  $u_p = I_F$ .

## 8. Conclusion:

We have just proposed a generalization of the Ganter algorithm, as well as a distributed version of this algorithm.

- On one hand this generalization allows to determine the Galois lattices associated to rather general contexts, without needing to re-code the data in binary values.
- On the other hand, when one analyze the relationship between product-order and lexicographical order on the Cartesian product  $B^{[n]} = B_1 \times B_2 \times \dots \times B_n$ , with  $B_j = \{0, b_j - 1\}$ , for  $j = 1 \dots n$ , this generalization may seem to be natural. Moreover the idea of seeing the elements of  $B^{[n]}$  as writings of integers in base  $(b_1 \dots b_n)$  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 big contexts. We just need to have rather powerful computers among which we shall distribute, as well as possible the work of construction of the closed itemsets.

## 9. References:

- [1] 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, September 2003.
- [2] Ganter, B: 1984, two basic algorithms in conceptual analysis. Technical report, Darmstadt University.
- [3] G. Lambert, R. Emilion, G. Lévy : Algorithmes pour construire les treillis de Galois généraux. To appear.