When concepts point at other concepts: *the case of UML diagram reconstruction*

Marianne Huchard [1] and **Cyril Roume**[2] and **Petko Valtchev** [3]

Abstract. *Relational datasets, i.e., datasets in which individuals are described both by their own features and by their relations to other individuals, arise from various sources such as databases, both relational and object-oriented, or software models, e.g., UML class diagrams. When processing such complex datasets, it is of prime importance for an analysis tool to hold as much as possible to the initial format so that the semantics is preserved and the interpretation of the final results eased. There have been several attempts to introduce relations into the Galois lattice and formal concept analysis fields. We propose a novel approach to this problem which relies on an extension of the classical binary data descriptions based on the distinction of several mutually related formal contexts. As we impose no restrictions on the relations in the dataset, a major challenge is the processing of relational loops among data items. We present an approach for constructing lattices on top of circular descriptions which is based on an iterative approximation of the final solution. The underlying construction methods are illustrated through their application to the restructuring of class hierarchies in object-oriented software engineering, which are described in UML.*

1 Introduction

Formal Concept Analysis (FCA) [12] focuses on the lattice structure induced by a binary relation between a pair of sets (called *objects* and *attributes*, respectively), known as the *Galois* lattice [1] or the *concept* lattice [34] of the relation.

Recently, FCA, Galois lattices and derived structures and techniques have been successfully applied to the resolution of practical problems from a wide range of scientific disciplines including data mining [27], knowledge acquisition [24], and software engineering [18, 30].

While the classical FCA problem statement only considers binary relations, i.e., objects being described by Boolean attributes, the many practical datasets include individuals of richer object descriptions. Thus, a main axis of research on FCA has aimed at integrating further attribute types, e.g., numerical, categorical, taxonomic, etc., into the initial framework, either by scaling back to binary attributes (via *conceptual scaling* as in [11]) or by extending the definition of the *Galois connection* [1] that underlies the lattice structure. Within this axis, a particular trend has investigated the processing of objects whose descriptions go beyond the limits of propositional logics, i.e., include some relations information [23, 22, 9]. Following a similar track, we address the specific problems of processing individuals that are characterized by their relations to other individuals.

In this paper, we put the problem in a specific application context which is the restructuring of class hierarchies in object-oriented software engineering (OOSE), an area where FCA and Galois lattices have already proven their utility as analysis tools [15, 7, 31]. In this particular framework, the input data are described as UML class diagrams which include classes and inter-class associations, while the aim is to discover potentially useful abstractions of both classes and associations which are further to be used by a human re-engineer in order to improve the current class hierarchy.

As both classes and associations are to be processed and since the incidence between an association and a set (usually a pair) of classes is a key information, the task amounts to constructing two lattices with their elements, i.e., the formal concepts, being characterized by local properties and relations to other formal concepts (here to concepts from the opposite lattice). The key difficulty with such a generic statement resides in the two-way dependency that relations induce on the two lattices.

As a contribution to the problem of relational FCA and Galois lattice construction, we present a method for constructing related concepts in the general case, i.e., even with cyclic dependencies between objects. Its key idea is to compute the final lattices as the least fixed point of a function that essentially maps a collection of lattices (one per object sort) into another collection of lattices. At each step, the function, which is defined only in an operational manner, uses the concepts discovered by the previous steps to refine the object descriptions and then reconstructs the lattices on top of the updated datasets.

The paper first figures out the difficulties met by FCA techniques when processing relations in the data and draws a clear distinction between previous work and our own approach which is then described in details (Section 2). Issues pertaining to our application context, the reorganization of class diagrams in UML, are then discussed with a particular stress on the way problems from the domain are translated into the language of FCA and Galois lattices (Section 3). Next, details of the encoding and the processing of a concrete example of a UML class diagram are discussed (Section 4). Finally, the paper list the open problems and other questions that our study has helped figure out (Section 5).

2 Construction Lattices from Complex Objects

2.1 Motivation

Since their introduction as data analysis tools, the Galois (concept) lattice-related have been repeatedly investigated for possible extensions toward more expressive object descriptions than pure binary at-

[1] LIRMM, 161 rue Ada, 34392 Montpellier Cedex 5, France
[2] LIRMM, CNRS et Université Montpellier 2, 161 rue Ada, 34392 Montpellier Cedex 5, France
[3] DIRO, Université de Montréal, CP 6128, Succ. Centre-Ville, Montréal Québec H3C 3J7

tributes. Some researchers explored the possibility of translating the various kinds of data back to binary variables, e.g., through *conceptual scaling* [11]. Others have considered the definition of the fundamental construct in this domain, i.e., the *Galois connection*, over descriptions whose elements have more complex inner structure, e.g., fuzzy [13], probabilistic [8] or rough sets [21].

Yet different research stream aimed at overcoming the inherent limits of pure attribute-value description formalisms that compare to zero-order or propositional logic languages. For instance, specific lattice construction methods have been defined for structured terms [5], on function-free predicate-logic languages (Datalog) [2] and conceptual graphs [25]. In a very rough manner, the descriptions of formal objects considered within this trend compare to a set of predicates linking object parts among them and therefore may be represented as graphs where such parts are vertices and the (binary) predicates are edges (see the left-hand side of Figure 1). Thus, the extraction of formal concept descriptions involves particular forms of graph isomorphism computation which obviously suffers on well-known limitations in the general case. Nevertheless, recent work on the subject [23, 22] has pushed further the frontier of tractable classes of graph-based descriptions.

The existing graph-based and other first-order approaches focus exclusively on relations that lay strictly within the boundaries of the considered formal objects. However, in many situations, in particular with datasets from object-oriented or object-relational databases, the individuals to be analyzed are related to other individuals (e.g., the relation `married-to` for a set of human beings) and inter-individual relations encode important information that may help the formation of meaningful abstractions [32]. In particular, some formal concepts of high interest for data mining tasks may be defined with respect to other formal concepts (e.g., the spouses of *Master Gold* credit card beholders). As the aforementioned relational methods fail to discover relations that cross the concept boundaries, new methods have to be devised to deal with such data.

To our best knowledge, only few studies have investigated the formal analysis of datasets where objects are related among them. For example, in the work of Faïd *et al.* [9] (which generalizes earlier work of Wille [35]), several types or *sorts* of formal objects are considered together with a set of relations among objects from different *sorts*. The paper presents an elegant way of extracting implication rules from objects with complex (relational) structure. However, the presented results are based on strong hypotheses about the data (only relations are used to describe objects, relations are oriented but cycles are prohibited, etc.) and therefore their generalization to more realistic datasets is a challenge of its own. In particular, situations like the one drawn on the right of Figure 1 will be impossible to tackle with the presented techniques. In the next paragraph we present a formal way of stating the problem of relational FCA.

2.2 Theoretical framework

The following problem statement generalizes the classical way of introducing the data in Galois lattice construction and FCA. The novel element is the presence of relational attributes that link objects from a family of *multi-valued* contexts (see [12]).

FCA basics First, we recall that a formal context $\mathcal{K}$ is a three-tuple $\mathcal{K} = (O, A, I)$ where O is a set of (formal) objects, A a set of (formal) attributes, and $I \subseteq O \times A$ a binary incidence relation between objects and attributes. The relation I induces two mappings, f and g, that link 2^O and 2^A both ways:

$$for\ X \in 2^O,\ f(X) = \{y \in A \mid \forall x \in X, (x, y) \in I\}$$
$$for\ Y \in 2^A,\ g(Y) = \{x \in O \mid \forall y \in Y, (x, y) \in I\}$$

The joint applications of both mappings, i.e., $f \circ g$ and $g \circ f$, define two closure operators on 2^O and 2^A, respectively. In the sequel, both f and g will be denoted by $'$ and the closure operators by $''$. The families of closed subsets of O and A, denoted $\mathcal{C}^o_{\mathcal{K}}$ and $\mathcal{C}^a_{\mathcal{K}}$ (or simply $\mathcal{C}^o$ and $\mathcal{C}^a$), present two remarkable properties: (i) the operators $'$ represent bijective mappings between these families, and (ii) when provided with set-theoretical inclusion, both families constitute complete sub-lattices of the respective powerset lattices, which, in addition are isomorphic. When considered as a separate entity, a pair of mutually corresponding closed sets $c = (X, Y)$ from $\mathcal{C}^o$ and $\mathcal{C}^a$, respectively, is called a *formal concept* in [12] where the set of objects, X, is called *extent* and the set of attributes, Y, *intent*. The set of all such pairs, $\mathcal{C}_{\mathcal{K}}$ (or simply $\mathcal{C}$), with an order that follows inclusion on extents constitutes a third lattice, $\mathcal{L}_{\mathcal{K}}$ or simply $\mathcal{L}$, known as the *Galois lattice* [1] or the *concept lattice* [34] of the context $\mathcal{K}$.

Since Galois/concept lattices have been initially intended as data analysis tools, there has been a significant effort aimed at the extension of the model toward more complex attribute structures. In particular, in formal concept analysis (FCA) [12], non binary (e.g., numerical, ordinal, categorical, etc.) attributes are introduced via a multi-valued context, i.e., a four-tuple $\mathcal{K} = (O, A, V, I)$ in which A is a set of (non necessarily binary) attributes, V is a set of (attribute) values and $I \subseteq O \times A \times V$ is a ternary relation with member tuples $(o\ a\ v)$ being interpreted as “the object o has a value v for the attribute a”. In order to extract the lattice $\mathcal{L}$, the context $\mathcal{K}$ is first transformed into an equivalent one-valued, or binary, context $\mathcal{K}^d$ called the *derived* context. For this purpose, standard scaling techniques may be used such as discretization of continuous variables, category binary encoding, etc. *Conceptual scaling* [11] provides a complete framework for transforming any multi-valued context into a binary one.

The following table presents a sample multi-valued context, called *Human*, that will be used as a running example throughout this section. It is made up of six objects (1, 2, ..., 6) representing human beings and two attributes, *age* and *work*, modeling the age and the years of work experience for a person, respectively.

	1	2	3	4	5	6
age	25	28	22	26	33	35
work	4	2	1	8	4	10

The context *Human*^{d}, obtained by scaling *Human* with (arbitrary) thresholds of 27 for *age* and 5 for *work*, is as follows:

	short	1	2	3	4	5	6
age ≤ 27	a	X		X	X		
age > 27	b		X			X	X
work ≤ 5	c	X	X	X		X	
work > 5	d				X		X

The lattice $\mathcal{L}_H$ corresponding to *Human*^{d} is given in Figure 2, on the left.

In the sequel, we present an extension of the multi-valued context framework to include relational information that is frequently met in datasets extracted from relational or object-oriented databases.

Relational contexts

Definition 1 (Relational context family)
A relational context family (RCF) $\mathcal{R}^s$ *is a pair* $(\mathbf{K}_{\mathcal{R}}, \mathcal{A}_{\mathcal{R}})$ *where* $\mathbf{K}_{\mathcal{R}}$

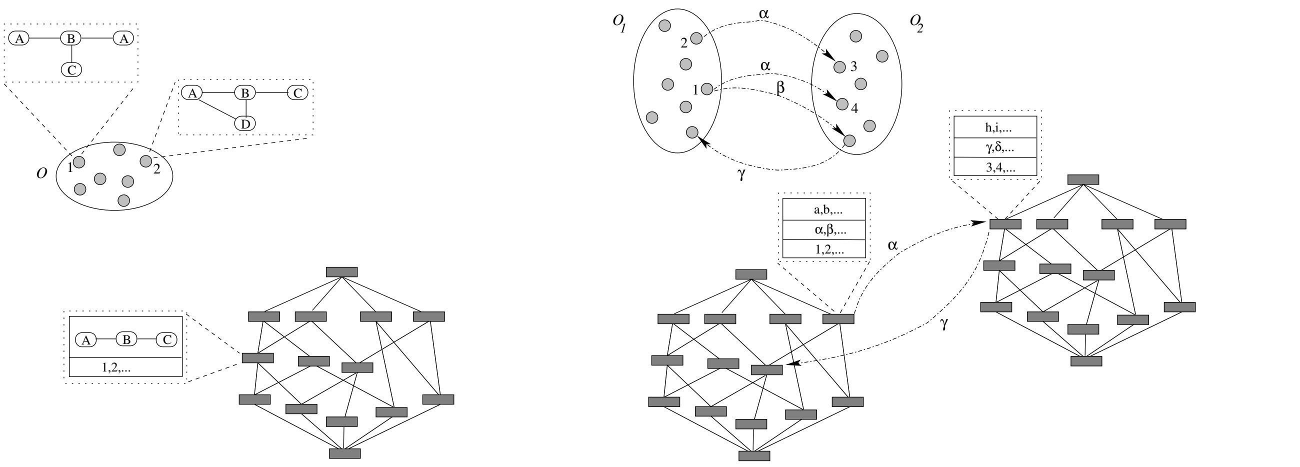

Figure 1. **Left**: Classical approach for processing relational data in FCA: relations remain within concept descriptions. **Right**: Our approach: relations between individuals are reflected by the intentional descriptions of the discovered concepts.

is a set of s multi-valued contexts $\mathcal{K}_i = (O_i, A_i, V_i, J_i)$ and $\mathcal{A}_\mathcal{R}$ is a set of p relational attributes (set-valued functions) α_j such that for each j, $1 \leq j \leq p$ there exist r and q in $[1, s]$ with $\alpha_j : O_r \to 2^{O_q}$.

The mappings $dom : \mathcal{A}_\mathcal{R} \to \{O_i\}$ (domain) and $cod : \mathcal{A}_\mathcal{R} \to \{O_i\}$ (co-domain) are defined for a RCF $\mathcal{R}^s$ in the following way: for any $\alpha : O_r \to 2^{O_q}$, $dom(\alpha_j) = O_r$ and $cod(\alpha_j) = O_q$. Moreover, the set of all relations of a given context within a RCF, is computed by the function $rel : \mathcal{K}_\mathcal{R} \to 2^{\mathcal{A}_\mathcal{R}}$, with $rel(\mathcal{K}_i) = \{\alpha | dom(\alpha) = O_i\}$.

For instance, consider a RCF $\mathcal{R}^1_H = (\{Human\}, \{\mu\})$, where the function $\mu : O_{Human} \to 2^{Human}$ models the spouse links between persons (obviously, $\mu : O_{Human} \to O_{Human}$ holds). In the above example, we set μ to the following set of object-value pairs: $\{(1, \{2\}), (2, \{1\}), (3, \{4\}), (4, \{3\}), (5, \{6\}), (6, \{5\})\}$.

Given a relational context family $\mathcal{R}^s$, our aim will be to construct s lattices of formal concepts $\mathcal{L}_i$, $(0 \leq i \leq s)$, one per multi-valued context $\mathcal{K}_i$, such that the concepts of a particular context not only reflect shared attributes, but also similarities in object relations. To clarify this goal, let us consider a particular function α_j that maps O_r into 2^{O_q} for some r and q. Let us now assume that the lattice $\mathcal{L}_q = \langle \mathcal{C}_q, \leq_q \rangle$ corresponding to a binary context $\mathcal{K}^d_q$ derived (in an unspecified way) from $\mathcal{K}_q$, is already available. Consider now the context $\mathcal{K}^+_r = (O^+_r, A^+_r, V^+_r, J^+_r)$ that is obtained from $\mathcal{K}_r$ in the following way. First, the set of objects is kept unchanged while a new attribute $\alpha_\uparrow$ is added in A^+_r. The values of the new attribute are sets of concepts from $\mathcal{C}_q$, i.e., $\alpha_\uparrow : O^+_r \to 2^{\mathcal{C}_q}$. The value $\alpha_\uparrow(o)$ is computed from the values $\alpha(o)$ by simply filtering all concepts in $\mathcal{C}_q$ whose extent contains at least one object from $\alpha(o)$. We shall call the above context $\mathcal{K}^+_r$ the *relational extension* of $\mathcal{K}_r$ upon the pair $(\alpha, \mathcal{L}_q)$. This is essentially captured in the next formal definition.

Definition 2 (Relational extension of a context within a family) *Given a context $\mathcal{K}_r$ from a RCF $\mathcal{R}^s = (\mathbf{K}_\mathcal{R}, \mathcal{A}_\mathcal{R})$, an attribute α in $\mathcal{A}_\mathcal{R}$ with $cod(\alpha) = O_q$, and a lattice $\mathcal{L}_q$ corresponding to a binary context derived from $\mathcal{K}_q$, the context $\mathcal{K}^+_r = (O^+_r, A^+_r, V^+_r, J^+_r)$ where:*

- $O^+_r = O_r$,
- $A^+_r = A_r \cup \{\alpha_\uparrow\}$,
- $V^+_r = V_r \cup 2^{\mathcal{C}_q}$,
- $J^+_r = J_r \cup \{(o\ \alpha_\uparrow\ v^*) |\ o \in O_r\}$ *where the set v^* may be defined also as $v^* = \{(X, Y) \in \mathcal{C}_q |\ \alpha(o) \cap X \neq \emptyset\}$.*

is called the relational extension of $\mathcal{K}_r$ upon $(\alpha, \mathcal{L}_q)$.

The concept of *relational extension* easily generalizes to a set of pairs $\mathcal{E} = (\alpha_j, \mathcal{L}_j)$. Finally, a binary context $\mathcal{K}^+_r$ is a *complete relational extension* of a multi-valued context $\mathcal{K}$ within a RCF $\mathcal{R}^s$ if it is a *relational extension* upon a set $\mathcal{E}$ in which for any α in $rel(\mathcal{K})$, there exists a pair $(\alpha, \mathcal{L}_t)$.

Let now $\mathcal{K}^{+^d}_r$ is a binary context derived from $\mathcal{K}^+_r$. Standard scaling procedures for taxonomic, or partially ordered, attributes like $\alpha_\uparrow$ would assign a binary attribute a for each v^* such that $(o\ \alpha_\uparrow\ v^*) \in V^+_r$. However, here we consider an equivalent, but more compact binary encoding that uses one attribute for each formal concept in $\bigcup_{o \in O_r} \alpha_\uparrow(o)$. With our running example, the derived context of the relational extension of *Human* with respect to $(\mu, \mathcal{L}_H)$ (see Figure 2, left), further called *Human*$^{+^d}$, is as follows:

	short	1	2	3	4	5	6
$age \leq 27$	a	X		X	X		
$age > 27$	b		X			X	X
$work \leq 5$	c	X	X	X		X	
$work > 5$	d				X		X
#2	e		X	X	X		
#3	f	X	X		X		X
#4	g			X		X	
#5	h	X				X	X
#6	i			X			
#7	j		X		X		
#8	k					X	
#9	l	X					X
#1	m	X	X	X	X	X	X
#10	n						

The second part of the above context includes the binary attributes corresponding to each concept in $\mathcal{L}_H$ (identified by a unique number in Figure 2). The attributes modeling the top and bottom concepts of $\mathcal{L}_H$ are singled out since neither of them contributes to the shape of the lattice corresponding to *Human*$^{+^d}$.

The lattice $\mathcal{L}^+_r$ of a context $\mathcal{K}^{+^d}_r$ which is obtained by any scaling that satisfies the above encoding schema is said to be *relationally compliant* to the lattice $\mathcal{L}_q$ via the relational attribute α. To generalize, we shall say that any lattice $\mathcal{L}_r$ corresponding to a relational extension of an initial multi-valued context $\mathcal{K}_r$ is *relationally compliant* to another lattice $\mathcal{L}_q$ if it is relationally compliant via all relational attributes α, such that $\alpha : O_r \rightarrow 2^{O_q}$. In particular, the relationally compliant property holds for lattices for which no such α exists. For example, within $\mathcal{R}^1_H$, the lattice $\mathcal{L}_H$ (see Figure 2, left) is not relationally compliant to itself via μ (since not isomorphic to the lattice of *Human*$^{+^d}$). In contrast, the lattice $\mathcal{L}^+_H$ (see Figure 2, right), which is drawn from *Human*$^{+^d}$, is not only relationally compliant to $\mathcal{L}_H$, but also to $\mathcal{L}^+_H$, i.e., to itself.

Problem statement We can now state our lattice-construction problem in the following way:

Given : A relational context family $\mathcal{R}^s = (\mathbf{K}_\mathcal{R}, \mathcal{A}_\mathcal{R})$ with s contexts $\mathcal{K}_i = (O_i, A_i, V_i, J_i)$.
Find : A set of s lattices $\mathbf{L} = \{\mathcal{L}_i\}_{0 \leq i \leq s}$ such that:

i. each $\mathcal{L}_i$ corresponds to a binary context derived from a complete relational extension of $\mathcal{K}_i$ whereby the elements of set $\mathcal{E}$ include only lattices from $\mathbf{L}$, and

ii. for each $0 \leq i, j \leq s$, $\mathcal{L}_i$ is relationally compliant to $\mathcal{L}_j$.

The above definitions represent a strict generalization the framework for complex object processing based on context concatenation as considered by Wille and Faïd *et al*. Indeed, when at most one relation is provided for each initial context, there is an intuitive lattice construction procedure that consists in regarding object sorts, i.e., contexts, as the nodes of a (set of) chain(s). The nodes of the chain are traversed in reverse order and at each step the lattice of the current context is constructed based on structural information about the lattice in the unique (if any) referenced node. At the end, at the highest level of the chain, all the lattices are already constructed. If several chains are present, they are processed separately.

This intuitive procedure generalizes easily to more complex structures in the relational context family like trees and even DAGs. However, DAGs are the least constraint structure of a family that admits this kind of treatment. Whenever loops appear in the structure, another technique should be applied. The reason for this limitation may be summarized as follows: in the simplest case, two contexts $\mathcal{K}_r$ and $\mathcal{K}_q$ are related both ways by two (or more) attributes in $\mathcal{A}_\mathcal{R}$, say α_j and α_l with $dom(\alpha_j) = \mathcal{K}_r$ and $dom(\alpha_l) = \mathcal{K}_q$. In order to construct the corresponding lattices $\mathcal{L}_r$ and $\mathcal{L}_q$ and to insure they are mutually compliant via α_j and α_l, one should chose which lattice is to be constructed first. However, whatever the choice, the other lattice should be available to insure compliance. This obvious contradiction motivates a different strategy for dealing with cycles in the relations.

2.3 Iterative multi-lattice construction

We propose a method (see Algorithm 1) for constructing concepts on top of each dataset, despite the obvious presence of loops in the inter-individual links.

To deal with loops, our method applies an iterative computation strategy, that approaches the final solution starting from a very rough approximation and refining it at each step. Thus the final solution is computed as the fixed point of a complex function that maps the set of lattices at a particular iteration to the set of lattices at the next one. More precisely, an iteration of the algorithm consists in subsequent visits of all contexts. The function uses the lattice structures discovered at the previous step to refine the information available for the current step. In other words, while the initial contexts remain the same, their complete relational extension evolve along the iterations

proc MULTI-FCA(**In:** $\mathcal{R}^s = (\mathbf{K}_\mathcal{R}, \mathcal{A}_\mathcal{R})$ a RCF,
  Out: $\mathbf{L}$ array of lattices)
$i \leftarrow 0$; halt $\leftarrow$ **false**
for j **from** 1 **to** s **do**
  $\mathbf{L}^i[j] \leftarrow$ FCA($\mathcal{K}_j$)
while not halt **do**
  $i++$
  for j **from** 1 **to** s **do**
    $\mathcal{K}^+_j \leftarrow \mathcal{K}_j$
    for all α **in** $rel(\mathcal{K}_j)$ **do**
      Let $cod(\alpha) = O_q$ **in**
        $\mathcal{K}^+_j \leftarrow$ EXTEND-REL($\mathcal{K}^+_j$, α, $\mathbf{L}^{i-1}[q]$)
    $\mathcal{K}^d_j \leftarrow$ SCALE-BIN($\mathcal{K}^+_j$, $\mathbf{L}^{i-1}$)
    $\mathbf{L}^i[j] \leftarrow$ FCA($\mathcal{K}^d_j$)
  halt $\leftarrow \bigwedge_{j=1,s}(\mathcal{L}^i_j = \mathcal{L}^{i-1}_j)$

Algorithm 1: Construction of the set of Galois (concept) lattices corresponding to a RCF.

since the precision of the available descriptive information increases. The entire iterative process halts when a grouping step does not lead to the discovery of previously unseen formal concepts. In an equivalent way, this also means that all lattices are mutually compliant.

When applied to the example RCF, $\mathcal{R}^1_H$, the above method halts in three steps. The initial step yields the lattice $\mathcal{L}^0_H$ which is visualized on the left of Figure 2 (initially called $\mathcal{L}_H$). From the new context *Human*^{1} (former *Human*$^{+^d}$) that relationally extends *Human* with respect to $(\mu, \mathcal{L}^0_H)$, the lattice $\mathcal{L}^1_H$ is obtained (see Figure 2, right). The new lattice leads to a further relational extension *Human*^{2} of the already extended *Human* context. However, all the new attributes inferred from $\mathcal{L}^1_H$ happen to be combinations of attributes in *Human*^{1}, which means, in particular that $\mathcal{L}^1_H$ and $\mathcal{L}^2_H$ are isomorphic.

The above approach is still to be studied for verification and validation purposes and the underlying properties are far from being well understood. However, first steps have already been made in this direction. Thus, the convergence of the algorithmic method, although not yet formally established, intuitively holds. Indeed, if we see our iterative process as a series of approximations of the final solution in terms of, say, formal contexts, each member of the series, except for the very first one, will be an extension of the previous member. As the set of objects O in each context remains steady (we never add new objects), only attribute A_i sets may vary between steps. However, the latter sets can only grow or remain the same with respect to the previous step ($A_i \subseteq A_{i+1}$, this can be proven inductively). Equivalently, the set of the formal concepts of such an augmented context, $\mathcal{C}_i$, is a super-set of the set same set extracted from the previous version of he context. Thus, between two steps the size of the lattice of a context could not decrease. Finally, let us observe that the number of the concepts in such a lattice at any moment could not exceed the size of the powerset of the object set, $|\mathcal{C}_{i+1}| \leq |2^O|$ which is constant.

Notwithstanding the general lack of formal results about the behavior of the MULTI-FCA method, the related construction techniques have already found interesting applications in software engineering as it is explained by the next section.

3 Application context

We illustrate the possible benefits of the algorithmic techniques described in Section 2 through an application to class hierarchy reorganization from UML models. In the following, we describe briefly the

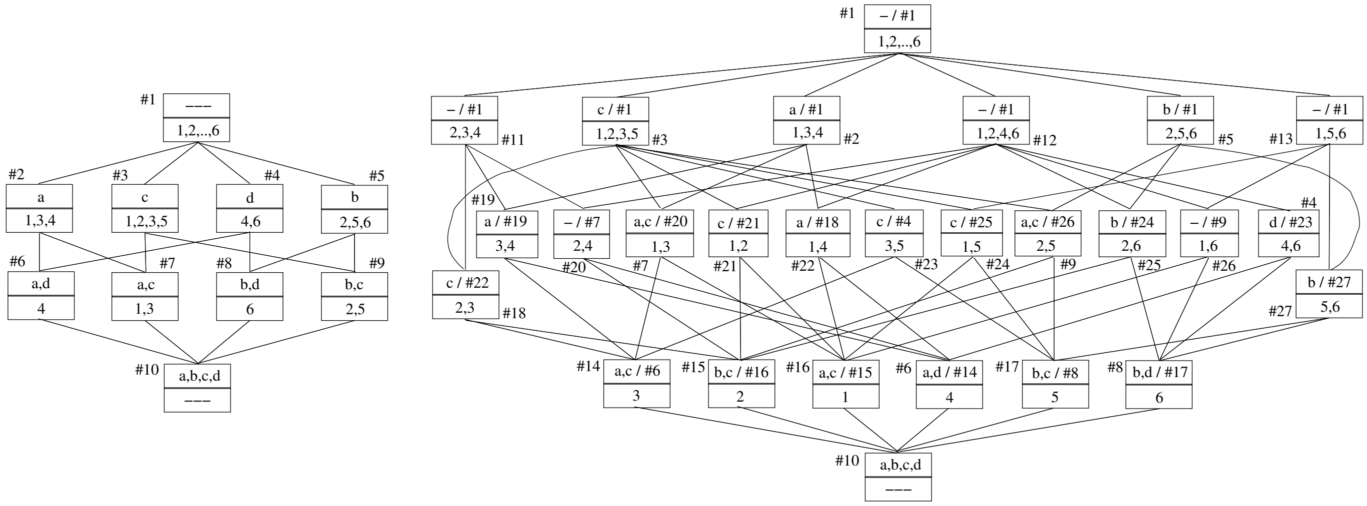

Figure 2. Two lattice corresponding to steps of MULTI-FCA() on the RCF $\mathcal{R}_H^1 = (\{Human\}, \{\mu\})$: $\mathcal{L}_H^0$ (left) and $\mathcal{L}_H^1$ (right). Concept intents in $\mathcal{L}_H^1$ include both conventional attributes and the inter-concept relations induces by μ (both parts are separated by a slash). For conciseness reasons, a relational attribute is represented by the identifier of the minimal concept shared by the μ_f values for objects in the concept extent.

general context of the application, i.e., the goal of hierarchy reorganization, input data, target hierarchy structure, etc.

3.1 Hierarchy restructuring for optimization

Class hierarchies are designed once but then used in various circumstances they have not necessary been intended for in the beginning. Thus, often changes must be made to adopt them to a new context, but also to improve their quality from software engineering (SE) point of view. Therefore, some restructuring may be necessary and this is done mostly at the re-engineering step [10]. As we have pointed out in [14], software designers look for hierarchies that are maximally factored. Simply put, factorization reduces the redundancies exist in the source code produced from the hierarchy and therefore increases the maintainability and the extensibility of the hierarchy.

Galois (concept) lattices provide strong theoretical foundations for the factorization task through the closure properties of the formal concepts. However, the structural complexity of the complete lattice has motivated the study of relevant lattice sub-structures of reasonable size.

3.2 Galois sub-hierarchy of a binary relation

Concept lattices have been introduced in object-oriented approaches for class hierarchy construction by [15]. In their approach, the authors propose to use a noteworthy sub-order of the lattice, in order to reduce conceptual as well as theoretical complexity. This sub-order, called the "Galois sub-hierarchy" can be defined by a simplification of the Galois lattice as proposed in [15, 16]. Many methods for automatic class hierarchy construction, like in [15, 17, 7, 4, 3, 26, 36, 20], get or approach the Galois sub-hierarchy, as it is shown in [19] for global algorithms. This reinforces the intuition according that Galois sub-hierarchy is "the normal form" for class hierarchies.

To be concise, we introduce here a straightforward definition which is easy to prove equivalent to that of [15] (see [19] for a proof of the equivalence).

Definition 3 (Galois sub-hierarchy)
Let (E, F, R) a formal context and following sets of concepts: $\mathcal{C}^E = \{(\{e\}'', \{e\}') \mid e \in E\}$; $\mathcal{C}^F = \{(\{a\}', \{a\}'') \mid a \in F\}$.

The Galois sub-hierarchy $GSH(R)$ is the set $\mathcal{C}^E \cup \mathcal{C}^F$, provided with $<_{GSH}$ which is defined by $(X_1, Y_1) <_{GSH} (X_2, Y_2)$ iff $X_1 \subset X_2$ (which is equivalent to $Y_2 \subset Y_1$).

In other words, elements of the Galois sub-hierarchy are the "(formal) object concepts" and the "(formal) attribute concepts" in [12], but note that here the relation is not supposed to be reduced in the sense of [1] and the Galois sub-hierarchy is therefore not limited to meet-irreducible and join-irreducible elements of the lattice.

In the case of a multi-valued context $\mathcal{K} = (O, A, V, J)$ we would have $E = O$, $F \subseteq A \times V$, and $(o, a, v) \in J$ iff $(o, (a, c)) \in R$ where o is a formal object and (a, c) is a formal attribute.

A natural compaction of node labels both in the Galois lattice and in the GSH consists of representing each formal object (attribute) only once in the entire structure: in the label of its object (attribute) concept, i.e., the minimal (maximal) concept it belongs to. In the retrieval of the initial labels from their compact versions, the following property is used: a formal object which belongs to the extent of a concept c also belongs to the extents of all its super-concepts. Thus, the extent of a simplified concept contains the names of the input classes "defined" by this node (the property set of these objects is the complete intend). When this extent is empty (this concept is necessarily in $\mathcal{C}^F$), a new factorization concept has been discovered. The intent of a simplified concept contains the properties "declared" by the node. When this intent is empty (this concept is necessarily in $\mathcal{C}^E$), all the properties of the node are declared by upper nodes and the node necessarily defines at least one input object. For SE applications, the Galois sub-hierarchy preserves the most informative part of the lattice: the nodes that define at least one object or declare at least one property; all other nodes (whose extent and intent are both empty) are eliminated.

Previous work on the applications of lattices and GSH to the reconstruction of class hierarchies in object-oriented programming [15, 26, 17] have considered a fixed schema. In the input, the classes of a hierarchy become the formal objects and the variables and methods the formal attributes. The simplified Galois sub-hierarchy provides a new class hierarchy, in which formal concepts are interpreted as output classes: the compact intent is the set of declared properties, while the compact extent determines which input classes are represented by the output class. This resulting class hierarchy has three essential properties: (i) properties are maximally factored, (ii) links within the hierarchy specifically correspond to specialization/generalization relationships between concepts, and (iii) the number of concepts introduced is minimal and linear in the total number of classes and properties.

In the next section, we only focus on the construction of the GSH, since more appropriated for hierarchy restructuring than the complete Galois (concept) lattice. However, the reasoning we apply naturally extends to the entire lattice.

3.3 Processing hierarchies in UML

UML (Unified Modeling Language) [28] is a popular language in the SE community used in the initial modeling and the design steps of a software life-cycle. It offers the possibility to express very fine knowledge about concepts and individuals, in particular about inter-individual links and the inter-concept relations they stem from.

In this paper we focus on conceptual models or class diagrams which are made up of classes and associations. In this context, classes represent domain categories and are identified by unique names. They may have attributes and methods which can be in turn described (with varying precision). Classes are visualized in a diagram as rectangles with distinct compartments for names, attributes, and, whenever applicable, methods. For example, within the UML model of the real estate domain shown in Figure 3, there are nine classes, inclusive `Landlord`, `Tenant` and `House`. Moreover, `Landlord` is described by a `name` and an `address`, while a `House` has a `type`.

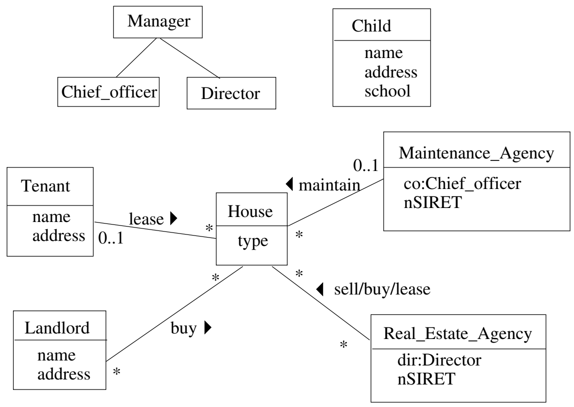

Figure 3. A UML static diagram - House Transactions

Associations in UML model the relations among members of the domain categories that are represented as classes. For example, in Figure 3, a landlord can `buy` a house, while a tenant can `lease` such a house. Like classes, associations may have attributes and methods, although usually they do not have any. Binary associations are graphically represented in UML by a line linking two classes. This line supports several annotations like the association name usually followed by a black triangle indicating the direction for reading the name: a tenant `leases` a house (not the reverse). The symbol or the number written close to a class end (for example `0..1` for the class end `Tenant`) indicates a multiplicity: here a house is supposed to be leased by at most one tenant (the multiplicity is `0..1`); a tenant can lease several houses (as indicated by the multiplicity symbol `*`). Other annotations will be mentioned in Section 4.1.

3.4 Motivating the reconstruction of the example

The example in Figure 3, which was drawn from [15], shows a set of associations which pertain to house transactions: tenants `lease` houses; landlords `buy` them; maintenance agencies are responsible for `maintaining` them (lift revision, façade and corridors painting, etc.); real estate agencies play the role of intermediates in various commercial house transactions like `selling, buying` and `leasing`. All offices are described by a registration number denoted by `nSIRET`. Classes `Manager`, `Chief-Officer` and `Director` are used as knowledge of the domain, as they may help in identifying a generalization of variables `co` and `dir`. In this aspect, the class `Child` is used as a "counter-example" for the discovery of useful abstractions in the house transaction domain, since children cannot be involved in house transactions.

Methods currently used in class hierarchy reconstruction are not devised for associations, since they were mainly dedicated for object-oriented languages that only consider variables and methods. When implemented in a programming language, binary associations are not systematically represented: they may appear through several attributes and methods in only one end class or in the two end classes, or the association is it-self implemented as a class. Retrieving and understanding associations after their implementation in the programming language model is recognized a difficult re-engineering problem as a lot of the semantics is lost. As a consequence, we consider that appropriate reconstruction of UML static diagrams needs considering associations as such, without a translation phase into the programming language model.

Our approach thus consists of considering two kinds of formal objects, classes and associations, which are separately dealt with. Their respective formal attributes should capture their mutual relationships. Separate the two processes is important for the designer to understand step-by-step the construction.

The result of the method we propose is presented in Figure 4. Classes and associations with symbolic names have been discovered. Their meaning is detailed at the bottom of the figure. Most of the new "concepts" are of great importance, as the general concept of house transaction, the persons (resp. the organisms) involved in a house transaction, etc. At a later step of design, such concepts could support new methods or variables, like an insurance policy or laws specific to some house transactions. As in all generalization methods, less relevant concepts (like `NC4` and `a2`) can appear. Final re-shaping is left to the designer who can eliminate such irrelevant concepts.

The process which realizes the GSH-based reconstruction of the above example by systematically applying the iterative method from Section is described with greater details in Section 4. In the following paragraph we present an extension of the traditional class-property model used in the hierarchy reorganization literature.

3.5 Generic properties

The knowledge of the application domain often indicates that some sets of formal attribute can be gathered into what we call a generic

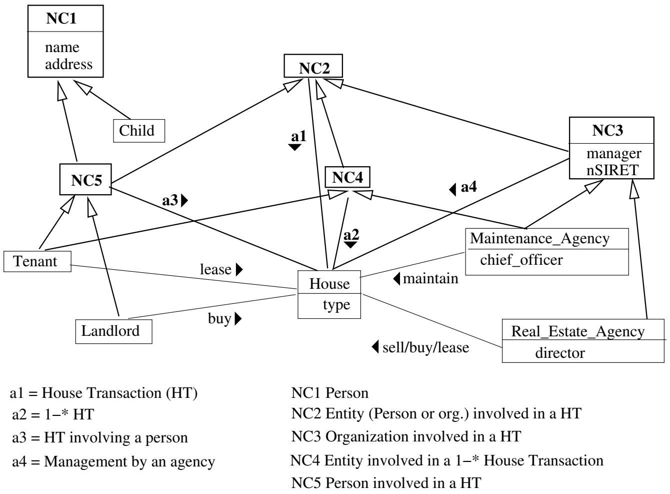

Figure 4. The UML diagram revisited - House Transactions

property, provided with a specialization order.

Definition 4 (Generic Property) *A generic property $\mathcal{F}$ is a set of properties which have close semantics and are partially ordered by a specialization order, denoted below by $\leq_{\mathcal{F}}$.*

The Galois sub-hierarchy ensures that generic feature specialization and concept specialization are in accordance provided that the binary relation encodes property specialization: for every generic feature $\mathcal{F}$, if $a_i \leq_{\mathcal{F}} a_j$, then $g(a_i) \subseteq g(a_j)$ holds. Even not explicitly mentioned, this policy is respected everywhere in our approach.

In a multi-valued context $\mathcal{K} = (O, A, V, J)$ the set A represents the set of the generic properties. Each generic property gathers a set of elements of V and a specialization order between them. Elements of V can be interpreted as values, which explains we sometimes denote a formal attribute (a, v) by $a = v$.

As a first example, consider in Figure 3 the variables `co:Chief_Officer` and `dir:Director`. The designer, guided by the knowledge of the domain expressed by specialization on class `Manager`, can consider that `co:Chief_Officer` and `dir:Director` can be generalized by `m:Manager`, and these three properties are gathered into a generic property. For example, $\mathcal{F} = \{$`co:Chief_Officer`,`dir:Director`, `m:Manager`$\}$ with `co:Chief_Officer` $\leq_{\mathcal{F}}$ `m:Manager` and `dir:Director` $\leq_{\mathcal{F}}$ `m:Manager`.

Some generalizations come from designers and some others can be automatically computed thanks to heuristics. As another common example, when a method is overridden, the call to the hidden method through the `super` keyword indicates a semantic specialization of the method. More generally, in several cases, the simple fact of overriding (even without any call to the overridden method) can be considered as a specialization for developers.

4 An Iterative Approach for Class and Association Abstraction

The application of the previous techniques requires the definition of an appropriate encoding for all the relevant information that a class diagram embodies. Thus, we first describe the translation of the UML class diagram reconstruction problem in terms of a relational context family (Section 4.1). Then the iterative process is described (Section 4.2).

4.1 Multi-valued contexts for Classes and Associations

The field of this study is limited to binary associations although this is not an actual limitation in the scope of results. From a theoretical point of view, any association with arity three or more can indeed be modeled with binary associations. Furthermore practical advice are to mainly use binary associations [29].

A key step in our proposal is determining the right formal attributes and generic properties that will describe classes and associations and lead to interesting generalizations. UML is a good guide for that as it is provided with a detailed syntax defined by its meta-model [28]. In Figure 5, main aspects of UML associations are highlighted. The UML meta-model states that an association is a generalizable element, which is composed of at least two association ends. An association end is characterized by:

- a type (the class C involved in this end). `Person` and `Order` are the two type ends of the association `place_order`;
- a visibility (or access control). When nothing is mentioned, the end is public;
- a multiplicity (number, interval, symbol '*').

It can also be specified that the association is an aggregation (a `Document` is an aggregation of `paragraphs`) and if instances of C involved in the association are ordered. An association end has also variables (qualifiers) which are often used to reduce the multiplicity: a `Bank` has several `clients`, but a bank *plus* a client number determine only one client. In the visual notation (as it appears in Figure 5), an association end is sometimes provided with a role name. When the association owns variables and methods it is considered as an association class (`Access` is an association class that supports the variable `passwd`). All these aspects cannot be detailed here for sake of simplicity, but Figure 5 gives an idea of the major possibilities.

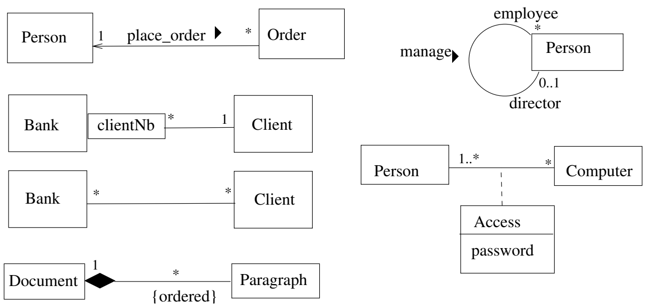

Figure 5. Examples of UML associations

Our method uses the relational context family $\mathcal{K}_{\mathcal{R}} = \{\mathcal{K}_1 = (O_1, A_1, V_1, J_1), \mathcal{K}_2 = (O_2, A_2, V_2, J_2)\}$, where $\mathcal{K}_1$ is the multi-valued context for classes (O_1 is the class set) and $\mathcal{K}_2$ is the multi-valued context for associations (O_2 is the association set).

Non Relational Class Description (A_1, V_1 and J_1) When the associations are not considered, it appears that a class C needs to be described by the following formal attributes: variables and methods as in the previous approaches. A_1 is the set of (non relational) generic variables and methods while V_1 is the set of variables and methods. J_1 represents the "owns-property" relation: $(o\ a\ v) \in J_1$ if the class o owns the element v of the generic property a. In this paper we do not deeply address this aspect, only give examples of variables in classes. The Reference [6] details this problematic in the case of UML.

Relational Class Description $(rel(\mathcal{K}_1))$ Information on the relationships between classes and UML associations is represented by three relational attributes in $\mathcal{A}_{\mathcal{R}}$:

- $origOf : O_1 \longrightarrow 2^{O_2}$, $a_i \in origOf(C)$ if the class C is origin of the association a_i,
- $destOf : O_1 \longrightarrow 2^{O_2}$, $a_i \in destOf(C)$ if the class C is destination of the association a_i,
- $clOf : O_1 \longrightarrow 2^{O_2}$, $a_i \in clOf(C)$ if the class C is an association class for a_i.

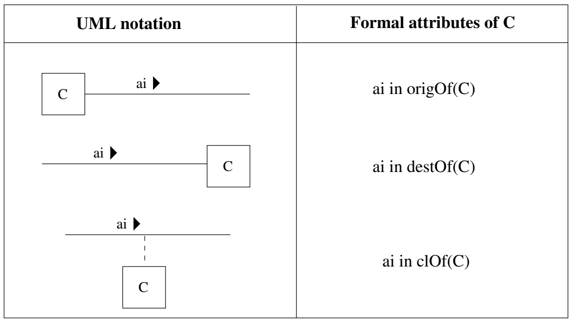

Figure 6. Overview of formal attributes for classes

Figure 6 gives a visual overview of these description rules. In Figure 7 is presented the relation describing the classes of the Figure 3 (names have been shorten).

									origOf=				destOf=			
	name	addr	sc	co	dir	man	nS	ty	l	b	m	s	l	b	m	s
Child	x	x	x													
Tenant	x	x							x							
Landlord	x	x								x						
Maint_Ag				x		x	x				x					
RE_Ag					x	x	x					x				
House								x					x	x	x	x

sc = school   ty = type   b = buy   s = sell/buy/lease
ns = nSIRET   l = lease   m = maintain

Figure 7. Class description $\mathcal{R}_{C\,0}$ - House Transactions

$origOf$, $destOf$ and $clOf$ can be considered as generic properties. Their specialization orders are directly based on association concept specialization in a Galois sub-hierarchy $\mathcal{GSH}_2 = \langle \mathcal{C}_2, \leq_2 \rangle$ corresponding to a context $\mathcal{K}_2^d$ derived from $\mathcal{K}_2$.

Rule 1 (Origin/Destination/Class) *$a_i \leq_{origOf} a_j$ (resp. $a_i \leq_{destOf} a_j$, $a_i \leq_{clOf} a_j$) if $a_i \leq_2 a_j$.*

Non Relational Association Description (A_2, V_2 **and** J_2 When the classes are not considered, it appears that an association a needs to be described by the following generic properties:

- role name of the origin nro and role name of the destination nrd,
- multiplicity of the origin mo and of the destination md,
- navigability from origin to destination $navOD$, and conversely $navDO$,
- access control (protection) on origin po and on destination pd,
- constraints on origin cto and on destination ctd (constraints such as *ordered*).

As for multiplicity, we consider that specialization is based on the restriction of possible values. Multiplicity is considered to be an interval of integers.

Rule 2 (Multiplicity) *$mo = E$, $E \subseteq \mathcal{N}$, (resp. $md = E$) specializes $mo = F$, $F \subseteq \mathcal{N}$, (resp. $md = F$) if $E \subseteq F$.*

A_2 is the set of the previous generic properties while V_2 is the set of their valuations, for example $mo = 1..*$. J_2 represents the "owns-property" relation: $(o\ a\ v) \in J_2$ if the association o owns the element v of the generic property a.

Relational Association Description $(rel(\mathcal{K}_2))$ Information on the relationships between classes and UML associations is represented by three relational attributes in $\mathcal{A}_{\mathcal{R}}$:

- $to : O_2 \longrightarrow 2^{O_1}$, $C \in to(a_i)$ if the class C is origin of the association a_i,
- $td : O_2 \longrightarrow 2^{O_1}$, $C \in td(a_i)$ if the class C is destination of the association a_i,
- $ca : O_2 \longrightarrow 2^{O_1}$, $C \in ca(a_i)$ if the class C is an association class for a_i.

An overview of formal attributes for associations appear in Figure 8 while description of associations of the house transaction example is given in Figure 9 (names have been shorten).

	mo=		md=	to=				td=
	0..1	*	*	Te	La	MAg	REAg	Ho
lease	x	x	x	x				x
buy		x	x		x			x
maintain	x	x	x			x		x
sell/buy/lease		x	x				x	x

Figure 9. Association description $\mathcal{R}_{A\,0}$ - House Transactions

to, td and ca can also be considered as generic properties. Their specialization orders are directly based on association concept specialization in a Galois sub-hierarchy $\mathcal{GSH}_1 = \langle \mathcal{C}_1, \leq_1 \rangle$ corresponding to a context $\mathcal{K}_1^d$ derived from $\mathcal{K}_1$. When instances origins of an association as are instances of a concept C_1 and C_1 is a subconcept of C_2, then instances origin of as are also instances of C_2 according to the natural semantics of class specialization in UML.

Rule 3 (End types) *$C_1 \leq_{to} C_2$ (resp. $C_1 \leq_{td} C_2$) if $C_1 \leq_1 C_2$.*

4.2 The Iterative Process

This section introduces an instanciation of the MULTI-FCA algorithm for the contexts K_1 and K_2 coming from UML class diagram. Besides Galois sub-hierarchy rather than Galois lattice are constructed.

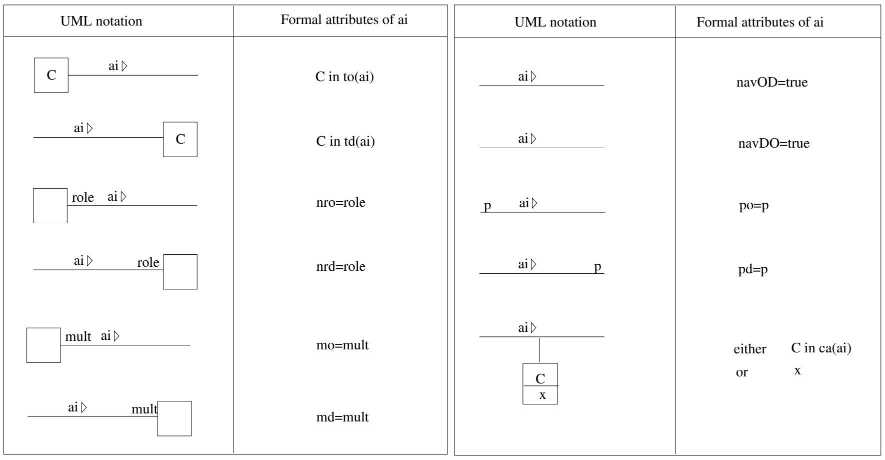

Figure 8. Overview of formal attributes for associations

This gives the generalization method $GAGCI$ (for "Generalization of Associations, Generalization of Classes, Iteration") which consists in iterating a pair of generalization methods:

- computation of the Galois sub-hierarchy associated with a multi-valued context for associations (at step i, denoted by $\mathcal{K}_2^i$)
- computation of the Galois sub-hierarchy associated with a multi-valued context for classes (at step i, denoted by $\mathcal{K}_1^i$)

The input relations are obtained by associating to every formal object (class or association) its formal attributes as well as their generalizations (using rules 1, 3 and 2). Each sub-step enriches the binary relation of the previous sub-step by new informations, until no new concept is added during a generalization step.

We denote by $\mathcal{C}_{\mathcal{K}_j^i}$ the set of concepts of the Galois sub-hierarchy associated with $\mathcal{K}_j^i$.

A step of the process is developed below.

Step i. ($GAGCI(i)$)

a Computation of $GSH()$

b Computation of $\mathcal{K}_1^{i+1}$

c Computation of $GSH(\mathcal{K}_1^{i+1})$

d Computation of $\mathcal{K}_2^{i+1}$

In particular step [b] enriches the description of classes as follows: let a be an association of $\mathcal{C}_{\mathcal{K}_2^i}$, *i.e.* an association concept created (or re-created) by generalizing associations of O_2, the class context has to evolves as to add $a \in origOf \uparrow (C)$ to every class C which is such that $b \in origOf \uparrow (C)$ with $b <_{GSH(\mathcal{K}_2^i)} a$. A similar operation is applied to $destOf$ and $clOf$.

Similarly step [d] enhances the description of associations as follows: let c be a class of $\mathcal{C}_{\mathcal{K}_1^{i+1}}$, *i.e.* a class concept created (or re-created) by generalization of classes of 0_1, the association context has to evolves as to add $C_2 \in to \uparrow (a)$ to every association a which is such that $C_1 \in to \uparrow (b)$ with $C_1 <_{GSH(\mathcal{K}_1^{i+1})} C_2$. A similar operation is applied to td and ca.

4.3 The House transaction example (continued)

The iterative process is applied to the house transaction example.

Step $GAGCI(0)$**.a** Construction of $GSH(\mathcal{K}_2^0)$ introduces two new associations:

- a_1 which generalizes all the input associations, which own or specialize $mo = *$, $md = *$ and $House$ as a destination,
- a_2, which generalizes $lease$ and $maintain$ which share the description of a_1 as well as $mo = 0..1$.

In Figure 10 are shown $SH(\mathcal{K}_2^0)$ (top) and induced generic properties (bottom).

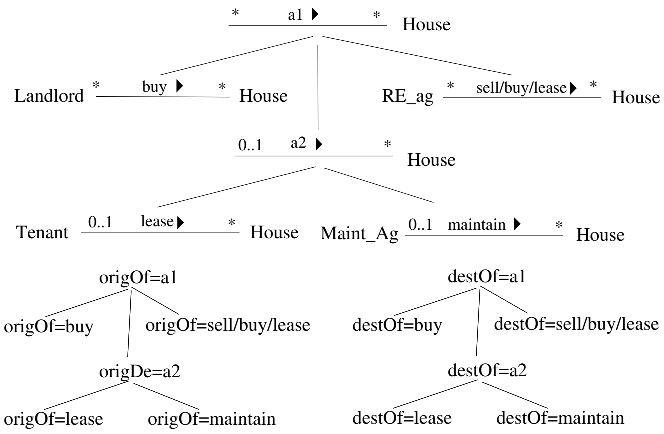

Figure 10. $GSH(\mathcal{K}_2^0)$ and generic properties - House Transactions

Step $GAGCI(0)$**.b** Generic properties are used to adjust the context $\mathcal{K}_1^0$. Figure 12 (top) presents the resulting relation $\mathcal{K}_1^1$.

Step $GAGCI(0)$**.c** $GSH(\mathcal{K}_1^1)$ is shown in Figure 12 (bottom). The class which actually is origin of an association a is the class which declares $origOf = a$.

									origOf=				destOf=				origOf=		destOf=	
	name	addr	sc	co	dir	man	nS	ty	l	b	m	s	l	b	m	s	a1	a2	a1	a2
Child	x	x	x																	
Tenant	x	x							x								x	x		
Landlord	x	x								x							x			
Maint_Ag				x		x	x				x						x	x		
RE_Ag					x	x	x					x					x			
House								x					x	x	x	x			x	x

Figure 11. $\mathcal{K}_1^1$ - House Transactions

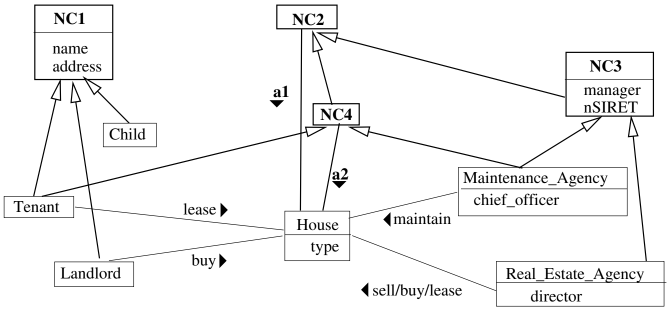

Figure 12. $GSH(\mathcal{K}_1^1)$ - House Transactions

Step $GAGCI(0)$.d $\mathcal{K}_2^0$ is now adjusted by the possible generalizations in the generic properties to and td (rule 3). The resulting relation $\mathcal{K}_2^1$ appears in Figure 14 (top).

Step $GAGCI(1)$.a $GSH(\mathcal{K}_2^1)$ is presented in Figure 14 (bottom). Classes $NC1$ and $NC3$ have induced two properties $to = NC1$ and $to = NC3$ which determine two new associations, $a3$ which generalizes $(lease, buy)$ and $a4$ which generalizes $(maintain, sell/buy/lease)$.

	mo=		md=	to=				td=	to=			
	0..1	*	*	Te	La	MAg	REAg	Ho	NC1	NC2	NC3	NC4
lease	x	x	x	x				x	x	x		x
buy		x	x		x			x	x	x		
maintain	x	x	x			x		x		x	x	x
sell/buy/lease		x	x				x	x		x	x	

Figure 13. $\mathcal{K}_2^1$ - House Transactions

Step $GAGCI(1)$.b These two new associations bring new information on classes, as shown in Figure 15. $\mathcal{K}_1^2$ is obtained from $\mathcal{K}_1^0$ (without integrating concepts of $GSH(\mathcal{K}_1^1)$) as some of them could be intermediate artifacts): $\mathcal{K}_1^0$ is filled using the generic properties $origOf$ and $destOf$ adjusted to take into account the new concepts of $GSH(\mathcal{K}_2^1)$. $GSH(\mathcal{K}_1^2)$ is shown in Figure 4.

Step $GAGCI(1)$.c When $GSH(\mathcal{K}_1^2)$ is constructed, a new concept appears ($NC5$) which factors $origOf = a3$, while $NC3$ reappears completed for factorizing not only $nSIRET$ and $resp$ as in the previous construction, but also $origOf = a4$.

Step $GAGCI(1)$.d According to the same rules as previously, the new class $NC5$ modifies the description of associations giving $\mathcal{K}_2^2$ (Figure 16). $GSH(\mathcal{K}_2^2)$ does not contain new associations (there is no new association closed set) but $a3$ has more information about its origin, which is now $<= NC5$. The process stabilizes.

5 Open questions

The function MULTI-FCA() was defined which maps a RCF to a set of relationally compliant lattices and illustrated it with a real-life example. However, the provided definition of MULTI-FCA() is purely operational, i.e., given through an algorithm. For many reasons, in particular for soundness and completeness proofs, an equivalent analytical expression of MULTI-FCA(), i.e., via a formula over the initial contexts, might prove more appropriate. Moreover, as we defined the target vector of lattices as the least fixed point of a complex function, one might want to know how many such fixed points exist.

The result about the convergence of the MULTI-FCA method (see Section 2.3) says nothing about the number of the iterations necessary for the algorithm to converge to a final solution. This is a separate topic that we are currently investigating.

The establishment of the theoretical complexity of the iterative algorithm is a challenge on its own since a well-founded reasoning about complexity would require the analytical expression of the function MULTI-FCA(). In contrast, a straightforward calculation would rely on factors like the cost of a single lattice construction, the number of the iterations (unknown *a priori*) and the size of the RCF.

Concerning practical performances, the method requires the subsequent computation of a series of lattice vectors where a single lattice computation is alone a computationally-intensive task. This fact indicates that the iterative method should be better mastered before we could apply it to large datasets. However, performance gains could be realized through various optimizations.

An important source of computational gains is the application of flexible algorithms for lattice construction. In fact, as we mentioned above, between two steps, a contexts evolves only by (possibly) extending its attribute set while the object set remains steady. Therefore, at iteration $i + 1$, instead of constructing the lattice $\mathcal{L}_{i+1}$ from scratch, one may use the available structure in $\mathcal{L}_i$ and simply "extend" it with the additional attributes. Techniques for lattice completion and lattice merge upon context joins have already been studied (see [33] for a discussion).

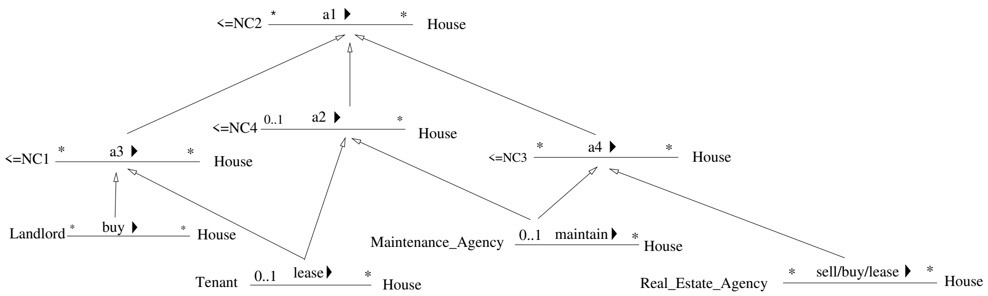

Figure 14. $GSH(\mathcal{K}_2^1)$ - House Transactions

									origOf=				destOf=				origOf=		destOf=		origOf=		destOf=	
	name	addr	sc	co	dir	man	nS	ty	l	b	m	s	l	b	m	s	a1	a2	a1	a2	a3	a4	a3	a4
Child	x	x	x																					
Tenant	x	x							x								x	x			x			
Landlord	x	x								x							x				x			
Maint_Ag				x		x	x				x						x	x				x		
RE_Ag					x	x	x					x					x					x		
House								x					x	x	x	x			x	x			x	x

Figure 15. $\mathcal{K}_1^2$ - House Transactions

6 Conclusion

We presented a framework for constructing Galois lattices and derived structures from relational datasets which deals with inter-individual links in a direct manner, i.e., without flattening the relational structure. The framework was originally designed as a method for discovering useful abstractions from UML class diagrams by extracting the Galois sub-hierarchies of the binary tables induced by UML classes and UML associations, respectively. Meanwhile, it has been completed and generalized as to fit to an arbitrary FCA problem involving relational descriptions.

Our approach relies on the definition of relational context families and relational extension between contexts and relational compliance between Galois (concept) lattices. A conceptual-level algorithmic method was described that constructs several lattices on top of a relational context family, i.e., a set of formal contexts whose objects are related by links. In case of circular dependencies between contexts of the family, the construction proceeds by subsequent lattice construction and relational extensions of contexts. Our method is therefore capable of discovering formal concepts characterized by their relations to other formal concepts, even if some loops exist in object structure.

The iterative construction strategy computes the fixed point of a lattice function that is still to discover (since so far defined only in an operational mode). Beside its novelty in the field of relational data mining, the method provides an example of how knowledge discovered at one step could be re-used in further step.

The practical impact of the new approach is clearly illustrated by its prime motivation, i.e., the need for UML class diagram analysis tools. As a matter of fact, $GAGCI$, the concrete method for constructing both GSH from a given UML class diagrams that has been presented in this paper, is a key part of the MACAO project funded by France Télécom. Separately, a variant of $GAGCI$ is currently being implemented in the CASE Tool OBJECTEERING (Softeam).

In the near future, besides all the work on the aforementioned open problems, we shall be carrying out a complete set of tests on the $GAGCI$ tool. In particular, the performances of the method on industrial-scale class diagrams will be investigated within the MACAO project. Another research track involves experiments on various scenarios of human interactions with the CASE tool where designers could guide the reconstruction algorithm.

ACKNOWLEDGEMENTS

We would like to thank Michel Dao, Robert Godin and Therese Libourel for fruitful discussions.

REFERENCES

[1] M. Barbut and B. Monjardet, *Ordre et Classification: Algèbre et combinatoire*, volume 2, Hachette, 1970.

[2] L. Chaudron and N. Maille, 'First order logic formal concept analysis: from logic programming to theory', *Computer and Information Science*, **13**(3), (1998).

[3] J.-B. Chen and S. C. Lee, 'Generation and reorganization of subtype hierarchies', *Journal of Object Oriented Programming*, **8**(8), (1996).

[4] W.R. Cook, 'Interfaces and Specifications for the Smalltalk-80 Collection Classes', in *Proceedings of OOPSLA'92, Vancouver, Canada*, special issue of ACM SIGPLAN Notices, 27(10), pp. 1–15, (1992).

[5] M.-C. Daniel-Vatonne, *Les termes : un modèle de représentation et structuration de données symboliques*, Thèse de doctorat, Université Montpellier II, 1993.

[6] M. Dao, M. Huchard, T. Libourel, and C. Roume, 'Spécification de la prise en compte plus détaillée des éléments du modèle objet UML', Technical report, Projet MACAO. Réseau RNTL, (2001).

[7] H. Dicky, C. Dony, M. Huchard, and T. Libourel, 'On automatic class insertion with overloading', in *Proceedings of OOPSLA'96, San Jose (CA), USA*, special issue of ACM SIGPLAN Notices, 31(10), pp. 251–267, (1996).

	mo= 0..1	mo= *	md= *	to= Te	to= La	to= MAg	to= REAg	td= Ho	to= NC1	to= NC2	to= NC3	to= NC4	to= NC5
lease	x	x	x	x				x	x	x		x	x
buy		x	x		x			x	x	x			x
maintain	x	x	x			x		x		x	x	x	
sell/buy/lease		x	x				x	x		x	x		

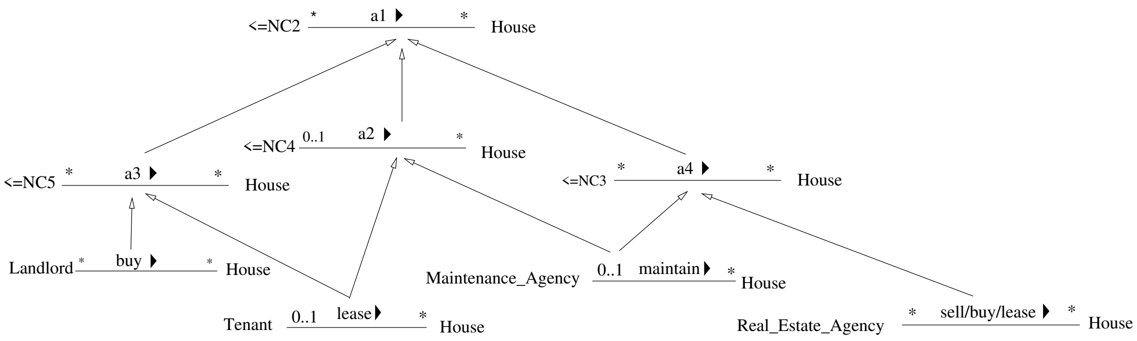

Figure 16. $\mathcal{K}_2^2$ and $GSH(\mathcal{K}_2^2)$ - House Transactions

[8] E. Diday and R. Emillion, 'Treillis de Galois maximaux et capacités de Choquet', *C.R. Acad. Sci. Paris*, **325**(1), 261–266, (1997).

[9] M. Faid, R. Missaoui, and R. Godin, 'Knowledge discovery in complex objects', *Computational Intelligence*, **15**(1), 28–49, (1999).

[10] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, *Refactoring: Improving the Design of Existing Code*, Addison-wesley, 1999. Object Technologies Series.

[11] B. Ganter and R. Wille, *Applications of combinatorics and graph theory to the biological and social sciences*, volume 17 of *The IMA volumes in Mathematics and its applications*, chapter Conceptual Scaling, 139–167, Springer-Verlag, New York, 1989.

[12] B. Ganter and R. Wille, *Formal Concept Analysis, Mathematical Foundations*, Springer-Verlag, 1999.

[13] R. Girard, *Classification Conceptuelle sur des Données Arborescentes et Imprécises*, Thèse de doctorat, Université de la Réunion, 1997.

[14] R. Godin, M. Huchard, C. Roume, and P. Valtchev, 'Inheritance And Automation: Where Are We Now?', in *Object-Oriented Technology ECOOP Workshop Reader*, Lecture Notes in Computer Science. Springer-Verlag, (2002).

[15] R. Godin and H. Mili, 'Building and maintaining analysis-level class hierarchies using Galois lattices', in *Proceedings of OOPSLA'93, Washington (DC), USA*, special issue of ACM SIGPLAN Notices, 28(10), pp. 394–410, (1993).

[16] R. Godin, H. Mili, G. Mineau, and R. Missaoui, 'Conceptual Clustering Methods Based on Galois Lattices and Applications', *Revue d'Intelligence Artificielle*, **9**(2), 105–137, (1995).

[17] R. Godin, H. Mili, G. Mineau, R. Missaoui, A. Arfi, and T.T. Chau, 'Design of Class Hierarchies Based on Concept (Galois) Lattices', *Theory and Practice of Object Systems*, **4**(2), (1998).

[18] R. Godin, H. Mili, G. W. Mineau, R. Missaoui, A. Arfi, and T. T. Chau, 'Design of Class Hierarchies based on Concept (Galois) Lattices', *Theory and Application of Object Systems*, **4**(2), 117–134, (1998).

[19] M. Huchard, H. Dicky, and H. Leblanc, 'Galois lattice as a framework to specify algorithms building class hierarchies', *Theoretical Informatics and Applications*, **34**, 521–548, (January 2000).

[20] M. Huchard and H. Leblanc, 'Computing Interfaces in Java', in *Proc. IEE International conference on Automated Software Engineering (ASE'2000)*, pp. 317–320, 11-15 September, Grenoble, France., (2000).

[21] R. Kent, 'Rough concept analysis: A synthesis of rough sets and formal concept analysis', *Funadamenta Informaticae*, **27**, 169–181, (1996).

[22] S. Kuznetsov, 'Learning of simple conceptual graphs from positive and negative examples', in *Proceedings of the Third European Conference PKDD'99, Prague, Czech Republic*, eds., J. Zytkow and J. Rauch, volume 1704 of *Lecture Notes in Computer Science*, pp. 384–391. Springer-Verlag, (1999).

[23] M. Liquiere and J. Sallantin, 'Structural Machine Learning with Galois Lattice and Graphs', in *Proceedings of the 15th International Conference on Machine Learning*, pp. 305–313. Morgan Kaufmann Publishers, (1998).

[24] G. W. Mineau and R. Godin, 'Automatic Structuring of Knowledge Bases by Conceptual Clustering', *IEEE Transactions on Knowledge and Data Engineering*, **7**(5), 824–828, (1995).

[25] G. W. Mineau, G. Stumme, and R. Wille, 'Conceptual structures represented by conceptual graphs and formal concept analysis', in *Conceptual structures: Standards and Practices*, eds., W. M. Tepfenhart and W. Cyre, volume 1640 of *Lecture Notes in Computer Science*, pp. 423–441. Springer-Verlag, Berlin, (1999).

[26] I. Moore, 'Automatic Inheritance Hierarchy Restructuring and Method Refactoring', in *Proceedings of OOPSLA'96, San Jose (CA), USA*, special issue of ACM SIGPLAN Notices, 31(10), pp. 235–250, (1996).

[27] N. Pasquier, Y. Bastide, T. Taouil, and L. Lakhal, 'Efficient Mining of Association Rules Using Closed Itemset Lattices', *Information Systems*, **24**(1), 25–46, (1999).

[28] Rational Software Corporation, *UML v 1.3, Notation Guide*, version 1.3 edn., juin 1999.

[29] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Modlisation et conception orientes objet*, Masson, 1995.

[30] G. Snelting and F. Tip, 'Reengineering class hierarchies using concept analysis', in *Proceedings of ACM SIGPLAN/SIGSOFT Symposium on Foundations of Software Engineering*, pp. 99–110, Orlando, FL, (1998).

[31] G. Snelting and F. Tip, 'Understanding class hierarchies using concept analysis', *ACM Transactions on Programming Languages and Systems*, **22**(3), 540–582, (May 2000).

[32] P. Valtchev, 'Building Classes in Object-Based Languages by Automatic Clustering', in *Proceedings of the 3rd International Symposium on Intelligent Data Analysis.*, eds., D. Hand, J. Kok, and M. Berthold, volume 1642 of *Lecture Notes in Computer Science*, pp. 303–314. Springer-Verlag, (1999).

[33] P. Valtchev, R. Missaoui, and P. Lebrun, 'A partition-based approach towards building Galois (concept) lattices', *to appear in Discrete Mathematics*, (2001).

[34] R. Wille, 'Restructuring the lattice theory: An approach based on hierarchies of concepts', in *Ordered sets*, ed., I. Rival, pp. 445–470, Dordrecht-Boston, (1982). Reidel.

[35] R. Wille, 'Conceptual structures of multicontexts', in *Proceedings of the 4th ICCS 1996, Sidney (AU)*, eds., P. W. Eklund, G. Ellis, and G. Mann, volume 1115 of *Lecture Notes in Computer Science*, pp. 23–39. Springer-Verlag, (August 1996).

[36] A. Yahia, L. Lakhal, R. Cicchetti, and J.P. Bordat, 'iO2 - An Algorithmic Method for Building Inheritance Graphs in Object Database Design', in *Proceedings of the 15th International Conference on Conceptual Modeling ER'96*, volume 1157 of *Lecture Notes in Computer Science*, pp. 422–437. Springer-Verlag, (1996).