# Some Links Between Formal Concept Analysis and Graph Mining

Michel Liquière

# 1 Presentation

This chapter presents a formal model to learning from examples represented by labelled graphs. This formal model is based upon lattice theory and in particular Galois lattices. We widen the domain of formal concept analysis, by the use of the Galois lattices model with structural descriptions of examples and concepts. The operational implementation of our model, called "Graal" (for GRAph And Learning), constructs a Galois lattice for any description language provided that the operations of comparison and generalization are determined for that language. These operations exist in the case of labelled graphs satisfying a partial order relation (homomorphism).
This chapter is concerned as well with the known problems regarding propositionalization (i.e. the transformation of a structural description into a propositional description). Using classical lattice results, we have a formal model for the transformation of a structural machine learning problem into a propositional one.

# 2 Basic Concepts and Notation

## 2.1 Labeled Graphs

Let L denote set of *labels*. A labeled digraph (for labeled directed graph) G=(V,E,$\alpha$) is a 3-tuple where V is the finite set of *vertices*, E $\subseteq$ V$\times$V is the set of *edges*, $\alpha$ : V $\rightarrow$ L is a function assigning labels to each vertex. Edge $(v, v') \in$ E originates at node $v \in$ V and terminates at node $v' \in$ V. For a vertex v $\in$ V, we note $N^+(v) = \{v'|(v, v') \in V\}$and $N^-(v) = \{v'|(v', v) \in V\}$, $N(v) = N^+(v) \cup N^-(v)$.
For two labeled graphs $G_1$=($V_1, E_1, \alpha_1$), $G_2$=($V_2, E_2, \alpha_2$). A *homomorphism* h is a mapping h :$V_1 \rightarrow V_2$ for which $\forall v \in V_1$, $\alpha_1(v)$=$\alpha_2$(h($v$)) and $\forall$ $(v_1, v_2) \in E_1 \Rightarrow$ (h($v_1$),h($v_2$)) $\in E_2$ . A *subgraph isomorphism* is a homomorphism with an injective mapping (see figure 1). An *isomorphism* is a homomorphism with a bijective mapping.

## 2.2 Order

A *pre-order* on a set X is a binary relation $\leq$ on X which is reflexive and transitive. If x $\leq$ y and y $\leq$ x then we shall write x $\cong$ y and say that x and y are equivalent elements. A *partial order* (or poset) on a set X is an anty-symmetric preorder. If x,y $\in$ X, where X is a poset, then we shall write x $<$ y to mean that x $\leq$ y and x $\neq$ y. If x $\leq$ y and y $\leq$ x then x=y (isomorphic element).
Given a pre-order on X then the set of equivalence classes X/$\cong$ (notation [x]) can be given a partial ordering by setting [x] $\leq$ [y] $\Leftrightarrow$ x $\leq$ y $\forall$ x,y $\in$ X (quotient space).
A *lattice* is a partial order $(L, \leq, \vee, \wedge)$ where every pair of elements (x,y) have a meet (also named inf for infimum) x $\wedge$ y and a join (also named sup for supremum) x $\vee$ y (Birkhoof, 1967) and satisfying the following properties : $\forall$ x,y,z $\in L$,

| | | |
|---|---|---|
| x $\wedge$ (y $\wedge$ z) = (x $\wedge$ y) $\wedge$ z | x $\vee$ (y $\vee$ z) = (x $\vee$ y) $\vee$ z | (associativity) |
| x $\wedge$ y = y $\wedge$ x | x $\vee$ y = y $\vee$ x | (commutativity) |
| x $\wedge$ x = x | x $\vee$ x = x | (idempotence) |
| x $\wedge$ (y $\vee$ z) = x | x $\vee$ (y $\wedge$ z) = x | (absortion). |

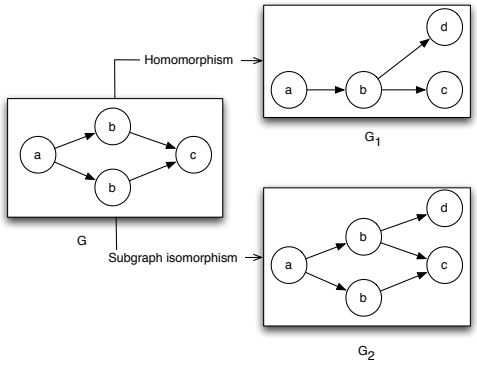


**FIG. 1.** Homomorphism and subgraph isomorphism

## 3 Formal concept analysis

"Formal concept analysis (FCA) has been introduced by Wille [37] and applied in many quite different realms like psychology, sociology, anthropology ..."[15]. An important contribution of FCA is a mathematical definition of a concept. From a philosophical point of view, a concept is a unit of thought consisting of two parts, the extension and the intension. The extension covers all objects belonging to the concept and the intension comprises all attributes valid for each of those objects. Hence objects and attributes play a prominent role together with several relations like e.g.the incidence relation, the hierarchical "subconcept-superconcept" relation between concepts and the implication between attributes (or objects). We know [37] that, from a *context* ($\mathcal{E}$,$\mathcal{A}$,$\mathcal{R}$) where $\mathcal{E}$ is a set of objects (examples), $\mathcal{A}$ a set of properties (attributes) and $\mathcal{R}$ a binary relation between $\mathcal{E}$ and $\mathcal{A}$, we can construct a minimal ordered set named a concept lattice [37], or Galois lattice [14].
This gives the following formal definitions : With e $\in \mathcal{E}$, a $\in \mathcal{A}$, e $\mathcal{R}$ a is asserting that "the object e verifies the attribute a" and "the attribute a is true for the object e".

In a context ($\mathcal{E}$,$\mathcal{A}$,$\mathcal{R}$), each *concept* (E,A) with E $\subseteq \mathcal{E}$ and A $\subseteq \mathcal{A}$. follows the properties :
• E contains each of the objects $e \in \mathcal{E}$ verifying all the attributes in A,
With A $\subseteq \mathcal{A}$, e(A) = {e $\in \mathcal{E}$ | e $\mathcal{R}$ a, $\forall$ a $\in$ A} represents the *extension part* of A.
• A contains each of the attributes a $\in \mathcal{A}$ true for the objects in E.
With E $\subseteq \mathcal{E}$, d(E)={a $\in \mathcal{A}$ | e $\mathcal{R}$ a, $\forall$ e $\in$ E} represents the *intention part* of E.
• a pair (E,A) with E$\subseteq\mathcal{E}$ and A$\subseteq\mathcal{A}$ , is a *concept* iff d(E)=A and e(A)=E.
For a context ($\mathcal{E}$,$\mathcal{A}$,$\mathcal{R}$), the structure L($\mathcal{E}$,$\mathcal{A}$,$\mathcal{R}$)={(E,A) | (E,A) is a concept of ($\mathcal{E}$,$\mathcal{A}$,$\mathcal{R}$) } ordered by $(E_1, A_1) \geq (E_2, A_2) \Leftrightarrow A_1 \subseteq A_2$ (formally equivalent to $E_2 \subseteq E_1$) is a concept lattice [37]. This classical order relation between concepts, is the relation "to be more general then" used in the machine learning community.

For a context ($\mathcal{E}$,$\mathcal{A}$,$\mathcal{R}$) and two concepts $(E_1, A_1), (E_2, A_2)$, the operations $\wedge$ and $\vee$ are :
$(E_1, A_1) \vee (E_2, A_2) = (E, A)$ with $E$=e($A_1 \cap A_2$) and $A$= $A_1 \cap A_2$
$(E_1, A_1) \wedge (E_2, A_2) = (E, A)$ with $E$=$E_1 \cap E_2$ and $A$= d($E_1 \cap E_2$).
The $\wedge$ operation (down) is defined from the intersection of the extensions while the $\vee$ operation (going up in the lattice) corresponds to the intersection of the intensions (see figure 2) Using machine learning terminology, the construction of the Galois lattice

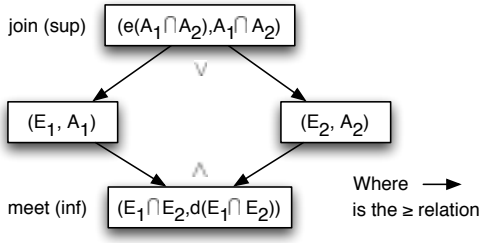


**FIG. 2.** Classical Galois lattice operations

using the $\vee$ operation is a *generalization search*. From two concepts the $\vee$ operation builds a more general concept. If we base our search on the $\wedge$ operation we have a *specialization search*.

### 3.1 Galois lattice construction from a binary relation

There are many algorithms for the construction of the Galois lattice for a context ($\mathcal{E}$,$\mathcal{A}$,$\mathcal{R}$) [15]. In this paper we present the algorithm [31] because it is simple, it uses general lattice operations and it is incremental (online algorithm) : it computes the lattice at step n+1 from the lattice at step n and the description of a new example (at the first step we begin with an empty lattice). In the next section we generalize this algorithm for a general description of the examples [1].

[1] In fact, for this purpose, we may used directly any Galois lattice construction algorithms which use general operations $\vee$ between descriptions. This is the case for [31], [10], [17], [32] but not for [7].

**Procedure:**maximal

**Data**: A couple (E,A)
**Data**: A list $L_n$ of concepts
**Result**: A boolean
**if** *(there is a* $(E_j, A_j) \in L_n$ *with* $A_j = A$*)* **then**
  **return** false;
**else**
  **return** true;
**end**

**Procedure:**AddExample

**Data**: $C_n$ :$(\mathcal{E}_n, \mathcal{A}_n, \mathcal{R}_n)$
**Data**: A new Example : a couple ({e},A), with e an identifier of an example, and A $\subseteq \mathcal{A}_n$ the description of the example e.
**Data**: A list $L_n$ of concepts
**Result**: A list $L_{n+1}$ of concepts
$L_{n+1}$=$L_n$ ;
**for** *each concept* $(E_i, A_i) \in L_n$ **do**
  **if** *(*$A_i \subseteq A$*)* **then**
    Replace $(E_i, A_i)$ in $L_n$ by ($E_i$∪{e},$A_i$);
  **else**
    // $\vee$ operation;
    compute (E′,A′)=(e($A_i \cap A$),($A_i \cap A$));
    **if** *maximal((*$E'$ *,*$A'$*),*$L_{n+1}$*)* **then**
      add (E′,A′) to $L_{n+1}$ ;
    **end**
  **end**
**end**
**if** *maximal(({e},A),*$L_{n+1}$*)* **then**
  add ({e},A) to $L_{n+1}$ ;
**end**
**return** $L_{n+1}$;

**Algorithm:** Norris's Algorithm :

**Data**: A context $(\mathcal{E}, \mathcal{A}, \mathcal{R})$
**Result**: The list L of all concepts in $(\mathcal{E}, \mathcal{A}, \mathcal{R})$
L = $\emptyset$;
**for** *each example e* $\in \mathcal{E}$ **do**
  L= AddExample(({e},d(e)),L);
**end**
**return** $L$;

**Algorithm 1**: A concept Lattice construction algorithm

Let's consider the relation $\mathcal{R}$ between $\mathcal{E}$={$e_0$,$e_1$,$e_2$,$e_3$,$e_4$} and $\mathcal{A}$={$a_0$,$a_1$,$a_2$,$a_3$,$a_4$,$a_5$,$a_6$} defined by the table in figure 3.

This binary relation gives the Galois lattice of figure 4. The presentation of Norris's algorithm uses the $\vee$ operation. In this case, it is a generalization algorithm : we begin

$$\begin{bmatrix} & a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ e_0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ e_1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ e_2 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ e_3 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ e_4 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

**FIG. 3.** Relation $\mathcal{R}$

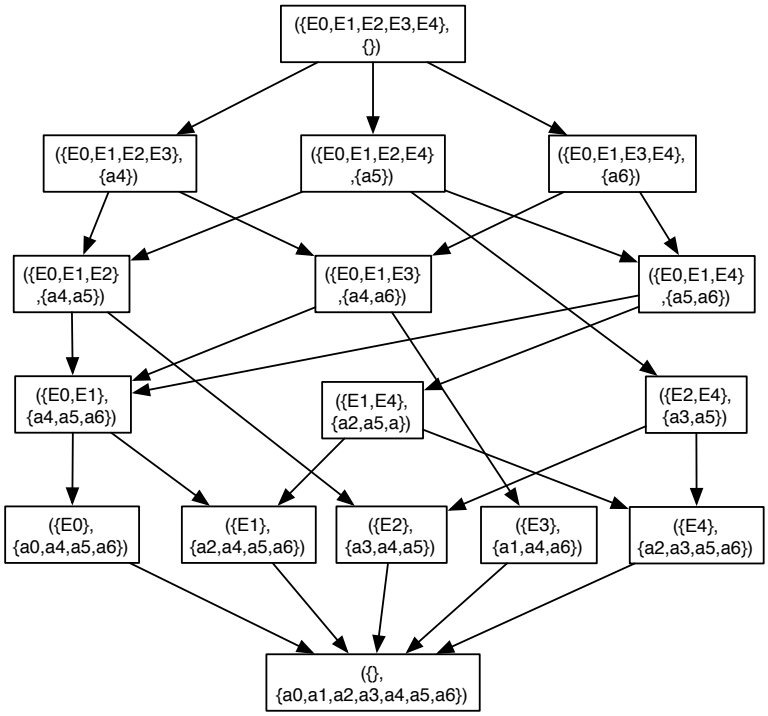


**FIG. 4.** The Galois lattice of $\mathcal{R}$, on the form of its Hasse diagram.

from some specific descriptions and construct some less specific description. Inversely, if we want to construct the lattice from the general descriptions to specific descriptions we have to replace the $\mathcal{R}$ relation by the inverse $\mathcal{R}^{-1}$ relation , and we have to give to the algorithm the couples ({a},E) where a $\in \mathcal{A}$, and E $\subseteq \mathcal{E}$. This is named "the duality principle for concept lattice" [15].

## 4 Extension space and description space give concept space

In this paragraph, we give another formalization of a Galois lattice based on a Galois connection between a lattice $\mathcal{X}$ and a lattice $\mathcal{D}$. The lattice $\mathcal{X}$ is defined from a set of examples $\mathcal{E}$, with $\mathcal{X}$=$\prod(\mathcal{E})$, the power set of the finite set $\mathcal{E}$. We name this lattice : *extension lattice*. The lattice $\mathcal{D}$ named *description semi-lattice* comes from a *description language* $\mathcal{L}$ which is a set of well formed expressions with :
$\geq_{\mathcal{L}}$ : a partial order between the expressions in $\mathcal{L}$. We note $=_{\mathcal{L}}$ the equality between to expression of $\mathcal{L}$. For example for graphs this equality can be an isomorphism relation.
$\otimes_{\mathcal{L}} : \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{L}$ a product operator with for $D_1 \in \mathcal{L}$ and $D_2 \in \mathcal{L}$, D=$D_1 \otimes_{\mathcal{L}} D_2$ of $\mathcal{L}$ verify : D $\geq_{\mathcal{L}} D_1$, D $\geq_{\mathcal{L}} D_2$ and $\forall$ D' | (D' $\geq_{\mathcal{L}} D_1$) and (D' $\geq_{\mathcal{L}} D_2$) $\Rightarrow$ D' $\geq_{\mathcal{L}}$ D.
This is a classical least general generalization operator [33].
Remark : this definition is also valid with a pre-order.

The relation between these two lattices is given by the two mappings $d : \mathcal{X} \rightarrow \mathcal{D}$ and $e : \mathcal{D} \rightarrow \mathcal{X}$ (see figure 5).

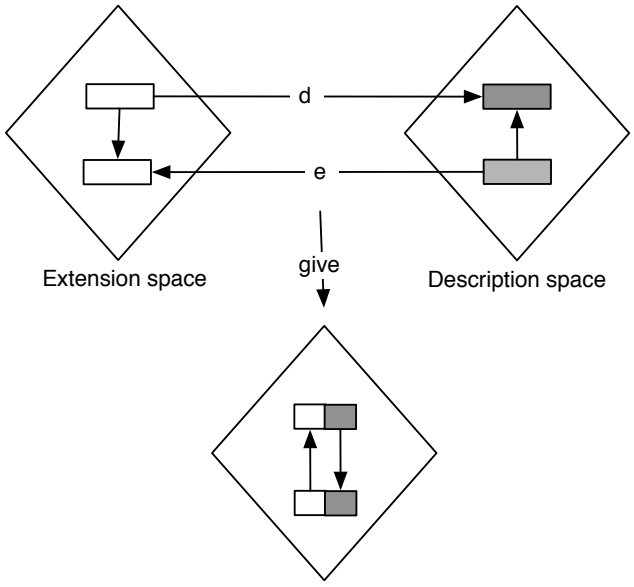


**FIG. 5.** Relations between the two spaces

We now generalize the notion of context in FCA into the notion of structural context [25] or general context [9] .
A *structural context* C is a triplet $(\mathcal{E}, \mathcal{L}, d)$ where $\mathcal{E}$ is a set of examples, $\mathcal{L}$ is a des-

cription language and d :$\prod(\mathcal{E})^2 \rightarrow \mathcal{L}$ a mapping. This mapping gives also the intension (also named description) d({e}) for an example e $\in \mathcal{E}$. We can easily transform each context $(\mathcal{E}, \mathcal{A}, \mathcal{R})$ into a structural context $(\mathcal{E}, \mathcal{L}, d)$, where $\mathcal{L}$ is the power set of $\mathcal{A}$.
For a structural context $(\mathcal{E}, \mathcal{L}, d)$ a concept is a pair (E,D) with :
• D $\in \mathcal{L}$ | D= $\bigotimes_{\mathcal{L}}$ d(e) for e $\in$ E.
• E $\in \prod(\mathcal{E})$ | E=$\{e_i \in \mathcal{E}$ | D $\geq_{\mathcal{L}}$ d($e_i$)$\}$.
For two concepts $(E_1, D_1)$, $(E_2, D_2)$ we have the partial order relation : $(E_1, D_1) \geq_c (E_2, D_2) \Leftrightarrow E_1 \supseteq E_2$ and $D_1 \geq_{\mathcal{L}} D_2$.

**Proposition 1.** *For a structural context $(\mathcal{E}, \mathcal{L}, d)$ the set of concepts, ordered by the relation $\geq_c$, is a Galois lattice. We note this lattice $\mathcal{G}(\mathcal{E}, \mathcal{L}, d)$.[25]*

From the previous definitions, we can define the lattice operations $\wedge$ (meet) and $\vee$ (join).
$(E_1, D_1) \vee (E_2, D_2)$ = (E,D) with D=$(D_1 \otimes_{\mathcal{L}} D_2)$ and E=$\{e \in \mathcal{E} \mid d(e) \leq_{\mathcal{L}} D\}$.
$(E_1, D_1) \wedge (E_2, D_2)$ = (E,D) with E=$E_1 \bigcap E_2$ and D=$\otimes_{\mathcal{L}}$ d($e_i$), for $e_i \in$ E. We denote this structural Galois lattice $\mathcal{G}(\mathcal{E}, \mathcal{L}, d)$ . With this result we can generalize the Norris's algorithm and use it a for the generation of a Galois lattice from a structural context :

**Procedure:**maximal
**Data**: A couple (E,D)
**Data**: A list $L_n$ of concepts
**Result**: A boolean
**if** *(there is a $(E_j, D_j) \in L_n$ with $D_j =_{\mathcal{L}} D$)* **then**
  **return** false;
**else**
  **return** true;
**end**

[2] powerset de $\mathcal{E}$

**Procedure:**AddExample

**Data**: $C_n$ :$(\mathcal{E}, \mathcal{L}, d)$
**Data**: A new Example : a couple ({e},D), with e an identifier of an example, and D $\in \mathcal{L}$ a description of the example e
**Data**: A list $L_n$ of concepts
**Result**: A list $L_{n+1}$ of concepts
$L_{n+1}$=$L_n$ ;
**for** *each concept* $(E_i, D_i) \in L_n$ **do**
  **if** *(*$D_i \geq_{\mathcal{L}} D$*)* **then**
    Replace $(E_i, D_i)$ in $L_n$ by ($E_i$∪{e},$D_i$);
  **else**
    // $\vee$ operation;
    compute (E′,D′)=(e($D_i \otimes_{\mathcal{L}} D$),($D_i \otimes_{\mathcal{L}} D$));
    **if** *maximal((*$E'$ *,*$D'$*),*$L_{n+1}$*)* **then**
      add (E′,D′) to $L_{n+1}$ ;
    **end**
  **end**
**end**
**if** *maximal(({e},D),*$L_{n+1}$*)* **then**
  add ({e},D) to $L_{n+1}$ ;
**end**
**return** $L_{n+1}$;

**Algorithm:**Norris's Algorithm for structural language :

**Data**: A structural context $(\mathcal{E}, \mathcal{L}, d)$
**Result**: The list L of all concepts in $(\mathcal{E}, \mathcal{L}, d)$
L = ∅;
**for** *each example e* $\in \mathcal{E}$ **do**
  L= AddExample(({e},d(e)),L);
**end**
**return** $L$;

**Algorithm 2**: A structural concept Lattice construction algorithm

This algorithm uses the operators $\geq_{\mathcal{L}}$, $=_{\mathcal{L}}$ [3] and $\otimes_{\mathcal{L}}$. This is a generalization method, if we want to use it on a specialization search, we need an initial set of graphs. This remark is further developed in the section *Graph mining and formal propositionalization*.

The complexity of the construction of a Galois lattice with this algorithm for a description language $\mathcal{L}$, is a function of :
1) the time and space complexity of the operations ($\geq_{\mathcal{L}}$,$\otimes_{\mathcal{L}}$),
2) the number of nodes in the lattice.
3) the algorithm used : for a description language $\mathcal{L}$ and a lattice L, the complexity of the AddExample function is O($|L|^2 \times P + |L| \times T$) where P represents the complexity of the $\geq_{\mathcal{L}}$ and T represents the complexity of the $\otimes_{\mathcal{L}}$ operation.
We will see in the next section an application of this formalization to a graph descrip-

[3] With a pre-order we can replace $=_{\mathcal{L}}$ by another equivalence relation.

tion of the examples and we will evaluate the complexity of the method in this case.

## 5 Graph description and Galois lattice

For a description language we want to use labeled digraphs. This representation is interesting because it is used in many applications [11], furthermore, conceptual graphs [35] are a specific case of labeled graphs.

Now we have to select the partial order we consider between labeled digraphs. A classical partial order relation is the subgraph isomorphism relation. The interpretation of such a relation between graphs is natural but there are two drawbacks : the complexity of the search of such a relation between two graphs and the definition of a product operator for this relation. There is another classical pre-order relation between graphs based on the homomorphism projection. This pre-order is also used in inductive logic programming (ILP) under the name $\theta$-subsumption [33] and in conceptual graph model with the term "projection"[35]. Now, our goal is to choose the operators $\geq_g$ and $\otimes_g$ for our labeled graphs.

### 5.1 $\geq_g$ for labeled graphs.

For two labeled digraphs $G_1$ and $G_2$, we note $G_1 \geq_g G_2 \Leftrightarrow$ there is a homomorphism [4] from $G_1$ *into* $G_2$.
For a set of labeled digraphs, the homomorphism relation is only a pre-order because the antisymmetric property is not fulfilled [40].
In order to use algorithm 2 , a partial order between elements of the description language is not necessary but it is better for the interpretation of the results. For the homomorphism relation we have the following equivalence relation so we can construct a quotient space with this relation.
Equivalence relation $\cong_g$ : For two labeled digraphs $G_1$ and $G_2$, we note $G_1 \cong_g G_2$ if both $G_1 \geq_g G_2$ and $G_2 \geq_g G_1$.
For this purpose, we use the class of core labelled graphs [40].

**Definition 1 (Core).** *A labeled graph G is called a core if it has no strict labeled subgraph G' $\subset$ G with G' $\cong$ G.*

**Proposition 2.** *For the equivalence relation defined above ($\cong_g$). An equivalence class of labelled graphs contains one and only one core labelled graph, which is the (unique) labeled graph with the smallest vertex number. [40]*

We can construct the core graph of a graph as proved by [29]. This operation is called reduction (notation R). and we have the following property :

**Proposition 3.** *If, for a class of labelled graphs, the homomorphism is polynomial, then the reduction operation R is polynomial [29]. We use R(G) to represent the core labeled graph of a labeled graph G.*

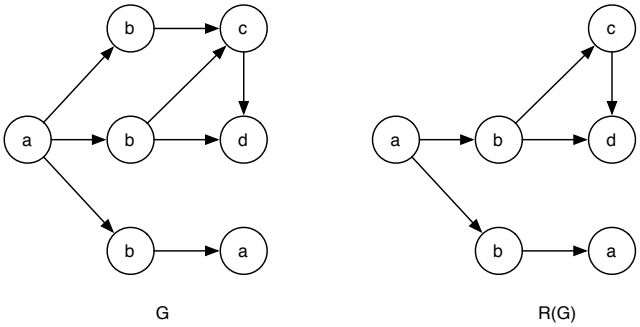


**FIG. 6.** A labeled graph G and its core graph R(G).

All labelled graph description of the example can be converted to an equivalent ($\cong$) core labelled graph, using the R operation.

**Proposition 4.** *The restriction of $\geq_g$ to the set of core labelled graphs is a lattice [40]*

For this lattice, the $\wedge$ operation is based on the disjoint sum $\oplus$ of two graphs [5], $G_1 \wedge G_2$= R($G_1 \oplus G_2$). [6] The $\vee$ operation is more complex and is defined in the following paragraph.

### 5.2 Homomorphism and product operator $\otimes_g$

For a description language $\mathcal{L}$ we need a product operator ($\otimes_{\mathcal{L}}$). There is a classical product operator for the homomorphism relation between labeled digraphs.

**Definition 2** ($\otimes_g$ **operation).** *For two labeled digraphs $G_1 = (V_1, E_1, \alpha_1)$, $G_2 = (V_2, E_2, \alpha_2)$ we define $G : G_1 \otimes_g G_2$ by G=(V,E,$\alpha$) with : V= $V_1 \times V_2$ with $\alpha_1(V_1) = \alpha_2(V_2)$ and $((x_1, x_2),(y_1, y_2)) \in E \Leftrightarrow (x_i, y_i) \in V_i$ (i=1,2).*

**Proposition 5.** *For the homomorphism relation, $G_1 \otimes_g G_2$ is the product of $G_1$ and $G_2$. proof : A proof of this proposition can be found in [5]*

Given the core labeled digraph $G_2$, $G_2$ the product is $\otimes_{gc}$ with $G_1 \otimes_{gc} G_2$= R($G_1 \otimes_g G_2$) and we have a partial order [25].

---

[4] defined in section 2.1

[5] We construct G :(V,E,$\alpha$ ) from $G_1 : (V_1, E_1, \alpha_1)$ $G_2 : (V_2, E_2, \alpha_2)$ with V=$V_1 \cup V_2$, E=$E_1 \cup E_2$

[6] We don't use this operation in the algorithm so the description space can be a sup semi-lattice.

### 5.3 Complexity

Using the generalization relation $\geq_g$, the product $\otimes_g$ and algorithm 2 we can build the Galois lattice for a set of examples described by labelled digraph. But the complexity of the operation $\geq_g$ is NP-hard for this kind of graphs.

**The size of the Galois lattice**

Given a context ($\mathcal{E}$,$\mathcal{A}$,$\mathcal{R}$), the number of nodes in the lattice is less than or equal to $2^{min(|\mathcal{E}|,|\mathcal{A}|)}$. This property comes from the relation between the two spaces (partition space and description space).
In the case of a general description of the example, the size of the Galois lattice is min($2^{|\mathcal{E}|}$, SD) (where SD is the size of the description space). In general, for a structural description of the example we have SD$\gg 2^{|\mathcal{E}|}$ (P-space problem [19])

**Complexity of the $\otimes_g$ operation**

For two labelled graphs G1=(V1,E1,$\alpha 1$) and G2=(V2,E2,$\alpha 2$), the time and space complexity of the product $\otimes_g$ is O($|V1| \times |V2| \times (|E_1| + |E_2|)$).
For a set of labelled digraphs $\Gamma$ the size of $\otimes_g G_i$ with $G_i \in \Gamma$ can be exponential even if we use the $\otimes_{gc}$ operator.
For core labeled graphs, the product $\otimes_{gc}$ is NP-hard (the reduction process R is NP-Hard).

**Complexity of the $\geq_g$ operation**

In the IncGen algorithm this operation is used several times. For general labeled digraphs the complexity of the homomorphism operation is NP-hard. However the homomorphism is polynomial for paths, trees and locally injective digraphs (see definition 3).

### 5.4 Locally injective digraph

We define a large class of labeled graphs, named LIG graphs, where the operations $\geq_{lig}$ and $\otimes_{lig}$ have a polynomial complexity.

**Definition 3 (Locally injective digraph (LIG digraph)).** *A labeled Digraph G :(V,E,$\alpha$) is locally injective $\Leftrightarrow$ for each vertex v $\in$ V, $\forall\, v_1, v_2 \in N^+(v)$, $v_1 \neq v_2 \Rightarrow \alpha(v_1) \neq \alpha(v_2)$ and $\forall\, v_1, v_2$ $v_1 \neq v_2 \Rightarrow \in N^-(v)$, $\alpha(v_1) \neq \alpha(v_2)$*

In figure 7 , $G_1$ is a LIG digraph, $G_2$ is not a LIG digraph because, for the node labeled with a c, there are two neighbors labeled with the same label.

**Proposition 6.** *The homomorphism operation is polynomial for locally injective digraphs.*

In fact, the subgraph isomorphism relation is also polynomial for this class of digraphs [25]

**Proposition 7.** *For two LIG digraphs $G_1$,$G_2$. G= $G_1 \otimes_g G_2$ is a LIG digraph.*

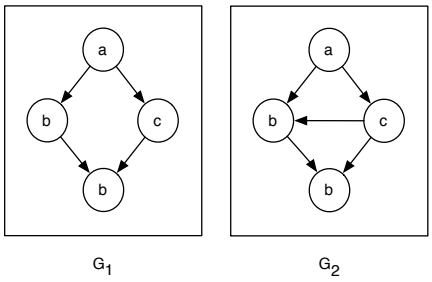


**FIG. 7.** Example of LIG digraphs and not LIG digraphs

These properties are interesting because for LIG we can use the operations $\geq_g$ and $\otimes_g$ with a polynomial complexity.

We can use the previous algorithm directly on a LIG digraph, but the product operator is redundant. In many cases we obtain a digraph where some part is equivalent to or less general than other parts. To avoid this drawback we can use the core GLI graph and the $\otimes_{gc}$ operator which is polynomial for GLI graphs. This kind of labeled graph is used in figure 8. In this figure the nodes with gray background are $\wedge$-irreducible elements (nodes with one and only one predecessor). We use this element in the section *Graph mining and formal propositionalization*.

### 5.5 Complexity for some description languages.

In this part we give some complexity results for the construction of a structural Galois lattice. The algorithm 2 can be used on any description language with a generalization operation. For example we can use it on Inductive Logic Programming using the classical generalization operator defined by Plotkin [33].

There are many description languages $\mathcal{L}$, we present here a small subset and we give the time and size complexity of the operations $\geq_{\mathcal{L}}, \otimes_{\mathcal{L}}$. In fact, for some language, the size of the description generalizing two descriptions increases just by a factor. However, several utilizations of the $\otimes_{\mathcal{L}}$ operation can produce a description exponential in size, so we give also this complexity.

For the classical subgraph isomorphism relation there is no product operator, for two graphs we can't have a unique generalization. In this case we have to extend the description from a graph to a set of graphs. Then we can define a partial order between the set of graphs and have a product operator between set of graphs. To do this, we consider each graph as the representation of the set of all the subgraphs of this graph (itself include). In this case the generalization of two graphs is the set constructed by the intersection of the two associated sets. We have the same methodology with strings if we want to use the inclusion as the partial order between set of strings . In this case we can use a classical algorithm for the research of all repeated substrings in a set of strings [12]. We will use this idea in the next paragraphs.

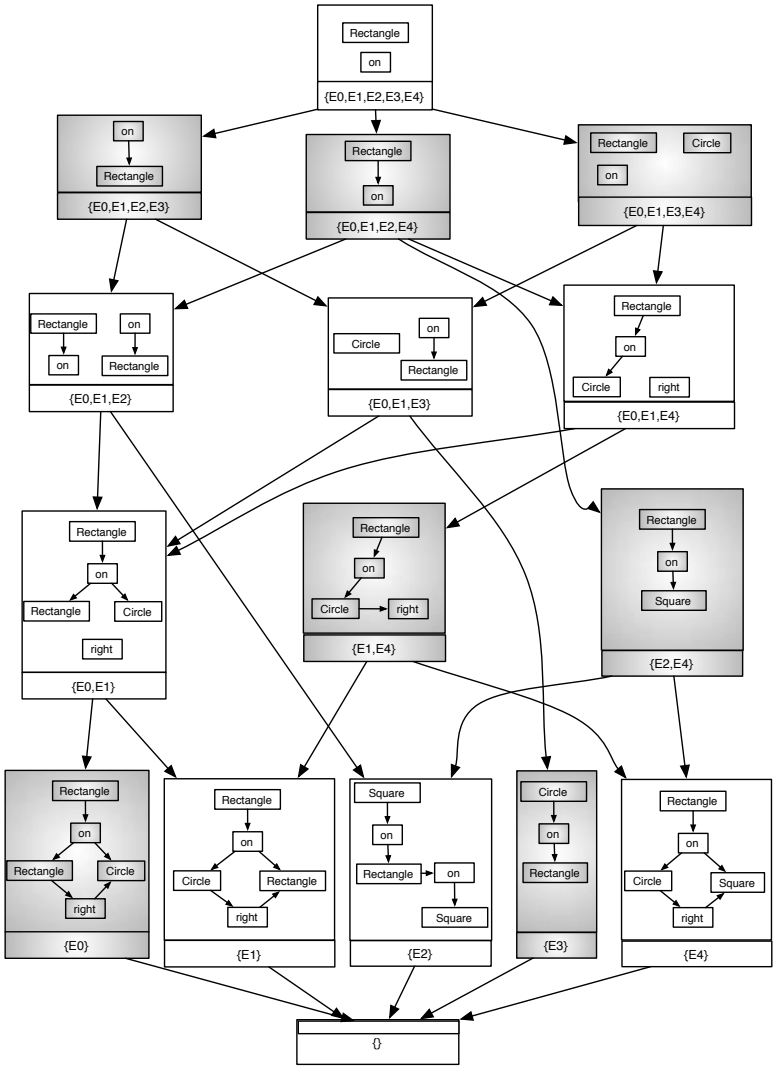


**FIG. 8.** The Galois lattice, on the form of its Hasse diagram, of $\mathcal{R}_{st}$

## 6 Graph mining and formal propositionalization

In the previous section we used a product operator, as a consequence, the construction of the lattice is a method by generalization (generalization method). We begin with specific graphs and we build generalizations of the graphs. In many practical application we need a specialization method. We begin with some general graphs and construct

| | $\geq$ | $O(\geq)$ | $\otimes$ | $O(\otimes)$ | $size$ |
|---|---|---|---|---|---|
| $Set$ | $\subseteq$ | $P$ | $\bigcap$ | $P$ | $P$ |
| $String$ | Lexical order | $P$ | maximal common prefix | $P$ | $P$ |
| $rooted\ Tree$ | rooted tree Inclusion | $P$ | maximal common prefix tree | $P$ | $P$ |
| $LIG\ Graph$ | homomorphism | $P$ | Graph product and Minimization | $P$ | ? |
| $automata$ | homomorphism | $P$ | automata product and Reduction | $P$ | $E$ |
| $Graph$ | homomorphism | $NP$ | Graph product and Reduction | $NP$ | $E$ |
| $Clause$ | $\Theta - subsumption$ | $NP$ | rlgg operator [33] | $NP$ | $E$ |

Where P means Polynomial, NP : NP-hard and E : Exponential

**FIG. 9.** Relation $\mathcal{R}$

some more specific graph using a specialization operator.
Using this top-down search, we can select concepts by the cardinality of its extension part. This defines a semi-lattice [28] named iceberg lattice [36].

For propositional description of the examples, and a context $(\mathcal{E}, \mathcal{A}, \mathcal{R})$ , $(\mathcal{A}, \mathcal{E}, \mathcal{R}^{-1})$ is also a context and we can use the classical concept lattice algorithm on this dual context. This is named "the duality principle for concept lattice" [15]. "In other words : if we exchange the roles of objects and attributes, we obtain the dual concept lattice". For a structural context, we haven't the duality principle directly however using some fundamental lattice properties we can manage this problem.

Many machine learning problems have a solution in propositional description of the examples, for example the Q-learning [34]. When we want to deal with a structural description, many new complexities appear (time and size complexity of the generalization and matching operations). For this problem, a natural idea is to convert the structural problem into a propositional one. This transformation doesn't change the complexity of the problem, but classical methods can be used with this new description.
For our problem, generalization of FCA and specialization search, this methodology seems really promising. To do this transformation we need a set SP of structural patterns (features). Then, we can use this set to recode each example with a boolean vector. Each boolean in this vector gives the information about the presence or absence of feature of SP in the description of the example. For a description language $\mathcal{L}$, the test of presence or absence uses a $\geq_{\mathcal{L}}$ operator. Such a transformation from a structural machine learning problem to a propositional problem is called propositionalization [21]. There is many recent works in this domain, [21], [3], [22]. These methods search the "best" features. Most of the authors define this problem as being the selection of a subset of m attributes from a set of n attributes. This give a theoretical complexity of $C_n^m$, that renders the model unusable in the general case. Then the propositionalization methods use heuristics for the selection of the attributes, like the stochastic selection [3] or syntactic selection [23].
Our purpose, has a great proximity, but proposes to consider the problem from another point of view. The idea is to find a propositional language, for a problem of classification, equivalent to a description language. This equivalence is not an expressiveness equivalence of the languages. This has already been studied in first order logic with the

universal relationship notion.
In this paper, using a fundamental theorem (theorem 1), we define a new equivalence between languages. This equivalence uses the structure of our classifications spaces. We consider that, for a set of examples, two description languages are equivalent if they give the same concept space.
To deal with this problem we have to define how two languages will be considered as equivalent for a set of examples. Then, it will be possible to search for a propositional language equivalent, in the specified senses, to the structural language.

### 6.1 Some properties of lattice and Galois lattice

The elements $\vee$-irreducible (resp.$\wedge$-irreducible) in a lattice are the elements with exactly on lower neighbour (resp. upper neighbour). We note J(L) (resp. M(L)) the set of all $\vee$-irreducible ($\wedge$-irreducible) elements of a lattice L.
For a lattice L, we can construct a binary relation B(L) between J(L) and M(L) with B($j_i$,$m_j$) = 1 $\Leftrightarrow$ $j_i \leq m_j$ (for $j_i \in$ J(L) and $m_j \in$ M(L)).

The $\wedge$-irreducible (resp.$\vee$-irreducible ) elements are the element which can't be derived from a $\wedge$ (resp. $\vee$) operation between elements of the lattice. So other element are not essential because we can reconstruct them from other elements.

**Proposition 8.** *In a finite lattice each element is the meet of $\wedge$-irreducible elements.*

For a context $(\mathcal{E}, \mathcal{A}, \mathcal{R})$ the set $\mathcal{A}$ (resp. $\mathcal{E}$) is a superset of the set of $\wedge$-irreducible (resp $\vee$-irreducible) elements.

**Theorem 1 (Lattice and Galois lattice).** *All lattice L is isomorph with the Galois lattice build from the binary relation B(L)=(J(L),M(L),$\leq$).*

This result is know since 1965 [4] and is also a fundamental theorem in FCA [15].
So from a given context we can remove all the reducible elements. The new context (name reduced context [15]) give a Galois lattice which is is isomorphic with L($\mathcal{E}, \mathcal{A}, \mathcal{R}$). This property is also true if we remove only a subset of the reducible elements and in particular, a subset of the reducible attributes. We develop this point of view in the next section and give a methodology for formal propositionalization.

### 6.2 Language equivalence for a set of examples

In real world, the choice of a good description language has been one of the most important task of human knowledge acquisition. In fact, the research of a simpler compact language, allowing to express the differences and resemblances between objects, is the foundation of many domain. For example, in the area of chess, a good player (computer or human) needs a set of concepts (like open column, center domination,...) to evaluate a position.

One of the simplest language is the propositional language. Therefore, we are interested in the problem to know if there exists, for a set of examples, a propositional

language equivalent to a given structural language. And more precisely what are the conditions for the apparition of this propositional language.
In the framework of our model, we obtain the following definition :

**Definition 4 (Context equivalence).** *Two structural contexts* $(\mathcal{E}, \mathcal{L}_1, d_1)$ *and* $(\mathcal{E}, \mathcal{L}_2, d_2)$ *are say equivalent iff* $\mathcal{G}(\mathcal{E}, \mathcal{L}_1, d_1) \equiv \mathcal{G}(\mathcal{E}, \mathcal{L}_2, d_2)$ *(isomorphism).*

So, for the same set of example, all classifications of the examples (extension part of the concept) obtained with the language $\mathcal{L}_1$ and with the language $\mathcal{L}_2$ are the same.

Using this definition and the theorem 1, we can now give a property on the existence of a propositional language equivalent to a structural description language for a set of example.

**Proposition 9.** *For a structural context* $(\mathcal{E}, \mathcal{L}, d)$ *there is a context (E,A,R) equivalent (definition 4).*

The property 8 and theorem 1 shows that each of all element in lattice can be characterized from the $\wedge$-irreducible elements. This is therefore also true in the case of structural Galois lattice. Thus the concepts of $(\mathcal{E}, \mathcal{L}, d)$ are in 1-1-correspondence with the concepts of $(\mathcal{E}, \mathcal{A}, \mathcal{R})$ where $\mathcal{A}$ is the set M(L) of $\wedge$-irreducible elements of L :$\mathcal{G}(\mathcal{E}, \mathcal{L}, d)$

### 6.3 Search for $\wedge$-irreductible elements

The property 9 gives a formal definition of propositionalization. Now the problem is the search for the set (or a subset for an approximation) of $\wedge$-irreducible elements. A theorical method can build the complete structural Galois lattice for a structural context, then we can easily find the $\wedge$-irreductible elements (all element with only one predecessor in the Hasse diagram). In figure 8, we show a distribution of the $\wedge$-irreducible elements in the lattice (nodes with gray background). With this method, we can find the set of irreductible elements, however, even for simple structural description languages $\mathcal{L}$, the complexity of the $\otimes_{\mathcal{L}}$ and $\geq_{\mathcal{L}}$ operations and the size of the lattice limit this method to a small set of examples.

In fact three others considerations induce the use of a specialization method :
1) Since all element in a complete lattice are the $\wedge$ of $\wedge$-irreducible element, the $\wedge$-irreducible elements are (statiscally) more "on the top" of the lattice.
2) we can limit the exploration to $\wedge$-irreducible elements seen on enough examples (classical support data mining heuristic).
3) we can use classical pattern mining for the construction,from general to specific, of set of patterns.

In the book [15] there is some important properties for the research of $\wedge$-irreducible (or $\vee$-irreducible elements) in the case of a propositional context.

**Proposition 10 (∧-irreducible).** *For a context $(\mathcal{E}, \mathcal{A}, \mathcal{R})$ where $\forall\ a_1, a_2\ _{(a_1 \neq a_2)}$ e($a_1$) ≠ e($a_2$), $a \in \mathcal{A}$ is a ∧-irreducible attribute $\iff$ $\exists\ e \in \mathcal{E}$ | ¬(e $\mathcal{R}$ a) and $\forall$ a'≠ a ∈ d(e(a)) (a closure) e $\mathcal{R}$ a.*

We have a dual property for ∨-irreducible elements.
For a ∧-irreducible $a$ we note *irre(a)* the not empty set, off all the examples verifying the last part of this property. We can see that a ∧-irreducible attribute $a$ is characterized by a set of examples (irre(a)). The example in irre(a) doesn't verified the attribute a but are true for all attribute b with e(a) ⊂ e(b). In fact the a attribute is fundamental for the separation of the example of irre(a) and the set of the others examples.

The previous property give can give a polynomial algorithm for the seek of ∧-irreducible element of a binary relation $\mathcal{R}$. But for a propositionalisation problem, the number of attributes, we have to test, can be exponential. Then the relation $\mathcal{R}$ can be really big. Furthermore, in some application, like reinforcement learning [34] the pattern occurs step after step. So a incremental algorithm is better for this case.

**Definition 5.** *A context $(\mathcal{E}, \mathcal{A}, \mathcal{R})$ is ∧-reduce iff all the attributes a ∈ $\mathcal{A}$ are ∧-irreducible.*

**Proposition 11.** *For a ∧-reduce context ($\mathcal{E}$,$\mathcal{A}_n$,$\mathcal{R}_n$). The context ($\mathcal{E}$,$\mathcal{A}_{n+1}$,$\mathcal{R}_{n+1}$), build by the addition of a new attribute $a_{n+1}$ to the context ($\mathcal{E}$,$\mathcal{A}_n$,$\mathcal{R}_n$) with $\mathcal{A}_{n+1}$=$\mathcal{A}_n$ ∪ $a_{n+1}$ and $\mathcal{R}_{n+1}$ is the new binary relation between $\mathcal{E}$ and $\mathcal{A}_{n+1}$ The context ($\mathcal{E}$,$\mathcal{A}_{n+1}$,$\mathcal{R}_{n+1}$) is ∧-reduce iff the three following properties are verified :*

*i)* $\nexists\ a_k \in \mathcal{A}_n$ | *e($a_k$)= e($a_{n+1}$)*

*ii) $a_{n+1}$ is a ∧-irreducible for the context ($\mathcal{E}$,$\mathcal{A}_{n+1}$,$\mathcal{R}_{n+1}$)*

*iii)* $\forall\ a_k \in \mathcal{A}_n$ | *e($a_k$)⊂e($a_{n+1}$) ⇒ irre($a_k$) $\bigcap$ e($a_{n+1}$) ≠ ∅*

If we use the property 11 we obtain the following algorithm :

**Procedure:** update
**Data**: $C_n$ :$(\mathcal{E}, \mathcal{A}_n, \mathcal{R}_n)$
**Data**: attribute a
**Result**: $C_n$ modified
F={ $a_k \in \mathcal{A}_n$ | e($a_k$) ⊆ e($a$)};
**for** $a_k$ *in F* **do**
  irre($a_k$)= irre($a_k$) ∩ e($a$) ;
  **if** *irre($a_k$)* = ∅ **then**
    $C_n$= $C_n$ - $a_k$;
  **end**
**end**
**return** $C_n$;

**Procedure:** is-Irreducible

**Data**: $C_n$ :$(\mathcal{E}, \mathcal{A}_n, \mathcal{R}_n)$
**Data**: attribute a
**Data**: E $\subseteq \mathcal{E}$
**Result**: True if a is $\wedge$-irreducible else False;
We compute also irre(a) if a is $\wedge$-irreducible
F={ $a_k \in A_n$ | e(a) $\subseteq$ e($a_k$)};
**if** *F* = $\emptyset$ **then**
  irre(a)=$\mathcal{E}$ - e(a)
**else**
  //Initialization of irre(a);
  irre(a)=e($a_j$) - e(a) // With $a_j$ one element in F ;
  F=F - $a_j$ ;
  // Work with $a_j$ is done **for** $a_k$ *in F* **do**
    **if** *e($a_k$)=e(a)* **then**
      **return** False;
    **else**
      irre(a)= irre(a) $\cap$ e($a_k$) ;
      **if** *irre(a)* = $\emptyset$ **then**
        **return** *False*;
      **end**
    **end**
  **end**
**end**
**return** *True*;

**Algorithm:** addFeature : (A propositionalisation Algorithm)

**Data**: A structural context $(\mathcal{E}, \mathcal{L}, d)$
**Data**: $C_n$ :$(\mathcal{E}, \mathcal{A}_n, \mathcal{R}_n)$
**Data**: feature a $\in \mathcal{L}$
**Result**: $C_{n+1}$ : $(\mathcal{E}, \mathcal{A}_{n+1}, \mathcal{R}_{n+1})$
;
E={$e_i \in \mathcal{E}$ with a$\geq_{\mathcal{L}}$ d($e_i$)};
**if** *is-Irreducible($C_n$,a,E)* **then**
  $C_{n+1}$=update($C_n$,a);
**else**
  $C_{n+1}$=$C_n$;
**end**
**return** $C_{n+1}$;

**Algorithm 3**: Formal Incremental Propositionalisation method

### 6.4 $\wedge$-irreducibles and pattern mining

The idea is to use the algorithm 3 on graph description with a little trick [16] : each graph is replace by the set (or a subset) if its subgraphs. For the research of these subgraphs we use a graph mining method. These kind of algorithms are optimized for a specialization search, they build the graphs at step n, from graphs of step n-1 (and step

n-2). This methods use different parameters and principally a support parameter which is, for a given pattern, the minimum number of example where this pattern is present. In our algorithm, we will use the pattern mining method as a feature generator. We name it PatternMiningMethod() and consider that the result of this method is a set of expressions (features, patterns) of the description language $\mathcal{L}$.

**Algorithm : Apropo** :
Input : a structural context ($\mathcal{E}$, $\mathcal{L}$,d)
s : support parameter
m : depth max
Output : A context ($\mathcal{E}$, $\mathcal{A}$, $\mathcal{R}$) (then a set of features $\mathcal{A}$)
n=0 ;
$C_0$=($\mathcal{E}$,$\emptyset$, $\mathcal{R}_0$)
while (n$\leq$ m)
{ F= PatternMiningMethod(($\mathcal{E}$, $\mathcal{L}$,d), s, n, .. and other specific parameters)
For all feature f in F
$C_{n+1}$=addFeature($\mathcal{E}$, $\mathcal{L}$,d),$C_n$,f)
n=n+1 ;
}

In the previous algorithm we can use any pattern mining method which use expression of $\mathcal{L}$), the description language use for the description of the examples. For the description of the example by graph there is now a large set of graph mining methods [18], [30] .... To reduce the number of features, we are tempted to replace the expressions with weaker even if that results in some loss of information. This transformation is formalized in the paper [16].
In 1989 we have proposed a machine learning method for the research of elementary paths present in a set of graphs [24]. This method is a level wise method which used the classical support criteria and with no limitation on the structure of the graph. Some more recent methods on the same problem are : [39] [30]. If we use this graph mining method in the algorithm 2 for the set examples in figure 8 we obtain the following binary relation :

If we limit the research to path with 3 nodes maximum, seen 3 times or more on the set of example in figure 8 we obtain the following set of features (figure 10)

With this set of paths we obtain the following $\wedge$-reduced context (figure 11 ) :
For this example, we have limited our research to paths then we have only a subset of all $\wedge$-irreducible elements since :
1) all the graphs aren't explored (path length 2).
2) the support parameters reduce the set of paths.
So we have a sub-lattice of the complete Galois lattice.

With a support of 1, we have the following (figure 6.4 ) set of $\wedge$-irreducible path elements (length < 3) :

| | $E_0$ | $E_1$ | $E_2$ | $E_3$ | $E_4$ |
|---|---|---|---|---|---|
| $Rectangle$ | 1 | 1 | 1 | 1 | 1 |
| $on$ | 1 | 1 | 1 | 1 | 1 |
| $right$ | 1 | 1 | 0 | 0 | 1 |
| $Circle$ | 1 | 1 | 0 | 1 | 1 |
| $Rectangle \rightarrow on$ | 1 | 1 | 1 | 0 | 1 |
| $on \rightarrow Rectangle$ | 1 | 1 | 1 | 1 | 0 |
| $on \rightarrow Circle$ | 1 | 1 | 0 | 0 | 1 |
| $Rectangle \rightarrow on \rightarrow Circle$ | 1 | 1 | 0 | 0 | 1 |

**FIG. 10.** paths with maximal length 2 with support 3.

| | $E_0$ | $E_1$ | $E_2$ | $E_3$ | $E_4$ |
|---|---|---|---|---|---|
| $Circle$ | 1 | 1 | 0 | 1 | 1 |
| $Rectangle \rightarrow on$ | 1 | 1 | 1 | 0 | 1 |
| $on \rightarrow Rectangle$ | 1 | 1 | 1 | 1 | 0 |
| $right, on \rightarrow Circle, Rectangle \rightarrow on \rightarrow Circle$ | 1 | 1 | 0 | 0 | 1 |

**FIG. 11.** Matrix of $\mathcal{R}$.

The Galois lattice construct from this binary relation is the Galois lattice of the figure 4. For this example, the Galois lattice constructed will be isomorph to the structural Galois lattice of the figure 8 since the set of $\wedge$-irreducible elements are the same. So for this example, the description of the examples with labeled graph or by path of length 2 are equivalent.

### 6.5 Experimentation

We have made another experimentation on real data. This data are the classical Mutagenesis problem described with graph [11]. For this experimentation we don't use the example/counter-example knowledge. To reduce the complexity, for the research of patterns, we use, here also, two parameters : the support number s (this would mean that the corresponding pattern (path or tree) occurs in s graphs in the original database) and the maximal size of the patterns m (specify that the largest frequent pattern to be examined has m nodes). We have made one experimentation where s vary from 70 to 4 where m is 6. In the second experimentation s is 20 and m change from 2 to 8.

From the set of patterns extracts from the graphs by Gaston, we have merge in one pattern all the patterns with exactly the same extension. This give a more precise idea on the set of $\wedge$-irreducibles. For example with parameters s=20, m=8 we obtain 63487 patterns from Gaston but there is only 1121 patterns with different extensions.

The vertical axis give the number of pattern . The horizontal axis is the support parameters. We have 230 examples and the support varies from 70 (30.4 %) to 4 (1.7%).

| | $E_0$ | $E_1$ | $E_2$ | $E_3$ | $E_4$ |
|---|---|---|---|---|---|
| $Rectangle \rightarrow on$ | 1 | 1 | 1 | 0 | 1 |
| $on \rightarrow Rectangle$ | 1 | 1 | 1 | 1 | 0 |
| $Circle$ | 1 | 1 | 0 | 1 | 1 |
| $Square, on \rightarrow Square$ | 0 | 0 | 1 | 0 | 1 |
| $Circle \rightarrow right$ | 0 | 1 | 0 | 0 | 1 |
| $right \rightarrow circle, Rectangle \rightarrow right$ | 1 | 0 | 0 | 0 | 0 |
| $Circle \rightarrow on$ | 0 | 0 | 1 | 0 | 0 |

**FIG. 12.** Matrix of $\mathcal{R}$.

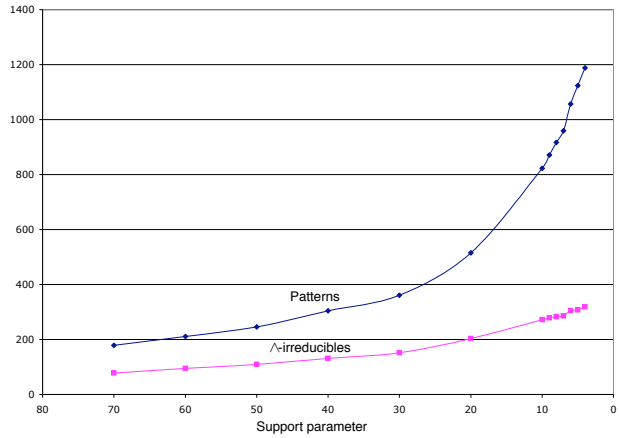


**FIG. 13.** Experimentation on mutagenesis data : modification of the support parameter

The upper graphic (patterns) is the number of patterns (with for each couple of patterns $p_1 \neq p_2$, e($p_1$) $\neq$ e($p_2$)) extracted by Gaston. The lower graphic ($\wedge$-irreducibles) is the number of $\wedge$-irreducible element for the corresponding set of patterns. The memory space need is less then 600k and the time, for the research of the all the $\wedge$-irreducibles element varies from 2s to 6s on an IMac G5 1.8Mhz computer.

For the first experimentation, the reduction factor is 3.5 for 1027 patterns and 5.2 for 3781 patterns. For the second experimentation, the reduction factor is 1.1 for 10 patterns (with 1 or 2 nodes) to a reduction of 3,4 for 1121 patterns (with a maximum of 8 nodes).

From these experimentations, we find an augmentation of the reduction factor which reduce the exponential augmentation of the number of patterns. We think that this property comes from the position of the $\wedge$-irreducible elements in a lattice. We have " for an element x of (a lattice) L let denote by $J_x$ the subset of $\wedge$-irreducibles below x and let $M_x$, dually, be the set of $\vee$-irreducibles above x. A fundamental if obvious remark is

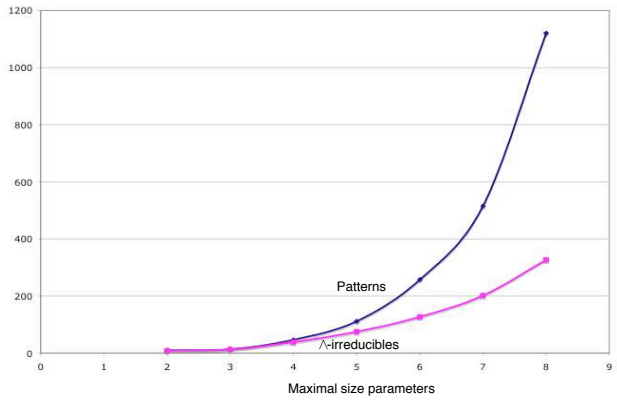


**FIG. 14.** Experimentation on mutagenesis data : modification of the maximal size parameter

that : the higher is x, the larger is $J_x$ while $M_x$ is smaller ... From a technical viewpoint, the structure of a lattice L is obviously encoded into the order relation on irreducibles by the bijection x $\rightarrow$ ($J_x$,$M_x$) (all x $\in$ L, since x is less or equal to y iff $M_x$ is a superset of $M_y$ and $M_{x \vee y}$= $M_x \cap M_y$"[14] Now, if we consider the property 10, we can say (with only an empirical proof here) that the $\wedge$-irreducible elements are "statically more" near the top of the lattice. So the specialization search is a good heuristics for the research of a subset of the $\wedge$-irreducible elements.

## 7 Conclusion

In this paper we have given some links between formal concept analysis and graph mining. The generalization of the formal concept analysis too general description of the examples, and in particular graph description, give a formal definition of the research space. We prove that, with specific graph description of the examples we can have a partial order and a product operator. In this case the structure of the classification space is a lattice. The structure of this space allows the use of classical algorithm for the construction of all the general concepts. But the construction of this lattice is made by a generalization operator. In many real world application, we need a specialization operator. In the second part of the paper we propose an answer to this problem with the definition of an equivalence between description language. This equivalence uses some particular elements of the lattice : the irreducible elements. These elements allows a re-description of the example into a propositional description but the research of these element is a new problem. We propose and online algorithm for the research of a subset of the set of $\wedge$-irreducible elements. So we can remove, from the set of patterns found with a graph mining algorithm, all the patterns which are redundant for the classification task. An experimentation follows which prove the validity of the approach for the reduction of and large number of patterns to a more practical size.

## Références


1. Agrawal.R, Imelinsky.T, Swami.A, Mining association rules between sets of items in large database in proc. ACM SIGMOD'93 conf. (1993) 207-216
2. Annalisa Appice and Michelangelo Ceci and Simon Rawles and Peter Flach, Redundant feature elimination for multi-class problems, *Proceedings of the 21st International Conference on Machine Learning (ICML'2004)*, ISBN :1-58113-838-5, ACM, 33–40, 2004.
3. Alphonse, E. , Rouveirol C., "Lazy propositionalization for Relational Learning In Horn W.," *14th European Conference on Artificial Intelligence, (ECAI'00)*, pages 256-260, IOS Press,, Berlin, Allemagne, 2000.
4. Barbut.M, Monjardet.B.., "Ordre et classification," *Hachette*, Paris, 1970.
5. Berge.C., "Graphes and Hypergraphes," *Bordas*, Paris, 1973.
6. Birkhoof, 1967, "Lattice theory", Third edition, American Mathematical Society, Providence , RI 1967.
7. Bordat, JP, "Calcul pratique du treillis de Galois d'une correspondance," *Math.Sci.humaines*, 24˚année, 96 :31-7, 1986.
8. A.L.Blum, P.Langley. Selection of relevant features and examples in machine learning. Artificial intelligence,97 :245-271,1997
9. Chaudron.l.,Maille.N., "Generalized Formal Concept Analysis," *ICCS'2000* Lecture notes in Artificial intelligence, 1867,pp. 357-370, 2000.
10. Chein.M., "Algorithme de recherche de sous-matrice premiere d'une matrice," *bull. math. R.S. Roumanie* 13, 1969.,
11. D.J.Cook, L.B.Holder. Graph-based data mining. IEEE Intelligent Systems, 15(2) :32-41,2000
12. Usefulness of the Karp-Miller-Rosenberg algorithm in parallel computations on strings and arrays Theoretical Computer Science,Volume 88 , Issue 1 (September 1991), Pages : 59 - 82 ,ISSN :0304-3975
13. Dehaspe.L, Toavonen.H, "Discovery of frequent datalog patterns," *Data Mining and knowledge Discovery,*3(1) :7-36.1999.
14. Duquenne .V, "Latticial structure in data analysis," *Theorical computer science*, 217 :407-436,1999.
15. B.Ganter, R.Wille, « Formal concept analysis, mathematical foundation », 1998, Springer Verlag Ed. ] R.Godin, R.Missaoui 1994, An incremental concept formation approach for learning from databases Theorical computer science 133 (1994) 387-419
16. Ganter.B., Kuznetsov.S., "Pattern Structures and Their Projections," *ICCS'2002* Lecture notes in Artificial intelligence, 2002.
17. R.Godin, R.Missaoui, "An incremental concept formation approach for learning from databases," *Theorical computer science*,133, 387-419, 1994.
18. A.Inokuchi,T.Washio,H.Motoda. Complete mining of Frequent patterns from Graphs : Mining Graph Data. Journal of Machine Learning,Kluwer Academic publishers, 50,321-354,2003.
19. J.Kearns, "The computational complexity of machine learning" *The MIT Press*,176p. 1994.
20. D.Koller, M.Sahami. Toward optimal feature selection. In proceedings of the thirteenth international conference on machine learning, pages 284-292, 1996.

21. Kramer.S, Lavrac.N., Flach.P.A, “Propositionalization approaches to relational data mining,” *Relational data mining*, Springer 2001.

22. Lavrac.N,Zelezny.F. Flach.P.A, “RSD :Relational subgroup discovery through first-order feature construction,” *Inductive logic programming conference*, Springer 2002.

23. N.Lavrac, P.Flach, An extended transformation approach to Inductive Logic programming. Mars 2000. Rapport de l’universite de Bristol. Departement of computer sciences.CSTR-00-002

24. Liquière.M , J Sallantin, 1989 "INNE : A structural learning algorithm for noisy Data", European Working Session on Learning, Decembre 1989 pp 111-123

25. Liquiere.M, Sallantin.J 1998 “Structural machine learning with Galois lattice and Graphs.”, Machine Learning : Proceedings of the 1998 International Conferences,Margan Kaufmann ed (ICML 98) PP 305-313

26. Liquière.M 2001 “Du structurel au propositionnel, une approche formelle.” Grenoble,25 juin,CAP01,http ://www.lirmm.fr/ liquiere/papiers/cap2001.pdf

27. Lloyd. J.W., 2000 " A Logical Setting for the Unification of Attribute-Value and Relational Learning", ICML2000 Workshop on "Attribute-Value and Relational Learning"

28. Mephu Nguifo E. 1994 “Galois lattice, A framework for concept learning- design, evaluation and refinement”,pp 461-467, Tool with AI, 1994

29. Mugnier.M.L., “Knowledge representation and reasoning based on graph homomorphisms,” *Proc. 8th Int Conf on Conceptual Structures*, ICCS’2000, lecture notes in artificial intelligence,1867,pp.172-192,2000.

30. Siegfried Nijssen and Joost Kok. A Quickstart in Frequent Structure Mining Can Make a Difference. Proceedings of the SIGKDD, 2004. http ://www.liacs.nl/home/snijssen/gaston/

31. Norris E.M., “An algorithm for computing the maximal rectangles of a binary relation,” *Journal of ACM,* 21 :356-366, 1974.

32. L.Nourine, O.Raynaud, “A fast algorithm for building lattices,” *Information processing letters*, 71, 199-204,1999.

33. Plotkin G.D., “A further note on inductive generalization,” *in : B.Meltzer and D.Michie (eds), Machine intelligence,* vol 6, Edinburgh University press, Edinburgh, 101-124, 1971.

34. M.Ricordeau, Q-Concept-Learning : Généralisation à l’aide de treillis de Galois dans l’apprentissage par renforcement RFIA, Toulouse, janvier 2004

35. Sowa.J.F., “Conceptual structures - information processing in mind and machine,” *Reading, M.A. : Addison-wesley,* 1984.

36. G. Stumme and R. Taouil and Y. Bastide and L. Lakhal", "Conceptual Clustering with Iceberg Concept Lattices", Proc. GI-Fachgruppentre :en Maschinelles Lernen ’01,Universitat Dortmund 763, (FGML 2001),2001.

37. Wille.R, 1989.”Knowledge acquisition by methods of formal concept analysis”, in E.Diday, editor, Data analysis, Learning Symbolic and numeric knowledge, pages 365-380.1989.

38. Lei Yu and Huan Liu, Efficiently handling feature redundancy in high-dimensional data, *KDD ’03 : Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003,685–690,Washington, D.C.,ACM Press.

39. Zaki,, “SPADE : An efficient algorithm of mining frequent sequences.” *Machine learning journal,* 42 (1/2) :31-60, 2001.

40. Zhou.H, “Multiplicativity. Part I Variations, Multiplicative Graphs and Digraphs.” *Journal of graph Theory,* Vol 15,N°5,469-488. 1991.