ZooM: a nested Galois lattices-based system for conceptual clustering

Nathalie Pernelle[1] and Marie-Christine Rousset[1], Henry Soldano[2], and Véronique Ventos[1]

[1] Université Paris Sud - CNRS(LRI) & INRIA(Futurs)
LRI, Bat. 490,
91405 Orsay, France
{pernelle, mcr, ventos}@lri.fr
[2] Université Paris-Nord
LIPN, UMR-CNRS 7030, av. J.B. Clément
93430 Villetaneuse, France
soldano@lipn.univ-paris13.fr

Abstract. This paper deals with the representation of multi-valued data by clustering them in a small number of classes organized in a hierarchy and described at an appropriate level of abstraction. The contribution of this paper is three fold. First we investigate a partial order, namely nesting, relating Galois lattices. A Nested Galois lattice is obtained by reducing (through *projections*) the original lattice. As a consequence it makes coarser the equivalence relations defined on extents and intents. Second we investigate the intensional and extensional aspects of the languages used in our system ZooM. In particular we discuss the notion of α-extension of terms of a *class language* $\mathcal{L}$. We also present our most expressive language $\mathcal{L}_3$, close to a description logic, and which expresses optionality or/and multi-valuation of attributes. Finally the nesting order between the Galois lattices corresponding to various languages and extensions is exploited in the interactive system ZooM. Typically a ZooM session starts from a propositional language $\mathcal{L}_2$ and a coarse view of the data (through α-extension). Then the user selects two ordered nodes in the lattice and ZooM constructs a fine-grained lattice between the antecedents of these nodes. So the general purpose of ZooM is to give a general view of concepts addressing a large data set, then focussing on part of this coarse taxonomy.

Keywords: (Galois lattice, concept lattice, clustering)

1 Introduction

This paper deals with the representation of multi-valued data by clustering them in a small number of classes organized in a hierarchy and described at an appropriate level of abstraction. In our approach we propose a tool that enables the user build interactively various views of the data.

More precisely, our basic representation is a Galois lattice, i.e. a lattice in which terms of a representational language are partitioned into equivalence classes w.r.t their *extent* (i.e. the set of instances that satisfy the term). Each node of a Galois lattice corresponds to a *concept* represented as its extent and its *intent*. The *intent* of a concept is the most specific term corresponding to the concept extent, and so is the representative of one equivalence class of terms. Other work in Machine Learning addressed a similar issue : which are the relevant terms when searching in a concept space ? Constructive induction [23] is an attempt to discover relevant features in order to reduce both cost of the search and concept representation in concept learning. In a more recent work the top-down search is performed in a reduced concept space corresponding to a Galois lattice [5]. A similar effort was spent for problem solving in constructing a library of partial solutions in a goal space [30]. In that work it were necessary to represent partial solutions in a space less expressive than the original solution space in order to overcome efficiency shortcomings in retrieving previously stored partial solutions. The more general issue of dynamic shift of representations in Machine learning has been extensively investigated [31, 8] and in particular searching *abstractions* space prior to investigate more expressive concep spaces [12]. Galois connections and Galois lattices have been widely used in various fields concerning Data and Knowledge representation and organization [14, 19, 17]. A Galois lattice is a well-defined and exhaustive representation of the concepts embedded in a data set. Various techniques have been proposed to represent the Galois lattice and reduce its size by eliminating part of the nodes [15, 16]. Still the main drawback of this structure is that the size of such a lattice can be very large when dealing with real-world data sets. Our work addresses this problem by taking into account the limitations of a user (what he can take in) and his wishes (what he is interested in). The basic ideas are the use of different languages of description of classes (i.e. languages more or less expressive to describe clusters of instances) and the use of basic types of instances given by the user in order to obtain some a priori clusters that will then be gathered rather than instances.

The result is the system ZooM devoted to the construction of Galois lattices at different levels of abstraction. ZooM first builds a coarse lattice covering the whole set of instances and then refines a small subpart of the coarse lattice delimited by two nodes chosen by a user. More precisely, the instances we consider are described by varied sets of attributes. Attributes are optional as they can appear or not when describing an instance. Furthermore attributes are multi-valued and each instance is labelled by a *basic type*. In the first step, the data are partitioned according to their label: to each label c, we associate a *basic class* named c. The construction tasks are then based on the use of two languages of class description. In $\mathcal{L}_2$, the first language of classes that we consider, values of attributes are omitted and optionality is ignored. The second language of classes, $\mathcal{L}_3$, is more expressive and more precise: value restrictions of attributes are represented, and the language expresses the optionality and/or the multi-valuation of the attributes: a term of such a language can state that a given attribute is optional as it does not appear in all the instances satisfying the term. This second language actually allows us to distinguish new clusters of instances and so to obtain a more precise lattice. Furthermore, the labels are used at different levels. First, the terms that we construct represent clusters of basic classes rather than clusters of instances. Then we partly relax this constraint in the following way : an instance now belongs to the $\alpha - extent$ of a term c when it belongs to the extent of c and when at least α % of the instances sharing the same label also belongs to the extent of c. When $\alpha = 0$, a term then represents an unconstrained cluster of instances and when $\alpha = 100$ a term represents a cluster of basic classes. This allows to deal either with some representational noise in instances (with high values of α) or with heterogeneous user-defined basic classes (using small value of α).

From a theoretical point of view, we propose the general framework of *nested Galois lattices* which formalizes the links between the different lattices constructed by ZooM. More precisely let us consider a set of instances I, the powerset E of I and the set $\mathcal{L}$ of terms. A node in the corresponding Galois lattice is made of an intent (belonging to $\mathcal{L}$) and an extent (a subset of I). Let us consider that equality of intents and equality of extents define equivalence relations on E and $\mathcal{L}$. Then nodes of the Galois lattice are pairs of representatives of the corresponding equivalence classes on E and $\mathcal{L}$. Now, the basic notion underlying nesting of Galois lattices is the projection of partially ordered sets (posets). A projection transforms a poset in such a way that all eliminated elements are larger than some remaining element. This means that an intensional projection $proj_{\mathcal{L}}$ projects any term on a more general term and that resulting terms are unchanged (for instance $proj_{\mathcal{L}}(colour = darkblue) = proj_{\mathcal{L}}(colour = blue) = (colour = blue)$). As a consequence intensional projections result in a coarser language (i.e. a coarser view of what distinguishes instances). On the other hand, an extensional projection transforms a subset of instances in a smaller subset (for instance the α-extent of a term is included in the corresponding extent). More generally a projection (either intensional or extensional) transforms a Galois into a smaller, i.e. coarser, Galois lattice corresponding to larger equivalence classes. Such a coarse Galois lattice is denoted as *nested* in the former Galois lattice. *Nesting* corresponds to a partial ordering of Galois lattices.

The general framework of *nested Galois lattices* is described in the section 2. In section 3, we present the two languages and the various extension functions that will be used in ZooM, and we exhibit the nesting of the corresponding Galois lattices. Section 4 is devoted to the presentation of the system ZooM that has been partly implemented and experimented on real data issued from the GAEL project which aims at building flexible electronic catalogs of products organized as a hierarchy of classes. We then present some experimental results concerning real data coming from the C/Net electronic catalog of computer products (http://www.cnet.com). Finally, we present related work in section 5 and future work in section 6.

2 Nested Galois lattices and posets

2.1 Definitions, and preliminary results

Detailed definitions, results and proofs regarding Galois connections and Galois lattices may be found in [2, 10]. We present in this section necessary results and definitions in order to make the paper self-contained.

Definition 1 (Poset) *A poset is a partially ordered set, i.e. a set with a partial ordering relation $\leq$.*

We will further use two kinds of posets. The first kind will represent a concept language and will be denoted as $(\mathcal{L},\preceq)$ where $c1 \preceq c2$ means that the concept $c1$ is less specific than the concept $c2$ (i.e. all instances of $c2$ also are instances of $c1$). As a consequence $c1 \succeq c2$ means that $c1$ is more specific than $c2$. The second kind of posets will be denoted as $(E, \subseteq)$ and will represent a set of "extents", i.e. a set of subsets of an instance set I. Here $e1 \subseteq e2$ means that the subset $e1$ is included in the subset $e2$. We further denote as ext the correspondence relating an element of $\mathcal{L}$ to an element of E, and we denote as int the correspondence relating an element of E to an element of $\mathcal{L}$.
In most work on Galois lattices in knowledge representation E is the powerset $P(I)$ of an instance set I and $\mathcal{L}$ is the powerset of a property set A (see Example 1 below). However for our purpose we need a more general presentation as we will investigate more expressive languages and will add constraints to *extents* thus resulting in a set E smaller than $P(I)$. Furthermore in what follows there is no formal difference between the two kinds of posets E and $\mathcal{L}$, and all the results hereunder stand for any pair of posets.

Definition 2 (Galois connection) *Let$(\mathcal{L},\preceq)$ and $(E, \subseteq)$ be two posets. Let int and ext be two correspondences defined on E and $\mathcal{L}$ by int: $E \rightarrow \mathcal{L}$, ext: $\mathcal{L} \rightarrow E$ and such that:*

C1- $\forall (e, e1) \in E,\ e \subseteq e1$ *implies* $int(e) \succeq int(e1)$

C2- $\forall (c, c1) \in \mathcal{L},\ c \preceq c1$ *implies* $ext(c) \supseteq ext(c1)$

C3- $\forall c \in \mathcal{L},\ c \preceq int(ext(c))$, *and* $\forall\ e \in E,\ e \subseteq ext(int(e))$

Then $(E,\subseteq,\mathcal{L},\preceq,int,ext)$ defines a Galois connection between $(\mathcal{L},\preceq)$ and $(E,\subseteq)$.

The intuitive meaning of this definition appears through the following definitions and lemmas:

Definition 3 (Closure) *w is a closure operator on the poset P if and only if for any pair (x, y) of elements of P we have:*

- $x \leq w(x)$
- *if* $x \leq y$ *then* $w(x) \leq w(y)$
- $w(x) = w(w(x))$

The element x of P is closed for w if and only if $x = w(x)$

Lemma 1 *int* $\circ$ *ext is a closure operator on* $\mathcal{L}$*, and ext* $\circ$ *int is a closure operator on* E*.*

Lemma 2 *Let c be an element of* $\mathcal{L}$ *then e= ext(c) is closed w.r.t. ext*$\circ$*int. Let e be an element of E then c= int(e) is closed w.r.t. int*$\circ$*ext*

There is a one-to-one correspondence between closed elements of E and closed elements of $\mathcal{L}$. Such closed pairs (e, c) constitutes a poset, built from the Galois connection and such that $e = ext(c)$ and $c = ext(e)$, that we denote hereunder as a Galois poset.

Definition 4 (Galois poset) *Let (E,*$\subseteq$ *,*$\mathcal{L}$*,*$\preceq$*, int, ext) be a Galois connection, then the corresponding Galois poset (G,*$\leq$*) is defined as follows:*

G= $\{$ *(c,e) /c is a closed element of* $\mathcal{L}$*, and e is the closed element of E such that e=ext(c) (i.e. also c=(int(e))*$\}$

(c,e) $\leq$ *(c',e') iff e* $\subseteq$ *e' (which is equivalent to c* $\succeq$ *c')*

Furthermore we define equivalence relations on E and $\mathcal{L}$ as follows:

Definition 5 (Equivalence relations on E **and** $\mathcal{L}$ **)** *Let Eqint and Eqext denote the equivalence relations defined on E and* $\mathcal{L}$ *by:*

$e\ Eqint\ e'\ iff\ int(e) = int(e')$*, and* $c\ Eqext\ c'\ iff\ ext(c) = ext(c')$

We have then the following property:

Lemma 3 *Let e be an element of E, and c be an element of* $\mathcal{L}$*, then ext(int(e)) is the unique greatest element of the equivalence class of Eqint containing e and int(ext(c)) is the unique greatest element of the equivalence class of Eqext containing c.*

This means that a node in a Galois poset corresponds both to an equivalence class of extents and to an equivalence class of intents. Later in the paper we will define Galois posets corresponding to equivalence relations coarser than $Eqext$ and $Eqint$.

Furthermore we are interested in particular posets, namely lattices: lattices are particular posets in which each pair of elements have a least upper bound and a greatest lower bound.

Definition 6 (lattice) *A poset (P,*$\leq$*) is a lattice iff for all pair (x,y) of elements of P there exists one least upper bound (or supremum):*

$x \vee y$ *is such that* $x \vee y \geq x$ *,* $x \vee y \geq y$ *and for any u,* $u \geq x$ *and* $u \geq y$ *implies* $u \geq x \vee y$

and one greatest lower bound (or infimum):

$x \wedge y$ *is such that* $x \wedge y \leq x$ *,* $x \wedge y \leq y$ *and for any l,* $l \leq x$ *and* $l \leq y$ *implies* $l \leq x \wedge y$

The following results show that when E and $\mathcal{L}$ are lattices, then the corresponding Galois poset G also is a lattice.

Proposition 1 (Galois lattice) *Let (E,$\subseteq$,$\mathcal{L}$,$\preceq$, int, ext) be a Galois connection between the lattices ($\mathcal{L}$,$\preceq$) and (E, $\subseteq$)*

The corresponding Galois poset G is a lattice, denoted as a Galois lattice, with as supremum:

($c1 \wedge_{\mathcal{L}} c2$, $ext \circ int(e1 \vee_E e2)$) and as infimum:

($int \circ ext(c1 \vee_{\mathcal{L}} c2)$, $e1 \wedge_E e2$)

Example 1. The lattice $\mathcal{L}$ is the set $P(A)$ of all subsets of a set of properties and its partial ordering $\preceq$ simply is the inclusion order on A. Then $c1 \preceq c2$ means that $c1$ is less specific than $c2$ (i.e. more general), and corresponds to the fact that all properties in $c1$ belongs to $c2$. The lattice E is the set $P(I)$ of all subsets of a set I of instances and its partial ordering $\subseteq$ is the inclusion order on I.

We consider each instance as the set of its properties. The extension function ext is defined as follows: $ext(c)$ is the subset of instances that "belong" to c, i.e. possessing all the properties of c. The intention function int is defined as follows : $int(e)$ is the subset of properties that belong to all the instances of e. Then, $\wedge_{\mathcal{L}}$ and $\wedge_E$ are intersection operators, when $\vee_{\mathcal{L}}$ and $\vee_E$ are union operators.

The instances of our example are represented in figure 1 where each line represents the intent of an instance of $I = \{i1, i2, i3, i4, i5, i6, i7, i8\}$ and each column represents the extent of a property of $A = \{t1, t2, t3, a3, a4, a5, a6, a7, a8\}$. Insert figure 1 about here. The corresponding Galois lattice is presented in figure 2. Insert figure 2 about here.

Let us now consider for instance the element $\{a4\}$ of $\mathcal{L}$, then $ext(\{a4\}) = \{i1, i3, i4\}$ and $int(\{i1, i3, i4\}) = \{a4, a6\}$. So $\{a4, a6\}$ is a closed element of $\mathcal{L}$. In G this closed element is paired with $ext(\{a4, a6\}) = ext(\{a4\}) = \{i1, i3, i4\}$.

	t1	**t2**	**t3**	**a3**	**a4**	**a5**	**a6**	**a7**	**a8**
i1	1			1	1		1		1
i2	1			1		1	1		
i3		1			1		1		1
i4		1			1		1	1	
i5		1		1			1		1
i6			1	1			1		1
i7			1	1			1		1
i8			1	1		1	1		1

Fig. 1. Instances description. $Tab(i, j) = 1$ if the j^{th} attribute belongs to the i^{th} example.

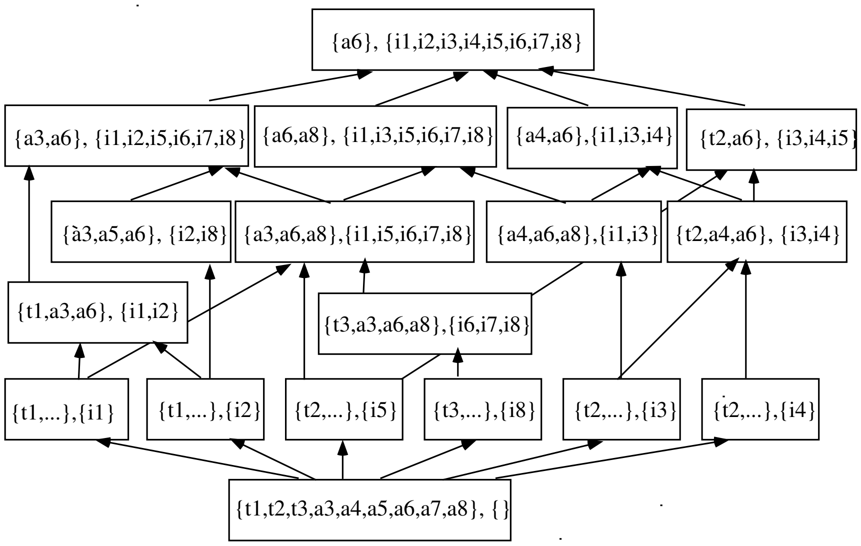

Fig. 2. The Galois Lattice corresponding to figure 1

Note that in the previous example, there is an implicit "belongs" function relying on the semantics of $\mathcal{L}$, such that $belongs(i, c)$ returns true whenever the instance i "belongs" to the element c of $\mathcal{L}$. Then $ext(c)$ returns the (maximal) subset of the instance set I the elements of which "belong" to c. Concerning the function int, note that $int(\{i1, i2, i3\})= int(\{i1\}) \vee_{\mathcal{L}} int(\{i2\}) \vee_{\mathcal{L}} int(\{i3\})$. So usually the intensional function int is built from its value on instances. Such a value $int(\{i\})$ is simply the most specific element of $\mathcal{L}$ which the instance i belongs to, and is denoted as the *abstract* of i.

2.2 Nested Galois posets and Galois lattices

The purpose of this section is to define a partial order on Galois posets, and Galois lattices. To define this order we start from its largest element corresponding to a pair of posets $\mathcal{L}$ and E, and we define a smaller Galois poset by applying a projection, i.e. a many-to-one application $proj$ from one of the poset (either E or $\mathcal{L}$) to itself. The image of $proj$ is a new poset that will define together with the other original poset a new Galois connection, and therefore a new Galois poset. This new Galois poset is a subset of the original Galois poset, and is stated as coarser (i.e. smaller) than the previous one. If again we see $\mathcal{L}$ as a concept language ordered by a "less-specific" partial ordering, and E as a set of possible extents of elements of $\mathcal{L}$, then $proj_{\mathcal{L}}$ projects various concepts on the same one, and is viewed as an operation that lowers the accuracy with which concepts are described. On the other hand, $proj_E$ projects various sets of instances on one set, and as a consequence, various concepts will have the same extent. Again, the following definition and properties are symmetrical (the role of E and $\mathcal{L}$ can be interchanged) but we still give the two variants of the definitions and properties in order to make easier the further use of these notions in the next section.

Definition 7 (Projection) *A projection $proj_P$ on a poset $(P, \leq)$ is a many-to-one application from P to P satisfying the following properties for all elements p, $p1$ and $p2$ of P:*

P1 - Idempotency : $proj_P(p) = proj_P(proj_P(p))$
P2 - Monotonicity : if $p1 \leq p2$ then $proj_P(p1) \leq proj_P(p2)$
P3 - Minimality : $proj_P(p) \leq p$

The following property states that projections can be composed:

Proposition 2 *Let $proj_1$ be a projection on the poset P, and $proj_2$ be a projection on the poset $proj_1(P)$, then $proj_2 \circ proj_1$ is a projection on P.*

Note that this property does not hold when $proj_2$ is a projection on P and is not stable on $proj_1(P)$, i.e. when for some p belonging to $proj_1(P)$, $proj_2(p)$ does not belong to $proj_1(P)$. This mean that we cannot just define projections on P and then freely compose them.

We use a projection to obtain a Galois poset coarser than a given Galois poset G. From now on, when considering a Galois correspondence $(E, \subseteq, \mathcal{L}, \preceq, int, ext)$, the corresponding Galois lattice is denoted as $G(int, ext)$.

Theorem 1 (Towards a Galois poset nested by extensional projection) *Let $(E, \subseteq, \mathcal{L}, \preceq, int, ext)$ be a Galois connection and let $proj_E$ be a projection on E. Let $E' = proj_E(E)$ and $ext' = proj_E \circ ext$. Then:*

1) $(E' \subseteq, \mathcal{L}, \preceq, int, ext')$ define a Galois connection.

2) If $(c = int(e), e)$ belongs to the Galois poset $G(int, ext)$ then $(proj_E(e), int \circ proj_E(e))$ belongs to the Galois poset $G(int, ext')$.
We will say that $G(int, ext')$ is nested in $G(int, ext)$ by extensional projection.

proof : see appendix.

Corollary 1 *If $(int(e), e)$ belongs to $G(int, ext')$ and $e = proj_E(e)$ then $(int(e), e)$ also belongs to the original Galois poset $G(int, ext)$.*

This means that $proj_E$ projects the Galois poset $G(int, ext)$ on the Galois poset $G(int, ext')$ and preserves every node of $G(int, ext)$ whose extent is unchanged by the projection.

Corollary 2 *If the Galois poset $G(int, ext')$ is nested in $G(int, ext)$ by extensional projection, the equivalence relation $Eqext'$ on $\mathcal{L}$ associated to $G(int, ext')$ is coarser than the equivalence relation $Eqext$ associated to $G(int, ext)$:*

if $ext(c1) = ext(c2)$ then $ext'(c1) = ext'(c2)$

In the same way the intensional projection defines a Galois connection.

Theorem 2 (Towards a Galois poset nested by intensional projection) *Let* $(E,\subseteq,\mathcal{L},\preceq, int, ext)$ *be a Galois connection and let* $proj_{\mathcal{L}}$ *be a projection of* $\mathcal{L}$. *Let* $\mathcal{L}' = proj_{\mathcal{L}}(\mathcal{L})$ *and* $int' = proj_{\mathcal{L}} \circ int$. *Then:*

1) $(E,\subseteq, \mathcal{L}', \preceq, int', ext)$ *define a Galois connection.*

2) If $(c, e=ext(c))$ *belongs to the Galois poset* $G(int, ext)$ *then* $(proj_{\mathcal{L}}(c), ext(proj_{\mathcal{L}}(c)))$ *belongs to the Galois poset* $G(int', ext)$.

We will say that $G(int', ext)$ *is nested in* $G(int, ext)$ *by intensional projection.*

Corollary 3 *If* $(c, ext(c))$ *belongs to* $G(int', ext)$ *and* $c = proj_{\mathcal{L}}(c)$ *then* $(c, ext(c))$ *also belongs to the original Galois poset* $G(int, ext)$.

Corollary 4 *If the Galois poset* $G(int', ext)$ *is nested in* $G(int, ext)$ *by intensional projection, the equivalence relation* $Eqint'$ *on* E *associated to* $G(int', ext)$ *is coarser than the equivalence relation* $Eqint$ *associated to* G *:*

if $int(e1) = int(e2)$ *then* $int'(e1) = int'(e2)$

Example 2. Let us consider a language $\mathcal{L}$ built from two attributes, namely $colour$ and $category$: $category$ has either value $t1$ or value $t2$; $colour$ has one of the following values : $blue$, red, $dark-blue$, $light-blue$, $dark-red$, $light-red$. These values are partially ordered in a hierarchy. The partial order on $\mathcal{L}$ is then such that, for instance $(colour = blue) \preceq (colour = darkblue) \preceq (colour = darkblue)\,\&(category = t1)$. Here $proj_{\mathcal{L}}$ consists in omitting to specify whether the colour is $dark$ or $light$. As a result $proj_{\mathcal{L}}((colour = blue)\,\&\,(category = t1))$ $= ((colour = blue)\,\&\,(category = t1))$ (unchanged). But this last term also is equal to $proj_{\mathcal{L}}((colour = darkblue)\,\&\,(category = t1))$ and to $proj_{\mathcal{L}}((colour = lightblue)\,\&\,(category = t1))$
An example of extensional projection is presented in section 3.7.

Projections define a partial order relation, that we call *nesting*, on Galois posets. This order corresponds to set inclusion on restrictions of the initial posets E and $\mathcal{L}$. Nesting is reflexive, transitive and antisymmetric[1].

We consider now a projection as a correspondence defined on Galois posets and we denote projected Galois posets as either $proj_E(G)$ or $proj_{\mathcal{L}}(G)$. This correspondence has the following property.

Proposition 3 *Let* $proj_E$ *be an extensional projection of* $G(int, ext)$ *and* $proj_{\mathcal{L}}$ *be an intensional projection of* $G(int, ext)$ *then:*

$proj_E(proj_{\mathcal{L}}(G)) = proj_{\mathcal{L}}(proj_E(G))$

The next theorem is of great practical interest since it gives a way to come back from a coarser Galois poset to a fine-grained one. The practical use of these

[1] Note that the partial order on $\mathcal{L}$ is not syntactic. For instance $a\&b$ and $b\&a$ are two representations of the same element of $\mathcal{L}$. This allows the relation $\preceq$ to be antisymmetric (i.e. $c \preceq c'$ and $c' \preceq c$ implies $c = c'$) and so to be a partial order.

nestings will be to first compute coarse Galois posets and then refine them (see section 4).

Theorem 3 (backward to a Galois poset from extensional projection) *Let (c, e'=ext'(c)) be an element of the poset $G(int, ext')$ nested in $G(int, ext)$ by $proj_E$, then: (c, e=ext(c)) is an element of $G(int, ext)$. As e'=$proj_E$(e) we will denote (c,e) as $proj_E^{-1}$(c, e').*

Proof: $proj_E(ext(c)) = proj_E \circ ext(int(e')) = e'$ as e' is closed for $ext \circ int$, so $e'= proj_E(ext(c)) \subseteq ext(c)$ (from P3) and $c = int(e') \succeq int(ext(c)$ (from C1). As $c \preceq int(ext(c)$ (from C3), then $c = int(ext(c)$, and so $(e = ext(c), c)$ belongs to G.

The same theorem holds for intensional projection.

Theorem 4 (backward to a Galois poset from intensional projection) *Let (c'=int'(e), e) be an element of the poset $G(int', ext)$ nested in $G(int, ext)$ by $proj_{\mathcal{L}}$ then: (c=int(e), e) is an element of $G(int, ext)$. As c'=$proj_{\mathcal{L}}$(c) we will denote (c, e) as ${proj_{\mathcal{L}}}^{-1}$(c', e).*

As a consequence, starting with coarse posets, we can define a coarse Galois poset and then come back to corresponding nodes in a more fine-grained Galois poset using either extensional inverse projection or intensional inverse projection. Unfortunately when composing intensional and extensional inverse projections, the resulting node will depend on the order on which we apply these inverse projections. In other words, $proj\,\mathcal{L}^{-1} \circ proj E^{-1}(e', c') \neq proj E^{-1} \circ proj\,\mathcal{L}^{-1}$ (e',c'). This simply means that when we intend to use both extensional and intensional inverse projection we have to chose the path that leads to an antecedent (e, c) of (e', c') from $proj_{\mathcal{L}}$ and $proj_E$.

When a poset is a lattice, the projected poset also is a lattice.

Proposition 4 (Nested Lattice) *Let P be a lattice with the supremum operator $\vee_P$ and the infimum operator $\wedge_P$ and let $proj_P$ be a projection. Then $proj_P(P)$ is also a lattice with the infimum operator $\wedge$ and supremum operator $\vee$ defined as follows:*

p1 $\wedge$ p2 = $proj_P$ (p1 $\wedge_P$ p2)
p1 $\vee$ p2 = p1 $\vee_P$ p2

Corollary 5 (Nested Galois lattice) *A Galois poset G' nested in a Galois lattice G is a Galois lattice.*

Proof: As $proj_E(E)$ (or $proj_{\mathcal{L}}(\mathcal{L})$) is a lattice then the galois poset G' is a Galois lattice (see previous section).
Note that projections can be endomorphisms i.e. $proj(p1 \wedge p2) = proj(p1) \wedge proj(p2))$. In this case let $p1$ and $p2$ be projected elements then $(proj(p1 \wedge p2) = proj(p1) \wedge proj(p2) = p1 \wedge p2$. This happens to be true for the languages investigated in the next section and it will ease computation of nested Galois lattices.

3 Languages, Extensions and Nested Galois Lattices

In our approach, nested Galois lattices are used in order to limit the complexity of a clustering step made on numerous data. The data (i.e. instances) that we consider are described using a set of attributes that can be multi-valued. Furthermore, each instance is labelled by a basic type. As we will see, the projection defined on the extensional part of a Galois lattice node exploits the fact that the instances are typed and the projection defined on the intensional part of a node exploits the fact that each instance is described using only part of the set of attributes. We first present the language of instances, the language of classes $\mathcal{L}_3$ and then the coarser language $\mathcal{L}_2$. Then we discuss nesting order of the Galois lattices corresponding to $\mathcal{L}_3$ and $\mathcal{L}_2$. Then α-extension is defined and the notion of extension in frequent Galois lattices is discussed.

3.1 Definitions and notations

We call *class* a set of instances. Before defining the language of instances and the two languages of classes $\mathcal{L}_2$ and $\mathcal{L}_3$ that we use to describe classes over those data, we provide some notations and basic definitions. Given a language of instances, a language of classes $\mathcal{L}$ defines the *terms* that are allowed. A *membership relation*, denoted by $isa_{\mathcal{L}}$, establishes the necessary connection between a given language of instances and an associated language of classes $\mathcal{L}$.

Definition 8 (Extension of a term) *Let Inst be a set of instances, and c a $\mathcal{L}$ term. The extent of c w.r.t Inst is the following set:*

$$ext_{Inst}(c) = \{i \in Inst \mid i \, isa_{\mathcal{L}} \, c\}$$

The subsumption relation is a partial order relation between terms, induced by the inclusion relation between term extents.

Definition 9 (Subsumption between terms) *Let c_1 and c_2 be two $\mathcal{L}$ terms. c_1 subsumes c_2, denoted $c_1 \preceq_{\mathcal{L}} c_2$, iff for every set Inst of instances, $ext_{Inst}(c_2) \subseteq ext_{Inst}(c_1)$.*

In sections 3.4, and 3.3, we will provide a constructive characterization of subsumption for the two languages of classes that we consider.
The notions of *abstraction* and *least common subsumer* are the basis to describe gathered instances in the Galois lattice. The *abstraction* of an instance in a language of classes $\mathcal{L}$ corresponds, when it exists, to the most specific term in $\mathcal{L}$ which it is an instance of.

Definition 10 (Abstraction of an instance) *Let i be an instance, the $\mathcal{L}$ term c is an* abstraction *of i in $\mathcal{L}$ (for short $c = abs_{\mathcal{L}}(i)$) iff*
1. $i \, isa_{\mathcal{L}} \, c$, and
2. if d is a term such that $i \, isa_{\mathcal{L}} \, d$, then $d \preceq_{\mathcal{L}} c$.

Definition 11 (Least Common Subsumer) *Let $c_1, \ldots, c_n$ be terms in $\mathcal{L}$. The term c is a* least common subsumer *of $c_1, \ldots, c_n$ in $\mathcal{L}$,$c = lcs_{\mathcal{L}}(c_1, \ldots, c_n)$, iff*
1. $c \preceq_{\mathcal{L}} c_i$ for all $1 \leq i \leq n$, and
2. if d is a term satisfying $d \preceq_{\mathcal{L}} c_i$ for all $1 \leq i \leq n$, then $d \preceq_{\mathcal{L}} c$

A *class description* is a *term* of the language of classes which is intended to represent in an abstract and concise way the properties that are common to the set of the instances of the class. When the language of instances and the language of classes are such that the abstraction of an instance is unique, so as the least common subsumer of classes, the *class description of a set of instances* can then be defined as follows.

Definition 12 (Class description of a set of instances) *Let e=$\{i_1, \ldots, i_n\}$ be a set of instances. The $\mathcal{L}$ class description of e denoted $desc_{\mathcal{L}}(e)$ is: $desc_{\mathcal{L}}(e) = lcs_{\mathcal{L}}(abs_{\mathcal{L}}(i_1), \ldots, abs_{\mathcal{L}}(i_n))$.*

3.2 The Language of instances

An instance is described as a set of (Attribute, Value) pairs. The attributes used for describing the instances may vary from an instance to another. In addition, an attribute can be multi-valued. Furthermore, each instance is typed (i.e. labelled by a basic type). For instance a product appearing in an electronic catalog could be typed as a kitchen furniture.

Definition 13 (Terms of $\mathcal{L}_1$) *Let $\mathcal{B}$ be a finite set of basic types, $\mathcal{A}$ a finite set of* attributes, *and $\mathcal{V}$ a set of values.*
For each attribute att, let domain(att) be a subset of $\mathcal{V}$, called its domain. *A term of $\mathcal{L}_1$ is of the form:*

$$\{t, att_1 = V_1, \ldots, att_n = V_n\}$$

where $t \in \mathcal{B}$, $\forall i \in [1..n]$, $att_i \in \mathcal{A}$ and $V_i \subseteq domain(att_i)$.

For example, we can find a product in the C/Net catalog (see section 4.5), whose $\mathcal{L}_1$ description is:

$\{Removablediskdrive,\ CD/DVD/Type{=}\{CDRW\},$
$StorageRemovableType{=}\{SuperDisk\}, Compatibility{=}\{MAC, PC\}\}$

In the following, we will consider that the type t of a $\mathcal{L}_1$ description is a boolean attribute.

3.3 The language of classes $\mathcal{L}_3$

The language of classes $\mathcal{L}_3$ makes possible to restrict the possible values of an attribute ; it enables to distinguish the number of values of an attribute through different suffixes ($*$, $+$, ? , ϵ) whose notation is inspired by the one used in XML for describing document type definitions (DTDs), and whose formal semantics corresponds to standard description logics constructors.

Definition 14 (Syntax of $\mathcal{L}_3$) *A $\mathcal{L}_3$ term (of size n) is a tuple*

$$\{att_1^{suff_1} : V_1, \ldots, att_n^{suff_n} : Vn\}$$

where $\forall i \in [1..n]$, $att_i \in \mathcal{A}$, $V_i \subseteq domain(att_i)$, *and* $suff_i \in \{*, +, ?, \epsilon\}$

The meaning of $att_i^{suff} : V_i$ in a term c is the following:
$att_i^* : V_i$: the attribute is optional (it can be absent in a description of an instance of c) and it can be mono or multi valued. If the attribute is present in the description of an instance of c, all values must belong to V_i.
$att_i^+ : V_i$: each instance of c has at least one value for this attribute and all values must belong to V_i.
$att_i^? : V_i$: the attribute is monovalued and optional, if the attribute is present, its values must belong to V_i.
$att_i^\epsilon : V_i$: each instance of c has one and only one value for the attribute and this value must belong to V_i
att_i is not in the description of c: no instance of c has att_i in its description.

Note that suffixes are ordered, the generality order of suffixes is the following: $att_i^+ \prec att_i^*$, $att_i^? \prec att_i^*$, $att_i^\epsilon \prec att_i^+$, $att_i^\epsilon \prec att_i^?$, $not(att_i)^2 \prec att_i^?$.
This generality order is used during the subsumption computation and the least common subsumer computation (e.g. the lcs of $+$ and ϵ is $+$)[3].
The following definition formalizes the membership relation between an instance and a term of $\mathcal{L}_3$ thus providing a clear semantics to $\mathcal{L}_3$, and in particular to the suffixes of the attributes.

Definition 15 (Membership relation for $\mathcal{L}_3$) *Let i be an instance description in $\mathcal{L}_1$. Let c be a $\mathcal{L}_3$ term. i is an instance of c iff every attribute in i appears in c and for every $att^{suff} : V$ appearing in c,*
- when $suff$=$*$, if there exists V' s.t att=$V' \in i$, then $V' \subseteq V$,
- when $suff$=$+$, there exists $V' \subseteq V$ s.t att=$V' \in i$,
- when $suff$=$?$, if there exists V' s.t att=$V' \in i$, then V' is a singleton and $V' \subseteq V$,
- when $suff$=ϵ, there exists V' singleton s.t $V' \subseteq V$ and att=$V' \in i$.

The following propositions state the main properties of $\mathcal{L}_3$. Their proofs follow from results in tractable description logics where structural subsumption is complete.

[2] *not* is not a connective of the language but we use it here to mark the absence of the attribute att_i.

[3] Note that different terms of $\mathcal{L}_3$ can be semantically equivalent when they express exactly the same constraints (e.g.: $colour^+ : \{blue\}$ is equivalent to $colour^\epsilon : \{blue\}$). We described in [25] a normalization process leading to construct from any term of $\mathcal{L}_3$ a normalized term (e.g. normalization($colour^+ : \{blue\}$) = $colour^\epsilon : \{blue\}$)). The propositions that follow are sound if we consider that terms (or class descriptions) are normalized.

Proposition 5 (Characterization of subsumption in $\mathcal{L}_3$) *Let c_1 and c_2 be two $\mathcal{L}_3$ terms. $c_2 \preceq_{\mathcal{L}_3} c_1$ iff all the attributes appearing in c_1 appear also in c_2 and for every pair $att^{suff} : V$ appearing in c_2,*
- when $suff$=$*$, if there exists $att^{suff'} : V' \in c_1$, then $V' \subseteq V$,
- when $suff$=$+$, there exists $V' \subseteq V$ s.t $att^{+} : V' \in c_1$ or $att^{\epsilon} : V' \in c_1$
- when $suff$=$?$, if there exists $att^{suff'} : V' \in c_1$, then $suff'$=$?$ or $suff'$=ϵ, and $V' \subseteq V$,
- when $suff$=ϵ, there exists V' s.t $V' \subseteq V$ and $att^{\epsilon} : V' \in c_1$.

We note here that the more general term that can be defined in $\mathcal{L}_3$ is:
$\{att_1^* : domain(att_1), \ldots, att_n^* : domain(att_n)\}$ where $\mathcal{A} = \{att_1, \ldots, att_n\}$

Proposition 6 (Characterization of abstraction in $\mathcal{L}_3$) *Let $\{att_1{=}V_1, \ldots, att_n{=}V_n\}$ be an instance description in $\mathcal{L}_1$. Its abstraction in $\mathcal{L}_3$ is unique: $abs_{\mathcal{L}_3} = \{att_1^{suff_1} : V_1, \ldots, att_n^{suff_n} : V_n\}$, where $\forall i \in [1..n]$, if $\mid V_i \mid \geq 2$ then $suff_i$=$+$ else $suff_i$=ϵ.*

Proposition 7 (Characterization of lcs in $\mathcal{L}_3$) *Let $c_1, \ldots, c_n$ be n $\mathcal{L}_3$ terms. Let A be the set of attributes belonging to at least one term c_i. $c_1, \ldots, c_n$ have a unique least common subsumer $lcs(c_1, \ldots, c_n)$ in $\mathcal{L}_3$, whose description is characterized as follows:*

- *for every attribute $att \in A$, let V be the union of the sets of values associated with att in the class descriptions c_i's: $V = \bigcup_1^n \{v \in V_i \mid att^{suff} : V_i \in c_i\}$.*
  - *$att^{\epsilon} : V \in lcs(c_1, \ldots, c_n)$ iff $att^{\epsilon} : V_i \in c_i \ \forall i \in [1..n]$.*
  - *$att^{?} : V \in lcs(c_1, \ldots, c_n)$ iff*
    ($\forall i \in [1..n]$ $att^* : V_i \notin c_i$ and $att^+ : V_i \notin C_i$), and
    - *either $\exists i \in [1..n]$ s.t. $att^? : V_i \in c_i$,*
    - *or $\exists i \in [1..n]$ s.t. $att^s : V' \notin c_i$ for any s.*
  - *$att^* : V \in lcs(c_1, \ldots, c_n)$ iff*
    - *either $\exists i \in [1..n]$ s.t. $att^* : V_i \in c_i$,*
    - *or $\exists i \in [1..n]$ s.t. $att^+ : V_i \in c_i$, and $\exists j \in [1..n]$ s.t. $att^? : V_j \in c_j$ or $att^{s'} : V' \notin c_j$ for any suffix s'.*
  - *$att^+ : V \in lcs(c_1, \ldots, c_n)$ iff $\exists i \in [1..n]$ s.t. $att^+ : V_i \in c_i$ and $\forall j \in [1..n]$, $att^+ : V_j \in c_j$ or $att^{\epsilon} : V_j \in c_j$*

Proof of this proposition is given in appendix.

Example 3. The product described in 3.2 is an instance of the $\mathcal{L}_3$ term c_1:

$\{RemovableDiskDrive^{\epsilon} : \{true\}, CD/DVD/ReadSpeed^{?}:\{20x, 32x, 24x\}, CD/DVD/Type^{\epsilon}:\{CDROM, CDRW\}, Compatibility^{+}:\{MAC, PC\}, StorageRemovableType^{\epsilon}:\{SuperDisk, ZIP, JAZ\}\}$

c_1 represents the set of products that have in their description *(i)* necessarily the monovalued and boolean attribute *RemovableDiskDrive* whose value must be *true*, *(ii)* possibly the monovalued attribute $CD/DVD/ReadSpeed$,

whose value belongs to the set $\{20x, 32x, 24x\}$, *(iii)* necessarily the attribute $CD/DVD/Type$, which is monovalued and takes its value in the set $\{CDROM, CDRW\}$, *(iv)* necessarily the attribute $Compatibility$, which can be multivalued and takes its value(s) in the set $\{MAC, PC\}$, *(v)* necessarily the attribute $StorageRemovableType$, which is monovalued and takes its value in the set $\{SuperDisk, ZIP, JAZ\}$.
Let c_2 be the following $\mathcal{L}_3$ term:

$$\{Compatibility^{*}\!:\!\{PC, Unix\}, StorageRemovableType^{\epsilon}\!:\!\{DAT\}, \\ CompressedCapacity^{\epsilon} : \{8, 24, 32, 70\}\}$$

$lcs(c_1, c_2)=$
$$\{RemovableDiskDrive^{?} : \{true\}, CD/DVD/ReadSpeed^{?}\!:\!\{20x, 32x, 24x\}, \\ CD/DVD/Type^{?}\!:\!\{CDROM, CDRW\}, Compatibility^{*}\!:\!\{MAC, PC, Unix\}, \\ StorageRemovableType^{\epsilon}\!:\!\{SuperDisk, ZIP, JAZ, DAT\}, \\ CompressedCapacity^{?}\!:\!\{8, 24, 32, 70\}\}$$

Proposition 8 *The poset ($\mathcal{L}_3$, $\preceq_{\mathcal{L}_3}$) is a lattice.*

Proof:
$\forall$ pair (x,y) of terms of $\mathcal{L}_3$, there exists one least upper bound x $\otimes$ y such that x $\preceq$ x $\otimes$ y and for any u, x $\preceq$ u and y $\preceq$ u implies x $\otimes$ y $\preceq$ u [25].

$\forall$ pair (x,y) of descriptions in $\mathcal{L}_3$, there exists one greatest lower bound lcs(x,y)such that lcs(x,y)$\preceq_{\mathcal{L}_3}$x, lcs(x,y)$\preceq_{\mathcal{L}_3}$ y and for any l, l$\preceq_{\mathcal{L}_3}$x and l$\preceq_{\mathcal{L}_3}$y implies l$\preceq_{\mathcal{L}_3}$lcs(x,y). The proof is straightforward with regard to the definition of the lcs which is unique and stable in $\mathcal{L}_3$.

3.4 The language of classes $\mathcal{L}_2$

The language of classes $\mathcal{L}_2$ only distinguishes instances when they have a different structure, i.e. if they are described using a different set of attributes. Thus, for instance we will first observe whether data are described as coloured or not, using $\mathcal{L}_2$, and then observe whether their colour is the same using $\mathcal{L}_3$.

Definition 16 (Syntax of $\mathcal{L}_2$) *A $\mathcal{L}_2$ term (of size n) is a tuple of attributes $\{att_1, \ldots, att_n\}$, where $\forall i \in [1..n]$, $att_i \in \mathcal{A}$.*

The connection between the language of instances $\mathcal{L}_1$ and the language of classes $\mathcal{L}_2$ is based on the following definition of the membership relation.

Definition 17 (Membership relation for $\mathcal{L}_2$) *Let i be an instance description in $\mathcal{L}_1$. Let c be a $\mathcal{L}_2$ term: i is an instance of c iff every attribute appearing in c also appears in i.*

The following proposition, whose proof is straightforward, characterizes subsumption, least common subsumer and abstraction in $\mathcal{L}_2$.

Proposition 9 (Properties of $\mathcal{L}_2$) • *Let c_1 and c_2 be two $\mathcal{L}_2$ terms. $c_1 \preceq_{\mathcal{L}_2} c_2$ iff every attribute of c_1 is also an attribute of c_2.*

• *Let* $\{att_1 = V_1, \ldots, att_n = V_n\}$ *be an instance description in* $\mathcal{L}_1$. *Its abstraction in* $\mathcal{L}_2$ *is unique: it is* $\{att_1, \ldots, att_n\}$.

• *Let* $c_1, \ldots, c_n$ *be* n $\mathcal{L}_2$ *terms. Their least common subsumer is unique: it is made of the set of attributes that are common to all the* c_i*'s.*

3.5 The original Galois lattice defined on $\mathcal{L}_3$

Let I be the set of instances of a considered domain. For a sake of simplicity, when the extension of a term c is computed on the domain, we will denote this extension $ext(c)$ rather than $ext_I(c)$. Let $desc_{\mathcal{L}_3}$ be the correspondence defined according to definition 13.
The language of classes $\mathcal{L}_3$ can be used to define a Galois lattice G($desc_{\mathcal{L}_3}$,ext).

Proposition 10 *($\mathcal{P}(I)$, $\subseteq$, $\mathcal{L}_3$,$\preceq_{\mathcal{L}_3}$,$desc_{\mathcal{L}_3}$,ext) defines a Galois connection.*

Proof of this proposition is given in the appendix. This proposition and the fact that the posets ($\mathcal{L}_3$, $\preceq_{\mathcal{L}_3}$) and ($\mathcal{P}(I)$,$\subseteq$) are lattices result in the following corollary.

Corollary 6 *The Galois poset G($desc_{\mathcal{L}_3}$,ext) corresponding to the Galois connection ($\mathcal{P}(I)$, $\subseteq$, $\mathcal{L}_3$,$\preceq_{\mathcal{L}_3}$,$desc_{\mathcal{L}_3}$,ext) is a Galois lattice.*

In our approach, G($desc_{\mathcal{L}_3}$,ext) is the most fine-grained lattice: all lattices described in what follows are nested in G($desc_{\mathcal{L}_3}$,ext).

3.6 A nested Galois lattice : from $\mathcal{L}_3$ to $\mathcal{L}_2$

We can easily see that $\mathcal{L}_2$ is a sublanguage of $\mathcal{L}_3$. Actually, each term of $\mathcal{L}_2$ can be rewritten in $\mathcal{L}_3$. Let c={att1,..., attm} be a term of $\mathcal{L}_2$. Then, c is equivalent to the $\mathcal{L}_3$ term: {att1$^+$:domain(att1), ..., attm$^+$:domain(attm), attk*:domain(attk), ..., attn*:domain(attn)}, where {attk*:domain(attk), ..., attn*:domain(attn)} are the attributes that belongs to $\mathcal{A}$ but that do not appear in c.

We define an intensional projection on $\mathcal{L}_3$ called $fl2$ which gives for a term c in $\mathcal{L}_3$ the most specific term in $\mathcal{L}_2$ subsuming c :

Proposition 11 (the intensional projection $fl2$) *Let c be a class defined in* $\mathcal{L}_3$. *The application fl2 defined as follows :*
For each attribute att^{suff} : V *of c,*
if suff=+ or suff=ϵ then att$^+$: domain(att) $\in$ $fl2(c)$,
if suff=? or suff= then att*: domain(att) $\in$ $fl2(c)$,*
$\forall$ *att* $\in$ A *such that att* $\notin$ *c, att*: domain(att)* $\in$ $fl2(c)$,
is an intensional projection of $\mathcal{L}_3$.

Proof :
Identity on $fl2$: if c belongs to $fl2(\mathcal{L}_3)$, i.e. $\mathcal{L}_2$, then $c = fl2(c)$?
c belongs to $\mathcal{L}_2$ iff $\forall$ att^{suff}:V $\in$ c, we have V=domain and suff=+ or suff=*

and $\forall$ att $\in \mathcal{A}$, att $\in$ c. By definition of $fl2$, we can see that in such cases, $c = fl2(c)$.
Monotonicity: if c1 $\preceq$ c2, then $fl2(c1) \preceq fl2(c2)$?
Let us suppose that there exists c1 and c2 such that $c1 \preceq c2$ and $fl2(c1) \not\preceq fl2(c2)$.
We know that $\forall$ att $\in \mathcal{A}$, att $\in$ fl2(c1) and att $\in$ fl2(c2) (by definition of $fl2$). Besides, $\forall$ $att^{suff} : V_{c1} \in$ c1 such that $att^{suff} : V_{c2} \in$ c2, we know that $V_{c2} \subseteq V_{c1}$. We also know that the suffixes of the attributes that appear in $fl2(c1)$ or $fl2(c2)$ are either * or +.
So, $fl2(c1) \not\preceq fl2(c2)$ implies that $\exists$ $atti^{+} \in fl2(c1)$ such that $atti^{*} \in$ fl2(c2) (by definition of the subsumption).
If $atti^{+} \in fl2(c1)$ then $atti^{\epsilon}$ or $atti^{+} \in$ c1 (by definition of $fl2$).
If $atti^{*} \in fl2(c2)$ then $atti^{?} \in$ c2 or $atti^{*} \in$ c2 or $atti \notin$ c2 (by definition of $fl2$).
These properties cannot be true since in that cases, c1 $\not\preceq$ c2.
Minimality: $fl2(c) \preceq c$?
The proof is straightforward with regard to the definition of fl2 and to the definition of subsumption.

Example 4. Let $\mathcal{A} =$ {colour, form, size}, let $\{blue, red, yellow\}$ be the domain of colour, let $\{square, circle\}$ be the domain of form, let $\{12, 24, 32\}$ be the domain of size and let c be the $\mathcal{L}_3$ description $\{colour^{\epsilon} : \{blue, red\}, form^{*} : \{square\}\}$. Then, we have :
$fl2(c) = \{colour^{+} : \{blue, red, yellow\}, form^{*} : \{square, circle\}, size^{*} : \{12, 24, 32\}\}$ and this $\mathcal{L}_3$ description is equivalent to the $\mathcal{L}_2$ description: $\{colour\}$.

Following theorem 2 and corrolary 5 in section 2.2:

Proposition 12 *($\mathcal{P}(I)$,$\subseteq$, $\mathcal{L}_2$, $\preceq$, $desc_{\mathcal{L}_2}$,ext) defines a Galois connection whose corresponding Galois lattice G($desc_{\mathcal{L}_2}$,ext) is nested in G($desc_{\mathcal{L}_3}$,ext).*

3.7 A Nested Galois lattice : from extensions to Alpha extensions

Now, the idea is to use the types that are associated to each data in order to aggregate types (i.e basic classes) instead of aggregating instances. The data are partitioned according to their label: for each type t (for example the type *Removablediskdrive*), we create a *basic class* whose set of instances is the set of data of type t.

Example 5. For a sake of simplicity, the examples use the language of classes $\mathcal{L}_2$. Let $I =$ {i1,i2,i3,i4,i5,i6,i7,i8} be the set of instances such that their abstraction in $\mathcal{L}_2$ is represented in figure 1. Let us assume that the attributes $t1, t2, t3$ express the types of the instances.
Then, we create a basic class BC1 = {i1,i2} whose class description is:
$desc_{\mathcal{L}2}(BC1)$ = lcs($abs_{\mathcal{L}2}(i1)$,$abs_{\mathcal{L}2}(i2)$) = {t1,a3,a6}
We create a basic class BC2 = {i3,i4,i5} whose class description is:
$desc_{\mathcal{L}2}(BC2)$ = {t2,a6}
We create a basic class BC3 = {i6,i7,i8} whose class description is:

$desc_{\mathcal{L}2}(BC3)$ = {t3,a3,a6,a8}

The notions described below are used in order to define Galois lattices where nodes aggregate parts of basic classes which are bigger than a certain proportion α. The first one is the *alpha satisfaction* which allows us to know whether at least α % of a set of instances belong to the extent of a term.

Definition 18 (Alpha satisfaction) *Let α be a number belonging to [0,100]. Let $e=\{i_1, \ldots, i_n\}$ be a set of instances and T be a term of a class language $\mathcal{L}$.*
e alpha satisfies T ($e\ sat_\alpha\ T$) iff $\mid ext_e(T) \mid \ \geq \frac{|e|.\alpha}{100}$

Since the *alpha* satisfaction is defined according to a set of instances and a term of a language of classes, we can use it to check whether at least α % of a basic class satisfies a term of $\mathcal{L}_2$ or $\mathcal{L}_3$ and add this constraint to the classical membership relation between instances and terms. We call this notion the *alpha membership relation.*

Definition 19 (Alpha membership relation) *Let I be a set of instances, $\mathcal{BC}$ be a finite set of disjoint basic classes such that $\mathcal{BC}$ represents a partition of I[4], and a function BCl: $I \rightarrow \mathcal{BC}$ which returns from an instance the basic class of the instance ($BCl(i)$ = set of instances representing the basic class of i). Let i be an instance description in $\mathcal{L}_1$. Let T be a $\mathcal{L}$ term.*

$$i\ isa_\alpha\ T\ iff\ i\ isa\ T\ and\ BCl(i)\ sat_\alpha\ T$$

Example 6. Let I be the set of instances used in the example 5. Let T = {a6,a8} be a term of $\mathcal{L}_2$.
$ext_I(T) = \{i3, i5, i6, i7, i8\}$.
$ext_{BC1}(T) = \{i1\}$, $ext_{BC2}(T) = \{i3, i5\}$, $ext_{BC3}(T) = \{i6, i7, i8\}$
$BC3\ sat_{100}\ T$ since 100 % of the instances of $BC3$ belongs to the extent of T.
BC1 and BC2 do not $sat_{100}\ T$. $BC2\ sat_{60}\ T$ since $\mid ext_{BC2}(T) \mid \ \geq \frac{|BC2|.60}{100}$.
BC1 does not $sat_{60}\ T$.
Finally i3 $isa_{60}\ T$ and i5 $isa_{60}\ T$. However this is not the case of i4 that belongs to the same basic class BC2 as i3 and i5.

Note that the *alpha membership relation* is equivalent to the classical *membership relation* when α is 0 and that when α is 100, an instance i belongs to a term T if and only if the whole basic class of i belongs to T.

Finally, we use the *alpha membership relation* to define the following new notion of extension that will be used in the next section in order to obtain different and more flexible Galois Lattices.

[4] This constraint of being a partition could be removed. If it is the case, we obtain a different definition of α membership relation that could be for instance "at least one basic class of the instance is alpha satisfied".

Definition 20 (Alpha extension of a term) *Let I be a set of instances, $\mathcal{BC}$ be a finite set of basic classes, and T a $\mathcal{L}$ term. The extension of T w.r.t I is the following set:*

$$ext^{\alpha}(T) = \{i \in I \mid i \ \ isa_{\alpha} \ \ T\}$$

Example 7. In our example, $ext(T)$=$\{i1, i3, i5, i6, i7, i8\}$, $ext^{60}(T)$={i3,i5,i6,i7,i8}.

We use ext^{α} to define Galois lattices nested in G($desc_{\mathcal{L}}$,ext). Before, we need to define $fext_{\alpha}$ an extensional projection on E such that $fext_{\alpha} \circ ext$ is equivalent to ext^{α}.

Proposition 13 (the extensional projection $fext_{\alpha}$) *Let e be an element of $\mathcal{P}(I)$, let i be an instance of e, the application $fext_{\alpha}$ defined as follows :*
$i \in fext_{\alpha}(e)$ iff $\mid e' \mid \ \geq \frac{|BCl(i)|.\alpha}{100}$ where $e' = \{j\in\ e \mid j$ belongs to $BCl(i)\}$, is an extensional projection of $\mathcal{P}(I)$.

Proof:
Identity on $fext_{\alpha}$: if e belongs to $fext_{\alpha}(\mathcal{P}(I))$ then $e = fext_{\alpha}(e)$?
if e belongs to $fext_{\alpha}(\mathcal{P}(I))$ then $\forall$ i $\in$ e, we know that if e' = {j $\in$ e | j belongs to $BCl(i)\}$ then $\mid e' \mid \ \geq \frac{|BCl(i)|.\alpha}{100}$. So, $e = fext_{\alpha}(e)$.
Monotonicity: if e1 $\subseteq$ e2, then $fext_{\alpha}(e1) \preceq fext_{\alpha}(e2)$?
Let $e1' = fext_{\alpha}(e1)$. We know that e1' $\subseteq$ e1 by definition of $fext_{\alpha}$. So, e1' $\subseteq$ e2. By definition of $fext_{\alpha}$, e1' $\subseteq$ $fext_{\alpha}(e2)$.
Minimality: $fext_{\alpha}(e) \subseteq e$? (straightforward w.r.t the definition of $fext_{\alpha}$).

Following theorem 2 and corollary 5 in section 2.2:

Proposition 14 *($fext_{\alpha}(P(I))$,$\subseteq$, $\mathcal{L}$, $\preceq$, $desc_{\mathcal{L}}$,ext^{α}) defines a Galois connection whose corresponding galois lattice G($desc_{\mathcal{L}}$,ext^{α}) is nested in G($desc_{\mathcal{L}}$,ext).*

Figure 3 shows the Galois lattice G($desc_{\mathcal{L}_{\in}}$,ext^{100}) of example 5 and should be compared to the original Galois lattice G($desc_{\mathcal{L}_{\in}}$,ext) shown in figure 2. Insert figure 3 about here.

Finally the following properties order the Galois lattices associated to α-extensions.

Proposition 15 (The total order of extensional projections $fext_{\alpha}$) *For every pair (α,α') such that $\alpha > \alpha'$:*
1) $fext_{\alpha}$ is a projection of $fext'_{\alpha}(\mathcal{P}(I))$
2) $\forall$ $e \in E$, $fext_{\alpha}(e)$= $fext_{\alpha}(fext_{\alpha'}(e))$

Corollary 7 (The nesting order related to α-extensions) *For every pair (α,α')such that $\alpha > \alpha'$: G($desc_{\mathcal{L}}$,ext^{α}) is nested in G($desc_{\mathcal{L}}$,$ext^{\alpha'}$).*

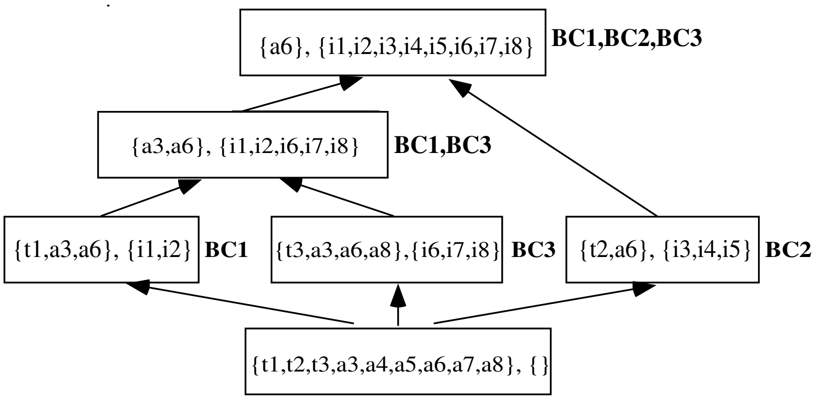

Fig. 3. Galois lattice of example 5 with ext^{100}

This order will be used in ZooM to stepwise increase the constraint associated to basic types on the clusters hierachy. As we will see, ZooM uses inverse projection, and so begins with some coarse Galois lattice G($desc_{\mathcal{L}}$,ext^{α}), then will locally refine this lattice by increasing α to a greater α' thus increasing the influence of basic classes on the concept organization.

3.8 Frequent Galois lattices

Another extensional projection can be defined in order to derive a frequent Galois lattice [32] from a Galois lattice.

An element (c, e) of a Galois lattice is in the frequent Galois lattice iff $\frac{|e|}{|I|}$ is greater or equal to a threshold s. This additional constraint also corresponds to an extensional projection.

Proposition 16 (Extensional projection $ffreq$) *Let E be a part of $\mathcal{P}(I)$ containing $\emptyset$, the correspondence $ffreq$ defined as follows :*
$\forall\ e \in E$, if $\frac{|e|}{|I|} < s$ then $ffreq(e) = \emptyset$ else $ffreq(e) = e$
is a projection of E.

Corollary 8 *Let E be a part of $\mathcal{P}(I)$ containing $\emptyset$, and let (E,$\subseteq$, $\mathcal{L}$, $\preceq$, $desc_{\mathcal{L}}$, ext) be a Galois connection, then G($desc_{\mathcal{L}}$, $ffreq \circ ext$) is nested in G($desc_{\mathcal{L}}$, ext).*

This last result means that $ffreq$ can be applied after any extensional projection. As Zoom works from coarse to fine-grained Galois lattices, the first Galois lattice to be constructed will be a frequent Galois lattice that will further be locally refined by applying extensional (and intensional) inverse projections.

4 ZooM

ZooM is a system devoted to the construction of Galois lattices at different levels of abstraction. The main goal of this system is to organize into clusters large sets of multivalued data while remaining in control the complexity of the construction tasks. This is done thanks to a two-step clustering method which first builds a coarse lattice and then refines a small subpart of the coarse lattice delimited by two nodes choosen by a user (we call this subpart the bounded coarse lattice). The result of the refinement step is a lattice in which the bounded coarse lattice is nested according to the definitions given in section 2 and instanciated in section 3.
The ZooM system is based on the shift of α in the *alpha extension* and on the use of the two previous languages of description of classes ($\mathcal{L}_2$ and $\mathcal{L}_3$). The first task of ZooM consists in building a coarse lattice covering the whole set of the data by using $\mathcal{L}_2$ for the intensional part and ext^{100} for the extensional part. Section 4.1 presents this first step. The G($desc_{\mathcal{L}2}$,ext^{100}) Galois lattice is shown to a user who chooses the subpart of the lattice that he wants to be refined.
Section 4.2 presents a refinement algorithm of the lattice using the $ext100$ and the language $\mathcal{L}_3$. Section 4.3 presents a refinement process of the bounded coarse lattice using ext and the language $\mathcal{L}_3$. Then, we present the use of $ext\alpha$ with α different from 0 and 100 in section 4.4. Finally, we give some experimental results linked to the dataset extracted from the C/Net catalog in the section 4.5.

4.1 The coarse lattice

The coarse lattice is a subset of the G($desc_{\mathcal{L}2}$,ext^{100}) Galois lattice since we use a frequent item set approach. As it is shown in section 3.8, this subset is itself a Galois lattice with ext^{100} and then $ffreq$ for extensional projections. The choice of ext^{100} for the extensional part of the nodes has been done in order to remain in control the complexity of the constructing task and in order to present readable results to the user. The choice of $\mathcal{L}_2$ for the coarse lattice has been guided by the two following reasons. First of all, it seems interesting to begin to gather basic classes which share attributes before searching similarities linked to the values of the attributes (e.g. we gather basic classes that have a size of memory before gathering basic classes with low memory). Besides, the use of $\mathcal{L}_2$ where attributes are not optional, avoids us to obtain 'uninteresting' clusters that may appear if $\mathcal{L}_3$ is used directly. Indeed, the suffixes * and ? in $\mathcal{L}_3$ allows us to express optional attributes but makes the least common subsumption operation less usable as a similarity measure. For instance, let us consider two classes $C1$ and $C2$ with $desc_{\mathcal{L}3}(C1)=\{colour^* : \{blue, red\}, form^* : \{square, circle\}, size^* : \{small\}\}$
$desc_{\mathcal{L}3}(C2)=\{colour^? : \{pink\}, material^* : \{plastic, glass\}\}$.
$lcs_{\mathcal{L}_3}$(C1,C2) $= \{colour^* : \{blue, red, pink\}, form^* : \{square, circle\}, size^* : \{small\}, material^* : \{plastic, glass\}\}$. The least common subsumer of $C1$ and $C2$ is not equivalent to the top (for instance the domain of material is {plastic,

glass, wood} but it seems that $C1$ and $C2$ are not really similar).

The construction of the coarse lattice in Zoom has been fully described in [25, 24]. We present now a brief summary of this twofold process.
First, the data are partitioned according to their label: for each type t, we create a *basic class* named t. Its $\mathcal{L}_2$ description, $desc_{\mathcal{L}2}$(t), is obtained by computing the least common subsumer of the abstractions of its instances. This preliminary clustering step has a linear data complexity.
Second, the G($desc_{\mathcal{L}2}$,ext^{100}) Galois lattice is constructed. Our clustering algorithm, $\mathcal{L}_2$-*Cluster*, is described in [25, 24]. The algorithm is adapted from a frequent item set algorithm ([1]). In this step, since we use ext^{100} we may handle $\mathcal{L}_2$ descriptions of basic classes rather than descriptions of instances. So, the computational complexity of $\mathcal{L}_2$-*Cluster* does not depend on the size of the descriptions of initial data but only on the size of the $\mathcal{L}_2$ descriptions of basic classes. More precisely, the worst time complexity of $\mathcal{L}_2$-Cluster is exponential in the maximal size of the basic classes $\mathcal{L}_2$ descriptions.

Figure 4 shows the lattice returned by $\mathcal{L}_2$-*Cluster* when it is applied on $\mathcal{BC} = \{c_1, c_2, c_3, c_4, c_5\}$ and $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$ such that: $desc_{\mathcal{L}2}(c_1) = \{a_1, a_2, a_3\}$, $desc_{\mathcal{L}2}(c_2) = \{a_2\}$, $desc_{\mathcal{L}2}(c_3) = \{a_1, a_3\}$, $desc_{\mathcal{L}2}(c_4) = \{a_3, a_4\}$, and $desc_{\mathcal{L}2}(c_5) = \{a_1, a_3\}$
Insert figure 4 about here.

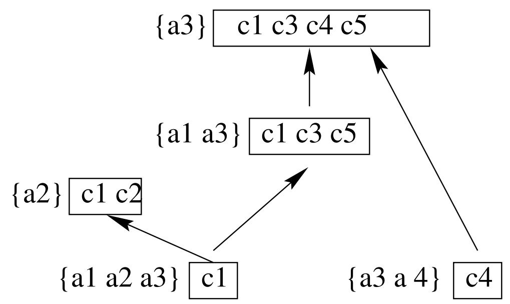

Fig. 4. Example of a lattice constructed by $\mathcal{L}_2$-Cluster

4.2 Refinement using $\mathcal{L}_3$ and ext^{100}

The goal of this step is to refine a part of the lattice G($desc_{\mathcal{L}2}$,ext^{100}) computed by $\mathcal{L}_2$-*Cluster* using the more expressive language $\mathcal{L}_3$. Since we still use ext^{100}, the Galois lattice contains only clusters of basic classes but some new clusters could be discover according to optional attributes and/or common values in the restriction of the attributes.
The refinement step is achieved after a user chose one node $Fatn$ and one of its descendents $Sonn$ in G($desc_{\mathcal{L}2}$,ext^{100}). Algorithm 1 describes how new nodes

```
Require: Two nodes Fatn and Sonn such that classes(Sonn) ⊂ classes(Fatn).
Ensure: return a lattice between Fatn and Sonn
let desc3(Fatn) = lcs_L3(classes(Fatn))
let desc3(Sonn) = lcs_L3(classes(Sonn))
L-Cl ← classes(Fatn) ∪ classes(Sonn)
LRes-Cl ← L-Cl ; Nodes ← {Sonn}
for every node n ∈ Nodes do
  for every basic class c ∈ classes(Fatn) \ classes(n) do
    Change ← false ; classes ← classes(n) ∪ {c}
    if classes ∉ L-Cl then
      L-Cl ← L-Cl ∪{classes}
      desc3 ← lcs_L3(desc3(n), desc3(c))
      (* Closure operation: *)
      for every basic class Cl ∈ classes(Fatn) \ classes do
        if desc3(Cl) ⪯_L3 desc3 then
          add Cl to classes ; Change ← true
      if classes ∉ LRes-Cl then
        add a node p to Nodes such that classes(p)=classes and desc3(p)=desc3
        LRes-Cl ← LRes-Cl ∪{classes(p)}
        if Change = true then
          L-Cl ← L-Cl ∪classes(p)
  Suppress n from Nodes
```

Algorithm 1: $\mathcal{L}_3$-Cluster

are possibly added between *Sonn* and *Fatn*. The first step of $\mathcal{L}_3$-*Cluster* consists in computing the descriptions in $\mathcal{L}_3$ for *Sonn* and *Fatn* (denoted $desc3$ in Algorithm 1). Those computations are least common subsumer calculations of class descriptions whose overall time cost is polynomial w.r.t to the size and the number of the basic classes involved in *Fatn*.
$\mathcal{L}_3$-*Cluster* computes a G($desc_{\mathcal{L}3}$,ext^{100}) Galois lattice bounded by *Fatn* and *Sonn*.

Let us illustrate the application of $\mathcal{L}_3$-Cluster on the nodes $c1\,c3\,c5$ and $c1$ of the lattice of figure 4, assuming that the $\mathcal{L}_3$ descriptions of the involved basic classes are:
$desc_{\mathcal{L}3}(c_1)=\{a_1^+ : \{v1, v3\}, a_2^+ : \{v2, v4\}, a_3^\epsilon : \{v6\}\}$, $desc_{\mathcal{L}3}(c_3)=\{a_1^\epsilon : \{v3\}, a_2^? : \{v4\}, a_3^\epsilon : \{v7\}\}$, and $desc_{\mathcal{L}3}(c_5)=\{a_1^\epsilon : \{v5\}, a_3^\epsilon : \{v7, v8\}\}$.
The new node $c1c3$ is added with the following description :
$desc_{\mathcal{L}3}(c_1, c_3) = \{a_1^+ : \{v1, v3\}, a_2^* : \{v2, v4\}, a_3^\epsilon : \{v6, v7\}\}$

Let nbc_1 be the number of basic classes of the father node. Let nbc_2 be the number of basic classes of the son node. The worst time complexity of $\mathcal{L}_3$-*Cluster* is exponential w.r.t nbc_1 - nbc_2.

4.3 Refinement using $\mathcal{L}_3$ and *ext*

This section presents the refinement of the bounded coarse lattice using $\mathcal{L}_3$ and *ext*. *ext* allows us to be completely free w.r.t. the first labelling provided by the basic classes.

Computation of the father and the son nodes The goal of this step is to obtain a node in G($desc_{\mathcal{L}3}$,ext) from a node belonging to G($desc_{\mathcal{L}2}$,ext^{100}). As both the language and the extension change, we must proceed in two steps: a step related to the intensional part and a step related to the extensional part. If the node is firstly refined according to the language and then according to the extensional function, we may obtain a different node than the node obtained by first changing the extension.
For instance, let {colour, form, size} be the set of attributes and
ext(BC1)={i1}, ext(BC2)={i2}, ext(BC3)={i3,i4,i5} with
i1 = {colour : {blue}, form : {square}},
i2 = {colour:{white}, size:{small}}, i3 = {colour:{blue}, size:{small}},
i4 = {colour:{pink}, form:{round}}, i5 = {form:{triangle}, size:{small}}.
Let n1({colour},{BC1,BC2}) the initial node in G($desc_{\mathcal{L}2}$,ext^{100}). If we begin by the change of extension, we obtain the node :
n2({colour}, {i1,i2,i3,i4}) in G($desc_{\mathcal{L}2}$,ext) and then
n3({colour$^{\epsilon}$:{blue, white, pink},form$^{?}$:{square, round}, size$^{?}$:{small}} ,{i1,i2,i3,i4})
in G($desc_{\mathcal{L}3}$,ext).
If we begin with the change of language, we obtain the node
n4({colour$^{\epsilon}$:{blue, white}, form$^{?}$:{square}, size$^{?}$:{small},{BC1,BC2}) in G($desc_{\mathcal{L}3}$,ext^{100})
and then
n5({colour$^{\epsilon}$:{blue, white}, form$^{?}$:{square}, size$^{?}$:{small}, {i1,i2,i3}) in G($desc_{\mathcal{L}3}$,ext).
Note that the nodes $n3$ and $n5$ both belong to G($desc_{\mathcal{L}3}$,ext), that their projection in G($desc_{\mathcal{L}2}$,ext^{100}) is $n1$, and that $n3$ is more general than $n5$.
In our approach, we chose to obtain the most general node, i.e. we first apply the refinement linked to the extensional function *ext* and then the refinement linked to the language $\mathcal{L}_3$.

Construction of the lattice The construction follows a top-down approach: we compute specializations of the father node which still subsume the son node. We choose to specialize nodes according to their intensional part. The specialization operations in $\mathcal{L}_3$ are the following:

Specialization operations:
Note that because of the presence of suffixes expressing optionality, the absence of an attribute in $\mathcal{L}_3$ implies that none instance of the class has the attribute in its abstraction. Unlike propositional languages, adding a missing attribute is not a specialization.
• suffix specializations: replace a suffix by a more specific suffix (see the generality order on suffixes in section 3.3).

• values specializations: remove one or several values from the current set of values.
However, the application of one or several specialization operations does not always lead to a node satisfying the Galois criteria. The general process of the construction consists therefore in alternating a specialization step and a closure step applied on the specialized description. Even if the closure step provides a gain in the complexity of the construction task, this construction remains very costly. The main reason is linked to the expressivity of $\mathcal{L}_3$ and more precisely to the presence of values that introduce a kind of disjunction in the language. The idea is then to generalize $\mathcal{L}_3$ by either keeping only few representative values for each attribute or by replacing values by general types that could be organized in a hierarchy (e.g. a *colour* will be *light* or *dark*). But, since the instances are described according to a language containing only values, we should define a pre-processing (the construction of the hierarchy of types and the link between the values and the types), which is out of the scope of the paper. Note that the field of description logics ([11, 4, 27] ...) can provide a good framework to obtain a more appropriate language. Indeed, $\mathcal{L}_3$ is a subset of the C-CLASSIC description logics [4] and the restriction of values that we use is equivalent to the value restriction in a role (what we call attributes in our approach) combining with the connective ONE-OF. We could use the classical value restriction where the fillers of roles are restricted to concepts (i.e. types) and not to individual values.
Previous work in machine learning defined and used reduced spaces [5, 13]. In [5], the authors present a specialization operator which works in a closed space. This operator has been used in a concept learning algorithm in a propositional language. In [13], the reduced space is used to construct a set of logical implications satisfied by the instances.

4.4 ZooM and extα

$\mathcal{L}_3$ and ext^{α}

In our approach, since $\mathcal{L}_3$ is quite expressive, the G($desc_{\mathcal{L}3}$,ext) Galois lattice contains too many nodes even if this lattice is bounded by a father node and a son node. On the other hand, the G($desc_{\mathcal{L}3}$,ext^{100}) Galois lattice is not enough flexible since we only gather whole basic classes. So, it seems interesting to use a *alpha extension* with α different from 0 and 100 in order to both remain in control the complexity of the construction task and to be less constrained by the basic type of the instances. We will then obtain G($desc_{\mathcal{L}3}$,ext^{α}) Galois lattices. The first idea is to construct a Galois lattice with α closed to 100 in order to handle some noise that can appear in the data. For instance, if a basic class contains one or several instances that are not 'representative' of the basic class (e.g. an exceptional instance like an ostrich in the basic class Bird), none instance of the basic class will be in the extension of a node n of G($desc_{\mathcal{L}3}$,ext^{100}) if the intensional part of n does not subsume the exceptional instances even if the other instances satisfy the description. By using ext^{98} for instance, we will

gather whole basic classes and instances satisfying the description and such that at least 98 % of the instances of their basic class satisfy the description (i.e. the basic class sat_{98} the description).
The second idea is to construct a Galois lattice with α equals to 50 or 25 for instance in order to gather at least half (or quarter) basic classes. The Galois lattice could be interesting in the case where basic classes are too general and should be divided (e.g. the basic class *computer* both contains *mac* and *PC* which makes its description too general). The use of ext^{50} or ext^{25} could be a good way to automatically obtain this division without having a pre-processing step where the basic classes are manually divided. Furthermore, if the basic classes are divided before the construction of the lattices, the basic classes obtained represent only a way to divide them (it is a static process). The use of ext^{α} allows us to divide dynamically basic classes according to different criteria. For example, we can obtain in a node of the lattice, the subset of instances of the basic class *computer* which have high size of memory and in another node the subset of instances of *computer* representing the *mac computers*.
We did not define construction algorithms of G($desc_{\mathcal{L}3}$,ext^{α}) Galois lattices with α different from 0 and 100 but we present now some drawbacks linked to ext^{α}. The first one is the fact that basic classes could have different degrees of homogeneity (some of them should be divided in three or four and others should be handled as a whole). It seems then inappropriate to fix the value of α. The second problem is that there are numerous different ways to obtain α % of a set of instances belonging to a basic class. We should therefore use some heuristics in order to construct only a part of the Galois lattice G($desc_{\mathcal{L}3}$,ext^{α}).

$\mathcal{L}_2$ and ext^{α}

ext^{α} can be used more easily using a less expressive language like $\mathcal{L}_2$. In this case, the refinement of a part of G($desc_{\mathcal{L}2}$,ext^{100}) can be done using a top-down approach. More precisely, we iterate three operations: an intensional specialization step where an attribute of the son node is added, the computation of the alpha extension of the description obtained and an intensional closure operation. This algorithm called $\mathcal{L}_2$alpha-Cluster has been implemented and some results are presented in the next section.

4.5 Experiments

We have evaluated our approach using a real dataset composed of 2274 computer products extracted from the C/Net catalog. Each product is described using a subset of 234 attributes, possibly multi-valued. There are 59 types of products and each product is labelled by one and only one type. The goal of the experiment was twofold : to assess the efficiency and the simplicity of the lattice for the first clustering step ($\mathcal{L}_2$-Cluster) and to show the accuracy of the refinement of a part of the lattice using the second clustering step ($\mathcal{L}_3$-Cluster).

In order to make the G($desc_{\mathcal{L}2}$,ext^{100}) lattice even simpler, the number of nodes obtained with $\mathcal{L}_2$-Cluster may be parametrized by a threshold n used to restrict the nodes that appear in the lattice to gather at least n basic classes. Figure ?? shows that, as it is mentioned in [33], applying this threshold results in much smaller lattices. Insert figure 5 about here. Figure 6 illustrates the simplicity of the $\mathcal{L}_2$ descriptions and the significance of the nodes. Insert figure 6 about here.

Threshold	1	2	3	4	5
Number of nodes	119	60	24	13	9
Maximal depth	4	3	2	1	1
Running Times (s)	12.3	10.4	2.1	1.1	1

Fig. 5. Quantitative results of $\mathcal{L}_2$-Cluster

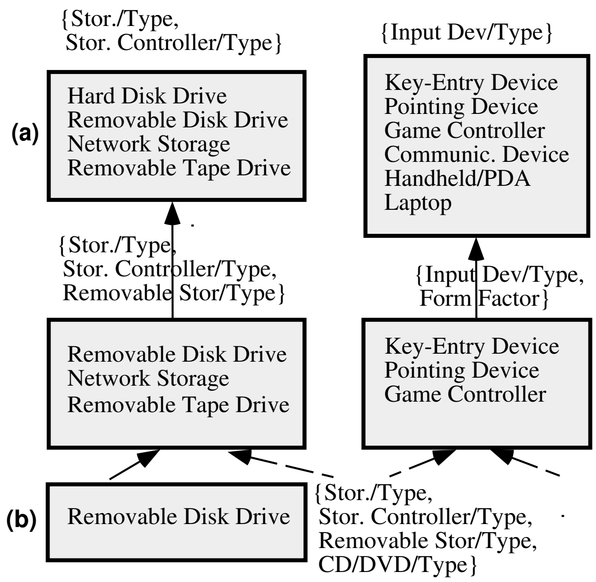

Fig. 6. A part of the G($desc_{\mathcal{L}2}$,ext^{100}) lattice for C/net (n=3)
The grey rectangles contains the names of the basic classes aggregated. The intents are presented near the nodes.

$\mathcal{L}_3$-Cluster allows to distinguish nodes that cannot be distinguished by $\mathcal{L}_2$-Cluster. For instance, if $\mathcal{L}_3$-Cluster is applied to refine the G($desc_{\mathcal{L}2}$,ext^{100}) lattice between the node (a) and the node (b) in figure 6, $\mathcal{L}_3$ is expressive enough to distinguish all possible subset of basic classes. Thus, six new nodes appear

between (a) and (b). For instance, the aggregation of all types of drivers (i.e. *RemovableTapeDrive*, *RemovableDiskDrive* and *HardDiskDrive*) is part of G($desc_{\mathcal{L}3}$,ext^{100}). This new cluster appears for the following reasons. First, no driver is described using the attributes *Storage Controller/RAID Level*, *Networking/Data Link Protocol* or *Networking/Type* (those attributes were optional in $\mathcal{L}_3$ description of (a)). Second, the value *SCSI* for the attribute *Storage Controller/Type* is not a possible value for a driver.
We also have experimented $\mathcal{L}_2$alpha-Cluster in order to refine the G($desc_{\mathcal{L}2}$,ext^{100}) lattice. Figure 7 shows the evolution of the number of new nodes when the user wants to know more about the basic class *DesktopComputer*. In that case, the refinement is done between this node and the bottom. This basic class aggregates 463 products and its description in $\mathcal{L}_2$ uses 8 attributes that are valued in all the products of *DesktopComputer* but 41 attributes are optional. The other 41 optional attributes allow to distinguish parts of this basic class when ext^{α} is used. In this basic class, a lot of attributes are valued for most of products. Thus, when α is near to 100, the refinement leads to new nodes which description includes these attributes that didn't appear in G($desc_{\mathcal{L}2}$,ext^{100}). The others products, that can be viewed as exceptional products will lead to new nodes when α is near to 0. Insert figure 7 about here.

α	100	99	98	95	90	80	50	40	30	10	0
number of nodes	0	14	18	23	26	27	30	30	31	33	36

Fig. 7. Evolution of the number of new nodes when a part of G($desc_{\mathcal{L}2}$,ext^{100}) is refined using α

This type of refinement can lead to interesting new nodes. For instance, if the refinement is made between the node (c) which aggregates *DeskTopComputer*, and *Laptop* and the node (d) *DeskTopComputer* with α equal to 90 percent, a new node appears (see figure 8). This new node aggregates all the instances of *DesktopComputers* and the instances of *Laptop* that have the attribute *modem/communication/type* valued. Indeed, most of the laptops have a modem in this catalog. Insert figure 8 about here.

5 Related work

Galois connections and Galois lattices have been widely used in various fields concerning data and knowledge representation and organisation [14, 19, 17]. In most contributions E is the set of parts of an instance set I and $\mathcal{L}$ is the set of parts of a property set A (see Example 1). However recent work in Knowledge Representation and Machine Learning investigates more complex languages. For instance in [22], Galois lattices as representations of the concept space for instance-based

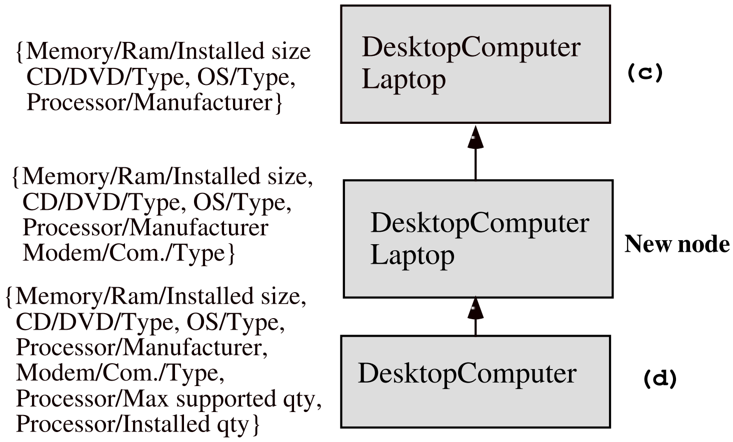

Fig. 8. Refinement of a part of G($desc_{\mathcal{L}2}$,ext^{100}) using $\alpha = 90$

Numeric-Symbolic learning use attributes that are not necessary binary. In [14] foundations of Formal Concept Analysis are presented where Concept lattices (with their extensional and intensional parts) has also been extended to non binary, numeric and multi-valued attributes through a process called *scaling*.[15] exploits scaling to deal with multiple values (as in $\mathcal{L}_1$ an instance can then be both red and blue). The incremental addition of non binary attributes [28](or *scales*) corresponds to an intensional projection for which the authors propose a graphical representation named *nested line diagrams*. The same graphical representation is used in [16] to organize the various frequent Galois lattices corresponding to an increasing constraint on frequency of nodes. In our terminology this corresponds to extensional projections. It should be emphasized that nested line diagrams are graphical representations and should not be confused with nested Galois lattices. However in the two previous work these representations do correspond, in our view, to an ordered set of nested Galois lattices. In [13] a Galois connection-based formal model of top down symbolic learning is proposed, where the value domain of each attribute is structured as a product of hierarchical posets. [9] proposes a Galois lattice for conceptual clustering where data and classes are represented as labelled recursive trees. [18] recently investigates Galois lattices in which the concept language is any structural language provided that the least common subsumer is unique and investigates complexity issues related to the cost of elementary operations. Other extensions come from symbolic Data Analysis [3] where various kind of symbolic objects, as for instance numeric intervals [26], and their related Galois lattices are investigated. In [3] functions similar to our projections are used to build Galois lattices coping with coherency constraints on the data organization. This corresponds to our construct of nested Galois lattices. However inverse projection, and composed projections were not investigated in this case as the purpose of the authors seems related to insure data coherency rather than to order and refine various represen-

tations. Regarding the reduction of the size of the Galois lattice, various criteria are used in supervised learning where descriptions are associated to positive and negative instances. For instance [21] uses entropy, [5] defines different research spaces according to the type of instances that are taken into account (i.e positive or/and negative). Other criteria cannot be defined in terms of intensional or extensional projections. For instance, [21] only develops the most general nodes. As shown in section 2, the selection of a subset of attributes does correspond to an intensional projection. This selection can be based on a saillance measure [21] or be dynamic and conducted by the end user [28]. Our aim here is not only to reduce the size of the lattice : we want to propose a change in the abstraction level of the lattice and this is realized by a shift from $\mathcal{L}_2$ to $\mathcal{L}_3$.
Other approaches perform reductions related to the extensional part of the lattice. First, section 2 shows that the selection of a subset E' of E can be viewed as an extensional projection. Thus, all the algorithms which construct a Galois lattice incrementally [6] actually refine a lattice through such a projection. Second, frequent concept lattices [32, 29, 20] are now often used in KDD and are empirically shown to significantly reduce the size of the lattice. In ZooM, such an extensional projection is also used, however our main extensional projection exploits the types associated to each data to reduce the lattice by lowering its abstraction level. In [7], the authors uses conceptual clustering to infer implication rules and functional dependencies. This approach is close to ours as it explicitly takes into account both the presence of attributes and their values. However, the language used is less expressive than $\mathcal{L}_3$ and the granularity change is not performed during the construction of the Galois lattice. The authors emphasize the second step where the Galois lattice is used to support implication rules. We plan to further extend our work by investigating the nature of implication rules that can be inferred from our Galois lattices when using α-extension.

6 Conclusion and future work

This paper has proposed an approach to organize into clusters large sets of multivalued data using Galois lattices. The complexity of the construction of such lattices is very high when handling real-world data. The main problem is then to find a good trade-off between the number of nodes of the Galois lattice and the expressivity of the language of classes used to describe the clusters.
The system ZooM constructs Galois lattices at different levels of abstraction and more precisely constructs Galois lattices that represent refinements of part of a coarse lattice. This bounded coarse lattice is nested in the refined lattice. The system ZooM uses different languages of classes and different extension functions. The scaling up used in ZooM is made possible because its complexity is controlled in different ways: (1) the data is aggregated into basic classes and the clustering is first performed on the set of basic classes instead of the data set (2) the two-step clustering method first builds a coarse hierarchy, based on a simple language for describing the clusters and a coarse notion of extension, and second refines a small part of the hierarchy delimited by two nodes, using a

more elaborate language and a more fine-grained notion of extension.
We gave some basis to this coarse-to-fine-grained order by defining the general notion of *nested Galois lattices* and we have shown that nesting is based on extensional and/or an intensional projections. The intensional projections related to the languages of class used in ZooM has been defined together with the extensional projection related to the different α-extensions.
A lot of work remains to be performed in order to improve our system. We should investigate how to adapt efficient algorithms aimed at the construction of Galois lattices (e.g. [29, 32]) in order to optimize the algorithms presented in the paper. This is not such a trivial task as efficient algorithms often relies on computation of the intent of singular instance when α-extension prevents such an independent use of instance intents. Another direction for future research is the choice of the languages of classes according to the data. The first idea is to perform a preliminary investigation of the data (resulting for instance in having type restriction rather than value restriction). The second idea is to relate the expressivity of the language to the α-extension used. So, we could automatically change the expressivity of the language when α changes (when α is closed to 100 an expressive language can be used, if α decreases a less expressive language should be used). Finally, we plan to extend our current work to take nested attributes and textual values into account in $\mathcal{L}_3$ in order to fully deal with XML data.

Appendix

Proof of theorem 1 (towards ... by extensional projection)

1) Following the definition of a Galois connection we have to prove three properties. Hereafter C1, C2 and C3 refeers to the properties written for the original Galois connection, and C'1, C'2 and C'3 the corresponding properties when considering the projected Galois connection (E',$\subseteq$, $\mathcal{L}$,$\preceq$, int,ext').

- C'1:

The function *int* is restricted to $proj_E(E)$ but satisfies C1 as (E, $\subseteq$,$\mathcal{L}$,$\preceq$, int, ext) define a Galois connection

- C'2

If $c \preceq c1$ then $ext(c) \supseteq ext(c1)$ (from property C2) and $proj_E\ (ext(c)) \supseteq proj_E(ext(c1))$ (from property P2) i.e. $ext(c) \supseteq$ ext($c1$).

- C'3

a) We have $proj_E(ext(c)) \subseteq ext(c)$ (from P3), and as a consequence $int(proj_E(ext(c))$ $\succeq$ $int(ext(c))$ (from C1). Furthermore $int(ext(c)) \succeq$ c (from C3) and as a consequence $int(proj_E(ext(c))) \succeq$ c, i.e. $int(ext'(c)) \succeq$ c.

b) Let e be an element of $proj_E(E)$, then $e = proj_E(ext(c))$ (from P1) and e $\subseteq ext(int(e))$ (from C3) so $e \subseteq proj_E(ext(int(e)))$ (from P2) i.e. $e \subseteq ext'(int(e))$.

2)

a) As an element of G, E is closed w.r.t $ext \circ int$:

$e = (ext(int(e))$, and so $proj_E(e) = proj_E(ext(int(e))$

Furthermore $proj_E(e) \subseteq e$ (from P3) and so $int(proj_E(e)) \succeq int(e)$ and

$ext(int(proj_E(e))) \subseteq ext(int(e)) = proj_E(ext(int(e))) = proj_E(e)$.

b) As$proj_E(e)$ is an element of $proj_E(E)$, we have $proj_E(e) \subseteq ext(int(proj_E(e)))$.

As a result of a) and b) we have $proj_E(e) \subseteq ext(int(proj_E(e)) \subseteq proj_E(e)$ and as a consequence $proj_E(e) = ext(int(proj_E(e)))$ and so $proj_E(e)$ is closed w.r.t $ext \circ int$. So $(proj_E(e), int(proj_E(e)))$ belongs to the Galois poset G'.

Proof of proposition 4 Nested lattice

1) p1 $\wedge$ p2 = $proj_P$(p1 $\wedge_P$ p2) ?

p1 $\wedge_P$ p2 $\leq$ p1 and p1 $\wedge_P$ p2 $\leq$ p2 (by definition of $\wedge_P$), $proj_P$ (p1 $\wedge_P$ p2) $\leq$ $proj_P$(p1) =p1 and $proj$ P (p1 $\wedge_P$ p2) $\leq$ $proj_P$ (p2) =p2 (fromP2) So $proj_P$(p1 $\wedge_P$ p2) is a lower bound of p1 and p2. Let us suppose that there exists an element z of $proj_P$(P) that is greater than $proj_P$(p1 $\wedge_P$ p2) and is a lower bound of p1 and p2:

$proj_P$(p1 $\wedge_P$ p2) $\leq$ z $\leq$ e1 and $proj_P$ (p1 $\wedge_P$ p2) $\leq$ z $\leq$ e2. As z belongs to P, we then have z $\leq$ p1 $\wedge_P$ p2. As z belongs to $proj_P$(P), we have z= $proj_P$ (z). But as $proj_P$ is monotonic we also have $proj_P$ (p1 $\wedge_P$ p2) $\leq$ $proj_P$ (z) = z. As a consequence z = $proj_P$(p1 $\wedge_P$ p2) is the unique greatest lower bound of p1 and p2 in $proj_P$(P).

2) p1 $\vee$ p2 = p1 $\vee_P$ p2 ?

p1 $\leq$ p1 $\vee_P$ p2 p1 and p2 $\leq$ p1 $\vee_P$ p2 p1 (by definition of $\vee_P$), $proj_P$(p1 $\vee_P$ p2) $\geq$ $proj_P$ (p1) =p1 and $proj_P$ (p1 $\vee_P$ p2) $\geq$ $proj_P$ (p2) =p2 (fromP2). So $proj_P$(p1 $\vee_P$ p2) is an upper bound of p1 and p2. Let us suppose that there exists an element z of $proj_P$(P) that is smaller than $proj_P$(p1 $\vee_P$ p2) and is an upper bound of p1 and p2:

$proj_P$(p1 $\vee_P$ p2) $\geq$ z $\geq$ e1 and $proj_P$ (p1 $\vee_P$ p2) $\geq$ z $\geq$ e2. As z belongs to P, we then have z $\geq$ p1 $\vee_P$ p2. As z belongs to $proj_P$(P), we have z= $proj_P$ (z). But as $proj_P$ is monotonic we also have $proj_P$ (p1 $\vee_P$ p2) $\geq$ $proj_P$ (z) = z. As a consequence z = $proj_P$(p1 $\vee_P$ p2) is the unique least upper bound of p1 and p2 in $proj_P$(P).

Furthermore from P3 (minimality of $proj_P$) we have $proj_P$ (p1 $\vee_P$ p2) $\leq$ p1 $\vee_P$ p2. And as p1 $\vee_P$ p2 is th least upper bound of p1 and p2 in P, we have p1 $\vee_P$ p2 $\leq$ $proj_P$ (p1 $\vee_P$ p2) $\leq$ p1 $\vee_P$ p2, and so $proj_P$ (p1 $\vee_P$ p2) = p1 $\vee_P$ p2.

Proof of proposition 5:

1. $c \preceq_{\mathcal{L}_3} c_i$ for all $1 \leq i \leq n$?
   let us suppose that there is a class c_i that is not subsumed by c. Then, one of these property is true (cf. definition of the subsumption): $\exists$ att $\in c_i$ such that att $\notin$ c.
   This property cannot be true since the set of attributes A of c is the set of attributes belonging to at least one description c_i.
   $\exists$ $att^{suff} : Vc \in$ c such that $att^{suff} : V \in c_i$ and V $\not\subseteq$Vc.
   This property cannot be true since Vc is the union of the sets of values associated with att in the class descriptions c_i's.
   $\exists$ $att^{+} : V_c \in$ c such that att^{+} or $att^{\epsilon} \notin c_i$
   This property cannot be true since $att^{+} \in$ c implies that att^{+} or att^{ϵ} appear

in all the c_i's.
$\exists$ $att^?: V_c \in$ c such that $att^+ : V \in c_i$ or such that $att^* : V \in c_i$
This property cannot be true since $att^? : V_c \in$ c implies that $att^+ : V \notin c_i$ or such that $att^* : V \notin c_i$.
$\exists$ $att^\epsilon : V_c \in$ c such that $att^\epsilon : V \notin c_i$
This property cannot be true since $att^\epsilon : V_c \in$ c implies that $att^\epsilon : V_c \in c_i$.
So, c $\preceq_{\mathcal{L}_3} c_i$ for all $1 \le i \le n$.

2. if d is a class description satisfying $d \preceq_{\mathcal{L}_3} c_i$ for all $1 \le i \le n$, then $d \preceq_{\mathcal{L}_3} c$?
Let us suppose that there exists d such that $d \preceq_{\mathcal{L}_3} c_i$ and such that $d \npreceq_{\mathcal{L}_3} c$.
Then, if c is not subsumed by d, one of the following property is true.
$\exists$ att $\in$ c such that att $\notin$ d.
This property cannot be true since att $\in$ c implies that $\exists$ c_i such that att $\in$ c_i. So, if att $\notin$ d then $d \npreceq_{\mathcal{L}_3} c_i$
$\exists att^{suff} : V_d \in$ d such that $att^{suff} : V_c \in c$ and $V_c \not\subseteq V_d$.
This property cannot be true since V_c is the union of the set of values associated to att in all the c_i's. So, if one of the value of V_c does not appear in V_d, $\exists$ c_i such that $d \npreceq_{\mathcal{L}_3} c_i$
$\exists$ $att^+ : V_d \in$ d such that att^+ or $att^\epsilon \notin c$
This property cannot be true since att^+ or $att^\epsilon \notin c$ implies that there exists at least one c_i such that att $\notin c_i$ or such that $att^?$ or $att^* \in c_i$. In that case, $d \npreceq_{\mathcal{L}_3} c_i$
$\exists att^? : V_d \in$ d such that $att^+ : V_c \in c$ or such that $att^* : Vc \in c$
This property cannot be true since $att^+ : V_c \in c$ implies that $\exists$ c_i such that $att^+ \in c_i$, and $att^* \in$ c implies that $\exists$ c_i such that $att^* \in c_i$ or such that $att^+ \in c_i$. So, $\exists$ c_i such that $d \npreceq_{\mathcal{L}_3} c_i$
$\exists$ $att^\epsilon : V_d \in$ d such that $att^\epsilon : V \notin$ c
This property cannot be true since $att^\epsilon : V \notin$ c implies that $\exists$ c_i such that $att^\epsilon \notin c_i$. So, $\exists$ c_i such that $d \npreceq_{\mathcal{L}_3} c_i$

Proof of proposition 10 :

- **Proof of C1**:
$\forall$ (e,e1) $\in$ E, e$\subseteq$e1 implies $desc_{\mathcal{L}_3}(e1) \preceq_{\mathcal{L}_3} desc_{\mathcal{L}_3}(e)$ The proof is straighforward with regard to the definition of the lcs.
- **Sketch of proof of C2**:
$\forall$ (c,c1) $\in \mathcal{L}_3$, c $\preceq_{\mathcal{L}_3}$ c1 implies ext(c1)$\subseteq$ext(c)
Let c and c1 be two terms of $\mathcal{L}_3$ such that c $\preceq_{\mathcal{L}_3}$ c1.
Let us postulate that there exists an instance i $\in$ E such that i is a member of c1 but i is not a member of c (i.e. c1 $\preceq_{\mathcal{L}_3}$ abs(i) and c $\npreceq_{\mathcal{L}_3}$ abs(i)). We know that abs(i) is not subsumed by c iff one the following property is true (cf definition):
$\exists$ att $\in$ abs(i) such that att $\notin$ c.
$\exists$ $att^* : V_c \in$ c such that $att^{suff} : V \in$ abs(i) and V $\not\subseteq V_c$.
$\exists$ $att^+ : V_c \in$ c such that att $\notin$ abs(i) or such that V $\not\subseteq$ V_c.
$\exists$ $att^? : V_c \in$ c such that $att^+ : V \in$ abs(i) or such that V $\not\subseteq V_c$.

$\exists$ $att^{\epsilon} : V_c \in$ c such that $att^{+} : V \in$ abs(i) or such that V $\not\subseteq V_c$ or such that att $\notin$ abs(i).
Let's consider the first property. We know that c1$\preceq_{\mathcal{L}_3}$abs(i) implies that $\forall$ att $\in$ abs(i), att $\in$ c1. Besides, c $\preceq_{\mathcal{L}_3}$ c1 implies that $\forall$ att $\in$ c1, att $\in$ c. Thus, $\forall$ att $\in$ abs(i), att $\in$ c. So, this property cannot be true.
Let's consider the second property, c $\preceq_{\mathcal{L}_3}$ c1 implies $\forall$ $att^{*} : V_c \in$ c, if there exists $att^{suff} : V_{c1} \in$ c1, $V_{c1} \subseteq V_c$. Besides, c1$\mathcal{L}_s$ abs(i) implies that if there exists $att^{suff} : V \in$ abs(i) then V$\subseteq V_{c1}$. So, if there exists $att^{suff} : V \in$ abst(i) then V$\subseteq V_c$.
We have proved this way that the last mentioned properties cannot be true.

– **proof of C3**:
$\forall$ c $\in \mathcal{L}_3$, c $\preceq_{\mathcal{L}_3}$ $desc_{\mathcal{L}_3}(ext(c))$ and $\forall$ e $\in$ E, e $\subseteq$ ext($desc_{\mathcal{L}_3}(e)$).
$\forall$ c $\in \mathcal{L}_3$, c $\preceq_{\mathcal{L}_3}$ $desc_{\mathcal{L}_3}(ext(c))$
We know that $\forall$ i $\in$ ext(c), c $\preceq_{\mathcal{L}_3}$ abs(i) by definition of the extension. So, we know that c $\preceq_{\mathcal{L}_3}$ $desc_{\mathcal{L}_3}(ext(c))$ by definition of the lcs.
Besides, $\forall$ i $\in$ e, $desc_{\mathcal{L}_3}(i1,..,in)$ $\preceq_{\mathcal{L}_3}$ abs(i) (by definition of the lcs)
So,$\forall$ e $\in$ E, e $\subseteq$ ext($desc_{\mathcal{L}_3}(e)$) (by definition of the extension).

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *20th VLDB Conference*, Santiago, Chile, 1994.
2. Gareth Birkhoff. *Lattice Theory.* American Mathematical Society Colloquium Publications, Rhode Island, 1973.
3. H.H. Bock and E. Diday. *Analysis of Symbolic Data.* H.H. Bock and E. Diday, Springer Verlag, 2000.
4. A. Borgida and P. F. Patel-Schneider. Complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial intelligence Research*, 1:278–308, 1994.
5. P. Brézellec and H. Soldano. Tabata: a learning algorithm performing a bidirectional search in a reduced search space using a tabu stratégie. In H. Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence*, pages 420–424, Brighton, Angleterre, 1998. J. Wiley.
6. C. Carpineto and G. Romano. A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, 1(24):95–122, 1996.
7. Claudio Carpineto, Giovanni Romano, and Paolo dAdamo. Inferring dependencies from relations: a conceptual clustering approach. *Computational Intelligence*, 15(4):415–441, 1999.
8. William W. Cohen. An analysis of representation shift in concept learning. In *Machine Learning*, pages 104–112, 1990.
9. M.C. Daniel-vatonne and C. Hemce. On a tree-like representation for symbolic-numeric data and its use in galois lattice method. In *Theoretical Informatics and Applications, Proceedings of the XVIII International Conference of the Chilean Computer Science Society, November 12-14, , Antofagasta, Chile*, 1998.
10. B. Davey and H. Priestley. *Introduction to Lattices and Order.* Cambridge University Press, 1990.

11. F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In J.A. Allen R. Fikes and E Sandewall, editors, *Principles of Knowledge Representation and Reasoning: 2nd International Conference*, pages 151–162, Cambridge, Mass., 1991.
12. G. Drastal and G. Czako. Induction in an abstraction space: A form of constructive induction. In *Proceedings of the 12th International Jointed Conference in Artificial Intelligence*, pages 708–712, 1989.
13. J. Ganascia. Tdis: an algebraic formalization. In *International Joint Conference on Artificial Intelligence*, pages volume 2, 1008–1013, Vancouver, BC, Canada, 1993.
14. B. Ganter and R. Wille. *Formal Concept Analysis: Logical Foundations*. Springer Verlag., 1999.
15. R. Godin, G.W. Mineau, R. Missaoui, and H. Mili. Méthodes de classification conceptuelle basées sur les treillis de galois et applicationz. *Revue d'Intelligence Artificielle*, 9(2):105–137, 1995.
16. Joachim Hereth, Gerd Stumme, Rudolf Wille, and Uta Wille. Conceptual knowledge discovery and data analysis. In *International Conference on Conceptual Structures*, pages 421–437, 2000.
17. M. Huchard, H. Dicky, and H. Leblanc. Galois lattice as a framework to specify building class hierarchies algorithms. In *Theoretical Informatics and Applications, EDP Sciences, n. 34*, pages 521–548, 2000.
18. M. Liquiere and J. Sallantin. Structural machine learning with galois lattice and graphs. In *Machine Learning: Proceedings of the International Conference ICML98, Morgan Kaufmann*, 1998.
19. Guy W. Mineau, Gerd Stumme, and Rudolf Wille. Conceptual structures represented by conceptual graphs and formal concept analysis. In *International Conference on Conceptual Structures*, pages 423–441, 1999.
20. R. Taouil N. Pasquier, Y. Bastide and L. Lakhal. Discovering frequent closet itemsets for association rules. In *ICDT*, pages 398–416, 1999.
21. G.P. Njiwoua Njamen. *Contribution à l'apprentissage symbolique automatique par l'usage du treillis de Galois*. Thèse, Université d'Artois, France, 2000.
22. P. Njiwoua and E. Nguifo. Iglue: An instance-based learning system over lattice theory. In *Proceedings of ICTAI'97*, pages 255–274, California, USA, 1997. IEEE Press.
23. Giulia Pagallo and David Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5:71–99, 1990.
24. N. Pernelle, M-C. Rousset, and V. Ventos. Automatic construction and refinement of a class hierarchy over multi-valued data. In *5th Conference on Principles and practice of Knowlewdge Discovery in Databases, PKDD'01, Lecture Notes in Artificial Intelligence*, pages 386–398. Springer-Verlag, 2001.
25. N. Pernelle, M-C. Rousset, and V. Ventos. Construction et affinement de hiérarchies partir de données semi-structurées. *Extraction des connaissances et apprentissage*, 1(1-2):81–92, 2001.
26. G. Polaillon. *Organisation et interprétation par les treillis de Galois de données de type multivalué, intervalle ou histogramme*. Thèse, Université Paris 9, France, 1998.
27. J. Quantz, G. Dunker, F. Bergmann, and I. Kellner. The flex system. Technical report, KIT-Report, Technische Universität, Berlin, Germany, 1996.
28. G. Stumme. *Local scaling in conceptual data systems*. Number 1115 in Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin, Germany, 1996.

29. G. Stumme, R. Taouil, Y. Bastide, and L. Lakhal. Fast computation of concept lattices using data mining techniques. In *Proceedings of the 7th International Workshop on Knowledge Representation Meets Databases KRDB'2000*, pages 129–139, 2000.
30. M. Tambe, A. Newell, and P. Rosenbloom. The problem of expensive chunks and its solution by restricting expressiveness. *Machine Learning*, 5:299–348, 1990.
31. P. E. Utgoff. Shift of bias for inductive concept learning. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, volume II, pages 107–148, Los Altos, CA, 1986. Morgan Kaufmann.
32. K. Waiyamai and L. Lakhal. Knowledge discovery from very large databases using frequent concept lattices. In *11th European Conference on Machine Learning, ECML'2000*, pages 437–445, 2000.
33. R. Wille. Restructuring lattice theory. In *Symposium on Ordered Sets*, pages 445–470, University of Calgary, Boston, 1982.