Structural machine learning with Galois lattice and Graphs.

Michel Liquiere
Lirmm,
161 rue ADA
34392 Montpellier cedex 5
France
Liquiere@lirmm.fr

Jean Sallantin
Lirmm,
161 rue ADA
34392 Montpellier cedex 5
France
Sallantin@lirmm.fr

Abstract

This paper defines a formal approach to learning from examples described by labelled graphs. We propose a formal model based upon lattice theory and in particular with the use of Galois lattice. We enlarge the domain of formal concept analysis, by the use of the Galois lattice model with structural description of examples and concepts. Our implementation, called "Graal" (for GRAph And Learning) constructs a Galois lattice for any description language provided that the two operations of comparison and generalization are determined for that language. We prove that these operations exist in the case of labelled graphs.

1. INTRODUCTION

The Galois lattice is the foundation of a set of conceptual classification methods. This approach defined by Barbut and Monjardet (Barbut, 1970), was popularized by (Wille, 1982), (Wille, 1992), who used this structure as the kernel of formal concept analysis.

Wille proposed considering each node of a Galois lattice as a formal concept. Each node has two parts: the extension (a subset of the examples) and the intention (the description). In addition, the lattice gives the relations (generalization/specialization) between concepts. An advantages of this formalization is a good description of the concept space. Additionallly, there are many methods for the construction of such lattice (Step search (Chein, 1969) depth first search (Bordat, 1986), incremental construction (Ganter, 1988), (Missikoff, 1989)).

In the context of machine learning, the automatic construction of such a hierarchy can be viewed as an unsupervised conceptual classification method as seen in (Michalski, 1982) because we give a general method and look for all the concepts that can be extracted from the examples.

In this way, research space is not limited by the use of parameters although this method cannot be used in all practical applications. Its advantage is that we can study precisely the impact of biais and heuristic.

An important limitation of the method using the Galois lattice is the classical propositional description of the examples (Wille, 1982), (Ganascia, 1993), (Mephu Nguifo, 1994), (Carpineto, 1994) . There is a great deal of research on the extension of the description language: valued attributes (Wille, 1989), (Carpineto, 1994), term (Daniel-Vatonne, 1993), graph (Liquiere, 1989), (Godin, 1995).

In the case of structural description, the actual methods use a two step mechanism.

1) the goal of the first step is to find structures repeated in the set of descriptions of the examples.

2) the second step uses the structures found and changes the description of the example. Each structural description is converted in a list of binary attributes (one attribute by structure). An attribute is true if the associated structure appears in the example.

• in our work (Liquiere, 1989), (Liquiere, 1994), we used labelled graphs, and the goal of our first step was to find repeated paths and trees in the description of the examples.

• in the work of (Daniel-Vatonne, 1993), the description language is based upon rooted tree (term) and the first step research path.

• Godin, Mineau (Godin, 1995) uses a similar method with conceptual graphs.

The first step research repeated triplet graphs (graph like <Object>-relation-<Object>). This limits the complexity of the research. The second step finds sets of triplet graphs viewed in the same set of examples, but the link between the nodes are overlooked. So the structural descriptions of the examples are not exploited.

In this paper we give a general one step mechanism, without changing the description of the examples. This mechanism uses a generalization operation and we specify this operation for different classes of description languages.

This paper is organized as follows.

A generalized method of learning from examples is presented in section 2.

In section 3, we specify this model for description with labelled graphs.

Then, we study the complexity of the operation in the case of descriptions with labelled graphs, in section 4

Finally, in section 5 we give an example of the results found.

2. THE GALOIS LATTICE AS A CORRESPONDANCE BETWEEN TWO LATTICES

Using lattice theory, the formal framework is based on the use of two lattices as in (Ganascia, 1993). This model, uses lattice results given in (Birkoff, 1967) and (Barbut, 1970). This approach was used in machine learning (Ganascia, 1993), but only in propositional description.

Ganascia writes: "this framework is adequate to represent classical top-down induction systems .. but it is too restricted to formalize first order logic languages …"

In fact, this approach can be used for structured description as well. Thus there is an unifying method for many types of description language.

2.1 Two isomorph lattices

This formalization is based on the use of two lattices: the *description lattice D* and the *instance lattice I.*

• The *instance lattice I*, corresponds to the set of parts of the *training set* ξ and is ordered by the inclusion relationship which is noted $\supset$ where A $\supset$ B means that A is included in B. Given two elements a and b, the least upper bound - and the greatest lower bound- corresponds to the classical union -i.e $\cup$ – and intersection -i.e $\cap$ –.

• The *description lattice D* contains all the possible descriptions ordered by a $\geq$ *relation*. This relation $\geq$corresponds to the generalization relationship.

$\geq$: D $\times$ D $\rightarrow$ {true, false} for two descriptions d_1, $d_2 \in$ D, $d_1 \geq d_2$ means that d_1 is more general than d_2. Let us just consider that it structures the description space with a partial ordering.

If $\geq$is a pre-order (antysymmetry property not fulfilled), we can define the *equivalence relation.* $\equiv$: D $\times$ D $\rightarrow$ {true, false}, $d_1 \equiv d_2$ iff $(d_1 \geq d_2)$ and $(d_2 \geq d_1)$

Because D is a lattice, two elements of D have *a least upper bound*. We note $d_1 \wedge d_2$ this bound. This is the least general generalisation of d_1 and d_2.

We have $\wedge$: D $\times$ D $\rightarrow$ D, if $d \geq d_1$ and $d \geq d_2$ then $d \geq d_1 \wedge d_2$.

This is a generalization operator (Plotkin, 1971), as defined in (Muggleton, 1994). For a set of description S,"A minimal generalization G of S is a generalisation of S such that S is not a generalisation of G, and there is no generalization G' of G such that G is a generalization de G' ".

2.2 Galois lattice.

Let us begin by building two correspondances between the lattice I and D.

First there is a *mapping d* between set ξ and the description space D: d: $\xi \rightarrow$D, for $e_i \in \xi$, $d(e_i) \in$ D is the description of the example e_i.

For example:

• with a propositional description, $d(e_i)$ is a list of attributes.

• in case of structural description, $d(e_i)$ can be a graph.

Now, from this simple description mapping, we can build two correspondences between I and D.

The correspondance α*:* D -> I associates each description d of D the set of all instances of the training set ξ which are covered by d.

$$\alpha(d)=\{e_i \in \xi | \ d \geq d(e_i)\}$$

Properties 1

1) $d \geq d' \Leftrightarrow \alpha(d) \supseteq \alpha(d')$

2) $\alpha(d_1 \wedge d_2) = \alpha(d_1) \cup \alpha(d_2)$

Proof in appendix

The correspondance β: I -> D is equivalent to making the least general generalization for the description of all the elements of H $\subseteq \xi$. This means that:

$$\beta(H)= \bigwedge_{e \in H} d(e)$$

Theorem 1 The correspondance α et β defines a *Galois connection* between I and D.

Proof in appendix, see also (Ganascia, 1993).

Now we have a generalization of the classical definition of concept.

For a set of example ξ, for a description space D, for an instance space I, *a concept C* is a pair [Ext $\times$ Int] with:

• Int $\in$ D | Int=β(Ext)= $\bigwedge_{e \in Ext} d(e)$

• Ext$\in$ I | Ext=α(Int)={$e_i \in \xi$| Int $\geq d(e_i)$}.

All the concepts are ordered by the superconcept-subconcept (generalisation-specialisation) *relation* $\gtrsim_c$.

$$[E_1, I_1] \gtrsim_c [E_2, I_2] \text{ iff } E_1 \supseteq E_2 \text{ and } I_1 \geq I_2$$

With $\gtrsim_c$, the set of all concepts has the mathematical structure of a complete lattice and is called the *Galois Lattice of the context (*ξ *, d, D).*

3. DEFINITION OF THE ORDER (≥) AND GENERALIZATION OPERATION (∧) FOR LABELLED GRAPHS

In section 2, we proposed a formal model. In this model we defined two basic operations ≥ and ∧. If these operations verify different properties (order, generalization operator), then the concept space is a Galois lattice.

Our goal is to use this model for structural description, more precisely for graphs descriptions.

In order to demonstrate this we must first define the operations ≥, ∧ and prove that the description space D is a lattice.

3.1 ≥ Definition for labelled graphs

In this paragraph, we define an pre-order between graphs using the homomorphism relation. We will show (3.2) that for a class of graphs (core graphs), this pre-order is an order.

Notations

We note a *graph G:(V,E,L)* .

The *vertex set* of G is denoted by *V(G).*

The *edge set* of G is denoted by *E(G)*. Each edge is a ordered pair (v_1,v_2), $v_1,v_2 \in V(G)$.

The *Label set* of G is denoted by *L(G).* For a vertex v we note L(v) the label of this vertex.

In the following paragraphs, we give properties for directed graphs, these properties are true as well for undirected graphs.

Definition *labelled graph homomorphism*

A *homomorphism* f:G_1->G_2 is a mapping

f:$V(G_1)$ -> $V(G_2)$ for which $(f(v_1), f(v_2)) \in E(G_2)$ whenever $(v_1,v_2) \in E(G_1)$ and $L(v_1)=L(f(v_1))$

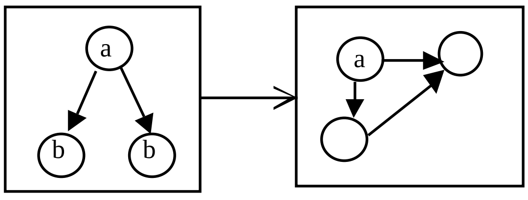


Figure 1: Homomorphism example G1 -> G2

This is not the classical subgraph isomorphism relation.

Operation ≥: D × D → {True, False}

For two labelled graphs G_1:(V_1,E_1,L_1) and G_2:(V_2,E_2,L_2), we note $G_1 \geq G_2$ iff there is a homomorphism from G_1 into G_2.

Operation ≡ : D × D → {True, False}

Two labelled graphs G_1 and G_2 are *homomorphically equivalent*, denoted by $G_1 \equiv G_2$, if both $G_1 \geq G_2$ and $G_2 \geq G_1$.

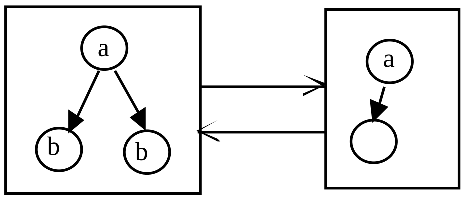


Figure 2: $G_1 \equiv G_2$

Operation ≠: D × D → {True, False}: $d_1 \neq d_2$ iff not ($d_1 \equiv d_2$)

3.2 D for labelled graphs

The homomorphism relation is only a pre-order because the antisymmetry property is not fulfilled (Chein, 1992). An order relation between element of D is necessary in order to use results of section 2.

The same problem occurs in Inductive Logic Programming (Muggleton, 1994)

“Because two clauses equivalent under θ-subsumption are also logically equivalent (implication), ILP systems should generate at most one clause of each equivalence class. To get around this problem, Plotkin defined equivalence classes of clauses, and showed that there is a unique representative of each clause, which he named 'the reduced clause' ”.

In the case of labelled graphs, we can use the same strategy. For this purpose, we use the class of core labelled graphs (Zhou, 1991).

Definition *retract*

A strict subgraph G' of G is a retract of G ((Zhou, 1991), if there is a homomorphism called a retraction r: G -> G' such that r(v)=v for each v∈V(G').

Definition *core*

A graph is called a core (or minimal graph (Fellner, 1982), or irredundant graph (Cogis, 1995)) if it has no proper retracts.

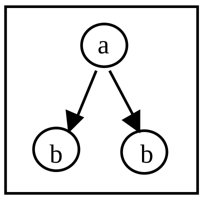
Property 2

For the equivalence relation defined above (≡). An equivalence class of labelled graphs contains one and only one core labelled graph, which is the (unique) graph with the smallest vertex number (Mugnier, 1994).

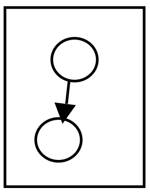
Notation *R*:

We can construct a core graph from a graph as proved by Mugnier (Mugnier, 1994). This operation is called *reduction* (notation R).

Let g be a labelled graph, R(g) is a core labelled graph such that g≡R(g).



G



(G)

Figure 3: Example G and R(G)

We need an order relation in order to use labelled graph. For core labelled graph, we have this order. So, we define *D as a set of core labelled graphs*. All labelled graph description of the example can be converted to an equivalent core labelled graph, using the R operation.

Theorem 2 The restriction of $\geq$to the set of core labelled graphs is a lattice (Zhou, 1991), (Poole, 1993), (Chein, 1994)

For this lattice, the $\vee$ operation is the disjoint sum of the graph, $g_1 \vee g_2 = g_1 + g_2$ (Chein, 1994) (g_1 with g_2 form a new graph).

The $\wedge$ operation is more complex and is defined in the following paragraph.

3.3 $\wedge$ Definition for labelled graphs

The $\wedge$ operation for graph is based on a following classical *Kronecker product operation* $\times$ (Weichsel, 1962).

Definition $\times$ *operation for graphs*

For two graphs, the product $G_1 \times G_2$ has the vertex set $V(G_1) \times V(G_2)$ and the edges $((v_1,v_2), (v'_1,v'_2))$, where $(v_1,v'_1) \in E(G_1)$ and $(v_2,v'_2) \in E(G_2)$.

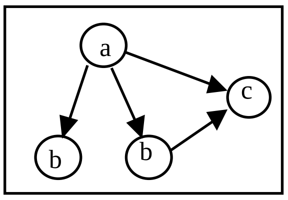
This product operation can be determined for labelled graphs.

Definition $\times$ *operation for labelled graphs*

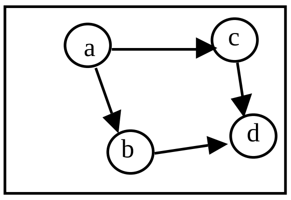
For two labelled graphs $G_1:(V_1,E_1,L_1)$ and $G_2:(V_2,E_2,L_2)$

The product $G(V,E,L) = G_1 \times G_2$ is defined by:

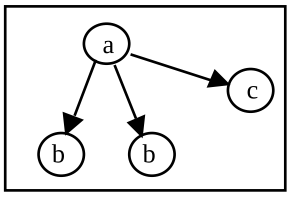
- $L = L_1 \cap L_2$
- $V \subseteq V_1 \times V_2 = \{ v \mid v=[v_1,v_2]$ with $L(v_1) = L(v_2)$ and $L(v)=L(v_1)\}$
- $U= \{(v=[v_1,v_2],v'=[v'_1,v'_2]) \mid (v_1,v'_1) \in V_1$ and $(v_2,v'_2) \in V_2\}$(edge oriented)



G1



G2



G1 x G2

Figure 4: Labelled graphs product

Lemma 1

if G_1,G_2,G are labelled graphs then

a) $G_1 \times G_2 \geq G_1$ and $G_1 \times G_2 \geq G_2$

b) if $G \geq G_1$ and $G \geq G_2$, then $G \geq G_1 \times G_2$

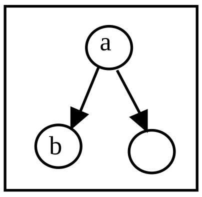
c) $G_1 \geq G_1 \times G_2$ if and only if $G_1 \geq G_2$.

Proof: from the definitions and (Zhou, 1991)

Remark: The product operation can be easely improved when the label set is a hierarchy or a lattice.

Definition *operation* $\wedge$: $D \times D \rightarrow D$

for G_1, G_2 two core labelled graphs, $G_1 \wedge G_2 = R(G_1 \times G_2)$



R(G1 x G2)

Figure 5: $G_1 \wedge G_2$, with G_1 and G_2 defined in Figure 4

3.4 Galois lattice for graphs

Now, We have all the operations for the construction of a Galois lattice when example are described by graphs.

Each node of this Galois lattice is a pair [Ext $\times$ Int] with

Ext is a subset of ξ and Int is a core graph. This core graph is the generalization of the description of the examples in Ext.

Theorem 3

With:

ξ a set of examples,

D a description lattice for core labelled graphs,

d a description mapping, d: ξ->D,

I the instance lattice, $\Pi(\xi)$ set power of ξ,

$\geq$an order relation $D \times D \rightarrow$ {True,false},

$\wedge$ the generalization operation $D \times D \rightarrow D$ for graphs.

We can define α,β.

for $g \in D$, $\alpha(g)=\{e_i \in \xi \mid g \geq d(e_i)\}$

for $H \in I$, $\beta(H)= \bigwedge_{e \in H} d(e)$

The correspondence α et β defines a Galois connection between I and D.

Proof : see lemma 1 and proof for the Theorem 1.

Using this Galois connection, we can define a Galois lattice (see 2.2). We name this lattice *T*.

We have defined a formal model for labelled graphs. This model uses , $\geq$,and $\wedge$ operations in the case of labelled graphs. In the next section, we study the complexity of these operations.

4. THE COMPLEXITY OF GALOIS LATTICE CONSTRUCTION

The complexity in the construction of a Galois lattice in our model, is a function of:
1) the number of nodes in the lattice,
2) the time and space complexity of the operations ($\wedge$,R),
3) the algorithm used for Galois lattice construction (see 5.1)

4.1 the size of the Galois lattice

Property 3: The number of nodes for T can be $2^{|\xi|}$.

proof: It is well known that, Galois lattice can be isomorphic to the power set of ξ (Bordat, 1986) which is the maximal complexity for the size of T.
A similar situation occurs in our model. The proof comes from the fact that each binary attribute description can be converted to a structural description.
For example the list [Big] [Blue] [Expensive] can be structurally described as:

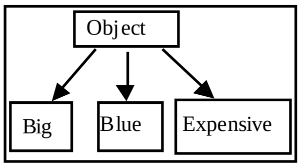


Figure 6: graph representation for an list of attributes.

Using this description, the $\wedge$ operation for the tree representation is equivalent to $\cap$ for the attribute representation.

4.2 Complexity of the $\wedge$ operation

In (Muggleton, 1994) S. Muggleton and L. de Raedt wrote:
"… ILP systems can get around the problem of equivalent clauses when working with reduced clauses only".
This affirmation is true but the problem of the complexity of the R operator has not been taken into account.

• for two labelled graphs, $G_1=(V_1,E_1,L_1)$ and $G_2=(V_2,E_2,L_2)$, the complexity of the product is: $O(n_1 x\ n_2)$ where $n_1=|V_1|$ et $n_2=|V_2|$.

For a set of graph P,

$G=\bigwedge_{Gi \in P} G_i = R(\times_{Gi \in P} G_i)$.

the size of $\times_{Gi \in P} G_i$ can be exponential.

Property 4

the operation R is co-Np-complete (Mugnier, 1994). So, in general application, this operation cannot be used.

However we do have an interesting result:

Property 5 (Mugnier, 1994)
If, for a class of labelled graphs, the homomorphism is polynomial, then the reduction operation is polynomial.

The homomorphism for the following class of labelled graphs is polynomial.

• trees (Mugnier, 1994),

• locally injective graph (Liquiere, 1994) (see definition below)

• 1/2 locally injective graph (see definition below) (see langage theory, automata (Aho, 1986))

Property 6

For a set of path or tree P, $G=\bigwedge_{Gi \in P} G_i$ is polynomial (time and size) (Horvath, 1995)

4.3 Study for a class of Graphs.

We study the complexity of the operation ($\bigwedge$,R) for the class of locally injective graphs (LIG) (Liquiere, 1994).

Notation

For a labelled graph G=(V,E,L), we note $N^+(v)=\{v' \mid (v,v') \in V\}$ and $N^-(v)=\{v' \mid (v',v) \in V\}$.

Definition *LIG graph*

For a labelled graph G=(V,E,L), G is *locally injective* if for each vertex $v \in V$, $\forall v_1,v_2 \in N^+(v)$, $v_1 \neq v_2 \Rightarrow L(v_1) \neq L(v_2)$ and $\forall v_1,v_2 \in N^-(v)$, $v_1 \neq v_2 \Rightarrow L(v_1) \neq L(v_2)$.

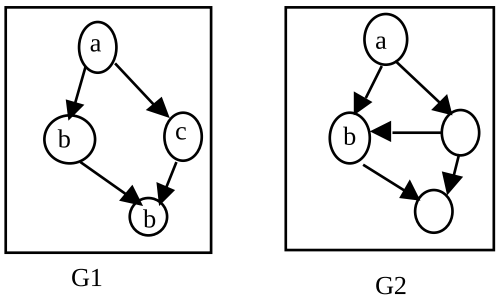


Figure 7. LIG Property

In figure 7, G_1 is an LIG graph. G_2 is not an LIG graph because, for the c node, there is two edges c -> b. G_2 is a 1/2 locally injective graph (see the next definition).

Definition *1/2 locally injective graph*

A *1/2 locally injective graph* is a oriented graph where $\forall v_1,v_2 \in N^+(v)$ (resp. $N^-(c)$), $v_1 \neq v_2 \Rightarrow L(v_1) \neq L(v_2)$.

Property 7: $\geq$ is polynomial for locally injective graph (Liquiere, 1994) and for 1/2 locally injective graph (Aho, 1986).

Property 8: For G_1, G_2 two LIG, G= $G_1 \times G_2$ is a LIG.

Partial proof: come from the definition of $\times$.

Property 9: A connected LIG is an irredundant graph (Cogis, 1995)

Property 10: For G a LIG, we note CC(G) the set of maximal connected subgraph of G. Then R(G)= $\{c_i \in CC(G) | \forall_{i,j,i \neq j}$ there is no projection from c_i to c_j }

Proof: property 9 => Property 10

These properties are interesting because for LIG we can construct the R, $\geq, \times$ and $\wedge$ operations, for two graphs, with a polynomial complexity.

Property 11: For a set of 1/2 locally injective graphs P, $G = \bigwedge_{G_i \in P} G_i$ is size exponential so time exponential (results for deterministic automata (Aho, 1986)).

Table 1: Complexity for different class of language

Language	**≥, ≡, R**	**Size for n graphs**
Path (Horvath,1995)	P	Polynomial
Tree (Horvath,1995)	P	Polynomial
LIG (Liquiere,1994)	P	?
1/2LIG (Aho,1986)	P	Exponential
Graph (Garey,1987)	NPC	Exponential

With P: polynomial, NPC: NP complete.

5. GRAAL IMPLEMENTATION

Traditionally machine learning offers mechanisms for a class of language. The idea is, if an algorithm is good for a general class of language, it would also work well for a less general class included in the first one. It is true, but in many cases, the general mechanism does not use all the interesting properties of the restricted language. So the complexity of the operation is not optimal for this language.

A second drawback of this approach, comes from the need for a translation process. Each description in the restricted language has to be converted into a more general one. For example a list of attributes is converted into a graph (Liquiere, 1994).

In our new method, Graal (for GRAph And Learning), we have implemented a general mechanism where description language and operations $\wedge, \geq$ are parameters. Our tool is generic but it cannot yet be used in practical cases when important sets of examples are described by large graphs. It in fact, an algorithm for formal analysis.

5.1 An utilization of a classical Algorithm for Galois Lattice construction.

We give an algorithm which can be used on any description language with operations $\leq$ and $\wedge$.

This algorithm is based on a classical method (Chein, 1969). Another algorithm can be used (Bordat, 1986) which gives the set of nodes of the Galois lattice and also the set of edges.

We note $[e_i \times d_i]$ the concept numbered i of T^k.

```
$T \leftarrow \emptyset$     /* concept set empty */
/* description of the examples */
$T^1 \leftarrow \{[\{i\} \times d(e_i)]\}$ and $i \in [1, |\xi|]$
$k \leftarrow 1$
**While** $|T^k| > 1$ **do**
$T^{k+1} \leftarrow \emptyset$
  **For each** $i<j$ and $i,j \in [1, |T^k|]$ (so we have: $[e_i \times d_i]$ $[e_j \times d_j]$ ) /* we create a new concept from two concepts already found in the previous step*/
        $d_{ij} \leftarrow d_i \wedge d_j$ /* description for the new concept */
        **if** $d_{ij} \neq \emptyset$ **then**
/* test if there is a concept with the same description */
            **if** $d_{ij} \in T^{k+1}$ **then**
                $e_{ij} \leftarrow e_{ij} \cup e_j$
            **else**
                $T^{k+1} \leftarrow T^{k+1} \cup [e_i \cup e_j \times d_{ij}]$
        /* test if the description is a generalisation */
            **if** $d_{ij} \equiv d_i$ **then**
                $T^k \leftarrow T^k - [e_i \times d_i]$
            **if** $d_{ij} \equiv d_j$ **then**
                $T^k \leftarrow T^k - [e_j \times d_j]$
            **End-if**
  **end-for**
$T \leftarrow T \cup T^k$
$k \leftarrow k+1$
**end-while**
$T \leftarrow T \cup T^k$
```

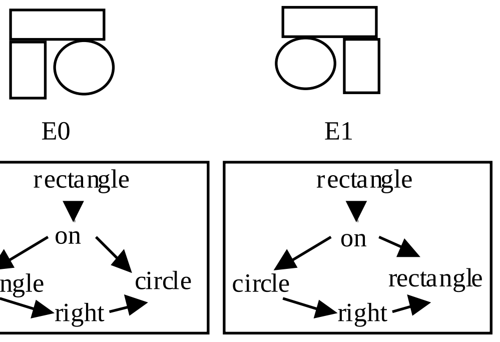
Graal is written in Java language and uses object programming properties. We have defined an abstract class (interface) so a user can add his own description language if he implements the interface.

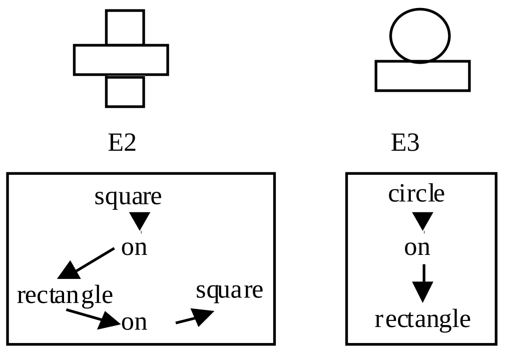
The complexity of Galois Lattice construction with the Bordat's algorithm (Bordat, 1986) is less than $O(n^3*p)$ where n is the number of objects and p the size of T.

5.2 An experimental example.

We present an example where each object is described by a locally injective labelled graph.

We use a classical example based on arch definition.





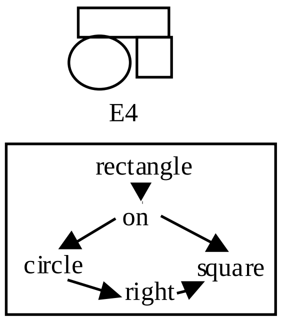


Figure 8: set of examples

The lattice is:

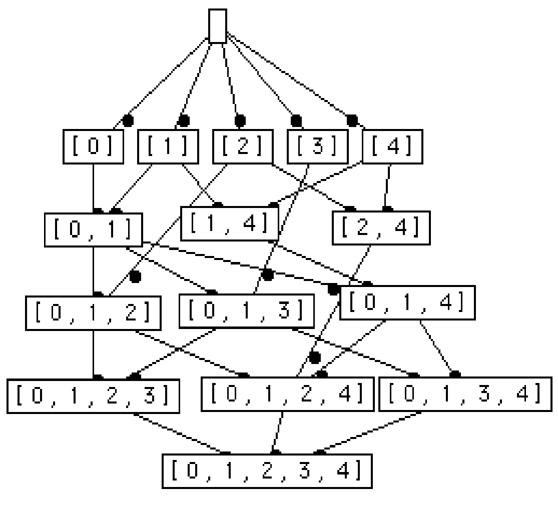


Figure 9: the structure of the Galois lattice for our set of examples.

For each node of the lattice there is a pair consisting of a graph and a set of examples. Additionally, if $node_1$.. $node_k$ are linked to $node_p$ then $node_p$ is the least common superconcept (generalisation) of $node_1$... $node_k$.

In figure 9 we observe the subset of examples (extension). In out tool, by double cliking on a node we obtain the following descriptions.

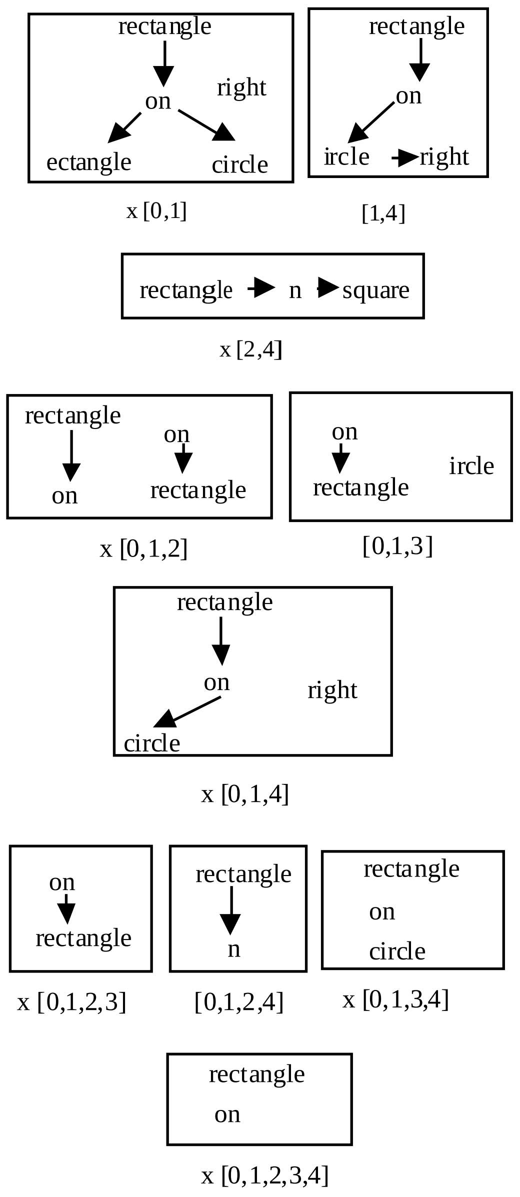


Figure 10: the graph and subset for each node.

This lattice gives all the classification for the examples without duplication. All concepts are differents (description and extension), two differents descriptions necessarily have distinct extensions.

Remark: For this example, the unconnected nodes like *on* can be interpreted as: there is something *on* something.

6. CONCLUSION

Our work enlarges and expands the domain of formal concept analysis by demonstrating that the Galois lattice can be used for structural description.

Coming from graph theory, our work provides operations and shows that they can be used to build a generalization operator for labelled graphs.

In addition, the LIG graphs we use are an excellent compromise between complexity and expressiveness.

Our method, written in Java, offers a general tools for formal structural concept analysis.

We are now working on the following improvements:

- To prove that LIG is PAC learnable or not,
- a survey of classical Galois lattice results in case of structural concept description,
- an implementation of heuristic in Graal, to make Graal a tool for practical application,
- an improvement of the approaches with categorical operations.

References

F.Aho, R.Sethi, J.Ullman, "Compilers, Principles, techniques and tools", Addison wesley 1986.

M.Barbut, B.Monjardet, 1970, " Ordre et classification: algébre et combinatoire". Hachette, Paris 70.

Birkhoof, 1967, "Lattice theory", Third edition, American Mathematical Society, Providence , RI 1967.

Bordat, JP 1986. “Calcul pratique du treillis de Galois d'une correspondance.” Math.Sci.humaines, 24°année, 96:31-7.

C.Carpineto,G.Romano,"A Lattice Conceptual Clustering System and its Application to Browsing Retrieval", Machine Learning 24,95-122 (1996).

M.Chein,"Algorithme de recherche de sous-matrice premiere d'une matrice", bull.math.R.S.Roumanie,13, 1969.

M.Chein, M.L Mugnier, "Conceptual Graphs: fundamental notions", RIA Volume 6- n°4, 1992,p 365-40

M.Chein, M.L.Mugnier "Conceptual Graphs are also Graphs", Research report.

M.C Daniel-Vatonne, C de la Higuera “Les termes: un modéle algébrique de représentation et dee structuration des données “, Mathématique, informatique et sciences humaines, 122, 1993, p41-63.

T.Dietterich "unsupervised concept learning and discovery". In Readings in machine learning. pages 263-266. Morgan Kaufmann 1990.

W.D. Fellner, “On minimal Graphs”, Theoret. Comp. sci. 17 (1982) 103-110.

J.G Ganascia, "TDIS: an Algebraic Formalization", pp1008-1013, IJCAI 1993.

B. Ganter, “Composition and Decomposition in Formal Concept Analysis”, Classification and related Methods of Data analysis, p 561-566, North-Holland, Springer, NY.

R.Godin,G.Mineau,R.Missaoui,H.Mili, “ Méthode de classification conceptuelle basées sur les treillis de Galois et application “, RIA,Vol 9 n°2, 1995.

M.R. Garey, D.S. Johnson, “Computers and Intractability: a guide to the Theory of NP-Completeness”, W.H Freeman, San Francisco (USA).

P.Hell, “Retract in graphs.”, Springer Verlag Lecture notes in mathematics 406 (1974) 291-301.

O.Cogis, O.Guilnaldo, "A linear descriptor for conceptual graphs and a class for polynomial isomorphism test " pp 263-277, LNAI 954, Third conf on Concept.Struc, ICCS 1995.

T.Horvath, G.Turan, "Learning logic programs with structured background knowledge", Inductive Logic Programming conference, p53-76, Leuven, 1995.

M.Liquiere,"INNE:(Induction in Network", EWSL, Montpellier, 1989.

M.Liquiere, " Graphs and structural similarities", in New approaches in classification and data analysis. Springer Verlag. Studies in classification, data analysis. 1994.

M.Liquiere, O.Brissac, "A class of conceptual graphs with polynomial iso-projection", In proceedings of the international Conference on Conceptual Structures, ICCS, 1994.

E. Mephu Nguifo, "Galois lattice: A framework for concept learning - design, evaluation and refinement", pp 461-467, 6° Tool with AI, New Orleans TAI 1994.

Michalski,R.S, Stepp.R " Learning from observation: conceptual clustering ." In machine Learning: An artificial Approach (vol 1). Tioga publishing.

M.Missikoff, "An algorithm for insertion into a Lattice; Application to Type Classification", Proc of th 3rd Int. Conf FODO, Paris, Juin 1989, Springer Verlag, LNCS 367.

S.Muggleton, L de Raedt, "Inductive Logic Programming: Theory and Methods", J.Logic Programming 1994:19, 20:629-679.

M.L Mugnier, "On Generalisation / Specialisation for Conceptual Graphs". J.expt.theor.Artif.intell, 6, 1994.

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

J.Poole,J.A.Campbell," A novel algorithm for matching conceptual and related graphs", ICCS 93, pp 293-307.

J.F.Sowa,"Conceptual structures: Information processing in mind and machine", Addison-wesley Pub., Reading, 460p, 1984.

P.M. Weichsel, "The kronecker product of graphs." Proc.Am.Math.Soc. 13 (1962) 47-52.

R.Wille " Restructuring Lattice Theory: an Approach Based on Hierarchies of concepts", in Ordered sets, I.Rival (ed), Reidel, p 445-470, 1982.

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

R.Wille "Concept lattices and conceptual knowledge systems." In Computers Math.application, volume 23,pages 493-515.1992.

H.Zhou, "Multiplicativity. Part I Variations, Multiplicative Graphs and Digraphs." Journal of graph Theory, Vol 15,N°5,469-488(1991).

Appendix

Properties 1 proof

Proof 1)

$g \geq g' \Leftrightarrow \alpha(g) \supseteq \alpha(g')$

=> property of the order relation $\geq$

<= α definition

$\alpha(g_1 \wedge g_2) = \alpha(g_1) \cup \alpha(g_2)$

Proof 2)

$\alpha(g)=\{e \in \xi / \ g \geq d(e)\}$

We have $g_1 \wedge g_2 \geq g_1$ and $g_1 \wedge g_2 \geq g_2$ so $\alpha(g_1 \wedge g_2) \supseteq \alpha(g_1) \cup \alpha(g_2)$

We know that $g_1 \wedge g_2$, $\forall g$, $g \geq g_1$ and $g \geq g_2$ then $g \geq g_1 \wedge g_2$.so $\alpha(g) \supseteq \alpha(g_1 \wedge g_2) \supseteq \alpha(g_1) \cup \alpha(g_2)$

if $\alpha(g_1 \wedge g_2) \supset \alpha(g_1) \cup \alpha(g_2)$ then for g'/ $\alpha(g')=\alpha(g_1) \cup \alpha(g_2)$ we have

$g_1 \wedge g_2 \geq g'$ and $g' \geq g_1$ then $g' \geq g_2$ so we don't have the property of $g_1 \wedge g_2$.

then $\alpha(g_1 \wedge g_2) = \alpha(g_1) \cup \alpha(g_2)$

Theorem 1 proof

α and β is a Galois connection iff :

a) $\forall$ I_1 and $I_2 \in I$, $I_1 \subseteq I_2 \Rightarrow \beta(I_1) \leq \beta(I_2)$

b) $\forall$ $g_1, g_2 \in D$, $g_1 \geq g_2 \Rightarrow \alpha(g_1) \supseteq \alpha(g_2)$

and for $h= \alpha \circ \beta$ and $h'=\beta \circ \alpha$

c) $\forall$ $H \in I$, $H \subseteq h(H)$

d) $\forall$ $g \in D$, $g \geq h'(g)$ (remark classicaly we note generalisation $\leq$so we have a more classical definition.

Proof a) We have $g_1 \wedge g_2 \geq g_1$ then

$\beta(I_1)=\bigwedge_{e \in E1} d(e)= g$ and $I_1 \subseteq I_2$ $\beta(I_2)= (\beta(I_1)) \wedge (\beta(I_2 - I_1)$ so $\beta(I_2) \geq \beta(I_1)$

Proof b) We have $g_1 \geq g_2$ and $g_2 \geq g_3 \Rightarrow g_1 \geq g_3$ because $\geq$is an order relation $\alpha(g_2)=\{e \in \xi / \ g_2 \geq d(e)\}$ we have $g_1 \geq g_2$ then $g_1 \geq d(e)$ with $e \in \alpha(g_2)$ so $\alpha(g_1) \supseteq \alpha(g_2)$

Proof c) We have $g_1 \geq g \Rightarrow g_1 \wedge g_2 \geq g$

$\beta(H)=\bigwedge_{e \in H} d(e)$, $\alpha(\beta(H))=\{e \in \xi \ / \ \beta(H) \geq d(e)\}$

But $\forall e \in H$, we have $\beta(H) \geq d(e)$ because $\beta(\{e\} \cup K) \geq d(e)$ property of $\wedge$.

Proof d) We have, $g \geq g_1$ and $g \geq g_2 \Rightarrow g \geq g_1 \wedge g_2$ so

$\forall$ $g \in D$, $h'(g)= \beta(\alpha(g))$, $\alpha(g)=\{e \in \xi / \ g \geq d(e)\}$,

$g \geq \bigwedge_{e \in \alpha(g)} d(e)$ because $g \geq d(e)$with $e \in \alpha(g)$.