Relational Concept Discovery in Structured Datasets

M. Huchard[a] M. Rouane Hacene[b] C. Roume[b] P. Valtchev[b]

[a]*LIRMM, UMR 5506, CNRS et Université Montpellier 2, 161 rue Ada, 34392 Montpellier Cedex 5, France*

[b]*DIRO, Université de Montréal, CP 6128, Succ. Centre-Ville, Montréal Québec H3C 3J7*

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, knowledge bases, 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. Therefore, several attempts have been made to introduce relations into the formal concept analysis field which otherwise generated a large number of knowledge discovery methods and tools. However, the proposed approaches invariably look at relations as an intra-concept construct, typically relating two parts of the concept description, and therefore can only lead to the discovery of coarse-grained patterns. As an approach towards the discovery of finer-grain relational concepts, we propose to enhance the classical (object $\times$ attribute) data representations with a new dimension that is made out of inter-object links (e.g., spouse, friend, manager-of, etc.). Consequently, the discovered concepts are linked by relations which, like associations in conceptual data models such as the entity-relation diagrams, abstract from existing links between concept instances. The borders for the application of the relational mining task are provided by what we call a relational context family, a set of binary data tables representing individuals of various sorts (e.g., human beings, companies, vehicles, etc.) related by additional binary relations. 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 a method for constructing concepts 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.

Key words: Galois/concept lattices, relational data mining, lattice constructing algorithms, conceptual scaling, relational loops

---

1 Introduction

Formal Concept Analysis (FCA) [14] 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 [35] 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 [25] and software engineering [17,29].

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 [13]) 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 since including relational information [23,21,12]. 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 FCA field and all of them proposed an appropriate redefinition of the Galois connection. However, these approaches exclusively considered the relations to be intra-individual, i.e., not to cross the border of the individual descriptions (e.g., spatial relation between the parts of a whole). Following a similar track, we address the specific problems of processing individuals that are characterized by their relations to other individuals, i.e., in a context in which relations are extra-individual.

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 proved their utility as analysis tools. 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, we present a method for constructing related concepts in the most 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 as an extension of the classical *conceptual scaling*, uses the concepts discovered by the previous steps to refine the object descriptions. The updated object descriptions give rise to richer lattices and the process of scaling/lattice update goes on, until the obtained lattices stop adding new features to object descriptions.

In the next section, we recall the basics of FCA and the conceptual scaling approach to the processing of non binary data. Difficulties with relational information in object descriptions together with our own approach to the construction of concepts that summarize inter-object links are discussed in Section 3. Finally, an example of how our framework applies to UML class diagram analysis is provided in Section 4. Conclusions are drawn in Section 5.

2 Background

The following problem statement generalizes the classical way of introducing the data in Galois lattice construction and FCA (see [14]). The novel element is the presence of relational attributes that link objects from a family of *many-valued* contexts.

2.1 Basic settings

The domain focuses on the partially ordered structure [1], known under the names of *Galois lattice* [1] or *concept lattice* [35], which is induced by a binary relation R over a pair of sets O (*objects*) and A (*attributes*). In the sequel, we follow the standard FCA notation and terminology, except in naming the

[1] An excellent introduction to partial orders and lattices may be found in [8].

basic sets: O (is in **o**bject) is used instead of G (for Gegenstand, 'object' in German) and A (is in **a**ttribute) instead of M (for Merkmal, 'feature' in German). Moreover, Galois and concept lattice are used indiscriminately, thus reflecting their equivalence as well as their independent forging in the lattice domain.

Definition 1 (binary (one-valued) context) *A formal context is a triple $K = (O, A, J)$ where O and A are sets (objects and attributes respectively) and J is a binary relation, i.e., $J \subseteq O \times A$. K is called one-valued (binary) context.*

In Figure 1 (left) is shown a one-valued context where objects are denoted by numbers and attribute by small letters.

Galois connection: The relation J induces two mappings, f and g, that link 2^O and 2^A both ways:

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

For example, $f(13) = ac$.

Galois connections are fundamental constructs in FCA. They are made out of a pair of partially ordered structures provided with two functions that link them both ways. The basic property of a Galois connection is as follows:

Definition 2 (Galois connection)
Let P and Q two partially ordered sets and α and β two mappings such that $\alpha : P \rightarrow Q$ and $\beta : Q \rightarrow P$. The pair (α, β) is said to be a Galois connection between P and Q if the following condition holds:

$$\forall x \in P, y \in Q,\ \alpha(x) \leq_Q y \Leftrightarrow \beta(y) \leq_P x.$$

Clearly, the functions f and g jointly establish a Galois connection between 2^O and 2^A.

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 C^o_K and C^a_K (or simply C^o and 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.

	a	b	c	d
1	X		X	
2		X	X	
3	X		X	
4	X			X
5		X	X	
6		X		X

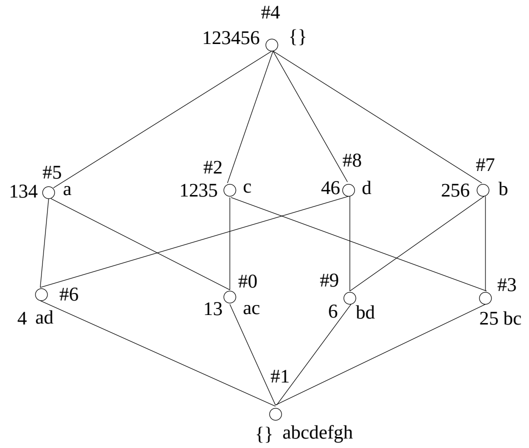


Fig. 1. **Left:** Binary table $K = (O = \{1, .., 6\}, A = \{a, b, c, d\}, R)$. **Right:** The Hasse diagram of the lattice derived from K (also the lattice L^0_H corresponding to the scaled context $Human$ presented in the sequel).

Formal concepts: When considered as a separate entity, a pair of mutually corresponding closed sets $c = (X, Y)$ from C^o and C^a, respectively, is called a *formal concept* in [14] where the set of objects, X, is called *extent* and the set of attributes, Y, *intent*. For example, $(13, ac)$ is a formal concept (closed pair), but $(2, bc)$ is not.

Concept lattice: Furthermore, the set of all concepts, C_K (or simply C), when appropriately ordered, constitute a third lattice, isomorphic to the initial ones. Indeed, C_K can be partially ordered by intent/extent inclusion:

$$(X_1, Y_1) \leq_K (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2 (Y_2 \subseteq Y_1).$$

Clearly, the order $\leq_K$ satisfies the complete lattice properties.

The partial order $L = \langle C_K, \leq_K \rangle$ is a complete lattice with joins and meets as follows:

- $\bigvee_{i=1}^k (X_i, Y_i) = ((\bigcup_{i=1}^k X_i)'', \bigcap_{i=1}^k Y_i)$,
- $\bigwedge_{i=1}^k (X_i, Y_i) = (\bigcap_{i=1}^k X_i, (\bigcup_{i=1}^k Y_i)'')$.

The lattice L_K or simply L is known as the *Galois lattice* of the binary relation [1] or the *concept lattice* of the context K [35]. The lattice L corresponding to the context shown in Figure 1, on the left, is given in the same figure, on the right. For example, in the same Figure 1, the join and the meet of $c_{\#6} = (4, ad)$ and $c_{\#0} = (13, ac)$ are $(134, a)$ and $(\emptyset, abcd)$, respectively.

Galois sub-hierarchy: 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 of the FCA-based framework. Before defining that, we introduce two auxiliary functions that will be relied on in the sequel. They translate the fact that for every object (attribute) there is a concept in the lattice that is extremal for the set of all concepts having that object (attribute). These concepts are called the object (attribute) concepts.

Definition 3 (Object and attribute concepts)
Let $K = (O, A, J)$ a formal context. Given an object o, the concept $(\{o\}'', \{o\}')$ is called the object concept of o. Similarly, given an attribute a, its attribute concept is $(\{a\}', \{a\}'')$. Two functions are thus defined mapping objects/attributes to concepts:

- $\mu : O \to C_K$, $\mu(o) = (\{o\}'', \{o\}')$,
- $\nu : A \to C_K$, $\nu(a) = (\{a\}', \{a\}'')$.

It is a well-known fact that $\mu(o)$ is the smallest concept having o in its extent, and, dually, $\nu(a)$ is the greatest one having a. When taken out of the lattice, both sets of concepts $\mu(O)$ and $\nu(A)$ form a specific sub-order that plays the role of a normal form for the information embedded in the context. Basically, the sub-order, called the "Galois sub-hierarchy", is a compressed representation of the lattice which encodes in a non-redundant way all the information that is necessary for the recovery of the complete lattice. The following definition is borrowed from [19]:

Definition 4 (Galois sub-hierarchy)
Let $K = (O, A, J)$ a formal context and following sets of concepts: $C^O = \{\mu(o) \mid o \in O\}$; $C^A = \{\nu(a) \mid a \in A\}$. The Galois sub-hierarchy $GSH(K)$ is the sub-order of L made out of the set $C^O \cup C^A$ and the restriction of the lattice order to that set, denoted by $<_{|GSH}$.

Construction of lattices Lattice construction from contexts has been a challenge since the very early days of FCA. The problem is a hard one since in the worst-case, there can be exponentially many concepts. However, in practical cases, only a small number of concepts do occur, so it makes sense to look for methods that discover and, if necessary, hierarchically organize them, in an efficient manner. There is nowadays a large variety of algorithms dedicated to the computation of either the set of all concepts or the entire lattice, i.e., concepts plus order [2] . A major distinction among these algorithms lays in the way they acquire input data. According to a classical dichotomy, batch algorithms consider all the data to be completely known beforehand. In contrast, on-line algorithms allow small changes in the data to be propagated

[2] An algorithmic study is beyond the scope here, hence the reader is referred to [22].

to the final result, i.e., the concept lattice, without starting from scratch. Thus, they can be used to simulate batch lattice construction by a sequence of object/attribute additions to an initially void context (also called incremental construction).

2.2 Non binary data and scaling

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

Some researchers explored the possibility of translating the various kinds of data back to binary variables, e.g., through *conceptual scaling* [18,13]. 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., probabilistic [10] or rough sets [20]. In particular, in FCA [14], non binary (e.g., numerical, ordinal, categorical, etc.) attributes are introduced via a many-valued context.

Definition 5 (many-valued context) *A many-valued context is four-tuple $K = (O, A, V, J)$ where O and A are sets (objects and attributes respectively), V is a set of values, and $J \subseteq O \times A \times V$ is a ternary relation with member tuples $(o\ a\ v)$ being interpreted as "the object o has a value v for the attribute a".*

The following table presents a sample many-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.

	o_1	o_2	o_3	o_4	o_5	o_6
age	25	28	22	26	33	35
work	4	2	1	8	4	10

In order to deal with many-valued context $K = (O, A, V, J)$, it is first transformed into an equivalent one-valued, or binary, context K^d called the *derived* context. For this purpose, FCA includes a specific set of scaling techniques that echo the classical discretization and scaling techniques used in statistics and later on in data mining.

Conceptual scaling [13] provides a complete framework for transforming any many-valued context into a binary one. The process of the scaling starts with the determination of the scale for a non binary attribute.

Scales: The scales correspond to separate dimensions of object descriptions. Basically, they represent hierarchies of useful abstractions on the dimension, called scale concepts. Initially, each scale S_a corresponding to an attribute a is assigned a scale context $K_a = (V_a, P_a, J_a)$, where V_a is the set of values of a, P_a contains important properties of these values are formal attributes, denoted by a^s_j and also called scale attributes, and $J_a \subseteq V_a \times P_a$.

With respect to the nature of the attribute, e.g., unordered versus totally or partially ordered, various standard scales are distinguished. Although depending on the nature of the attribute, properties invariably correspond to subsets of the co-domain of the attribute, $cod(a) = V_a$. In the sequel, we shall denote the extent of a scale attribute a^s_j in terms of values as $a^{s\prime}_j$. For example, in ordinal scales the properties are predicates that compare a value to a particular constant from the domain (e.g., "age $\geq$ 20"). The scales may be dependent of the domain knowledge, for example, domain ontologies can be used.

The scale lattice L_a corresponding to the context K_a provides a complete structuring of the set of all possible values into meaningful subsets. The concepts in L_a, called *scale concepts*, represent these subsets in an intensional manner, i.e., through conjunctions of properties (e.g., "age $\geq$ 20 and $\leq$ 27"). In case of a huge number of possible values in $cod(a)$, the lattice L_a may be impossible to obtain in a direct way, i.e., from K_a. However, the natural inclusion order between the scaling attribute extents can be used to recover the structure of L_a. In this case, the formal context used as a starting point is $K^f_a = (P_a, P_a, \leq_f)$, where f is defined by $a^s_j \leq_f a^s_l$ iff $a^{s\prime}_j \subseteq_f a^{s\prime}_l$.

Figure 2 shows possible scale lattices of the many-valued attributes *age* and *work experience* from the context *Human*, whereby a simple encoding through mutually exclusive scale attributes (with thresholds of 27 for *age* and 5 for *work* experience) has been used.

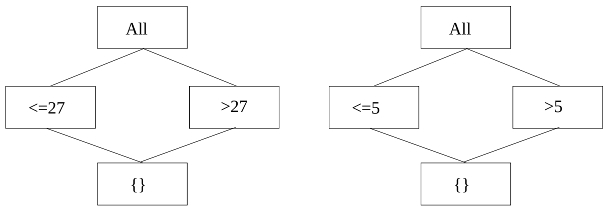


Fig. 2. Scales for *age* and *work experience*.

It is noteworthy that scale concepts for all scales of a many-valued concept jointly constitute the description space for the context's concepts.

Scaling: The scaling of the context K is an immediate step: for each object the value of the original non-binary attribute a is replaced by a set of binary ones, a_i^s. An object o from K gets a_i^s whenever $a_i^{s\prime}$ includes $o.a$ (the value of a for o). The result is a scaled context K^s which may be seen as the concatenation (*apposition* as in [14]) of a set of smaller contexts K_a^s, one per attribute. The context in the following table, say *Human*^{d}, was obtained by scaling *Human* with the scales depicted in Figure 2.

	short	o_1	o_2	o_3	o_4	o_5	o_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

The reader may check that the context *Human*^{d} is identical to the context in Figure 1.

From scaling to lattices: The lattice L^s for the scaled context K^s may be constructed directly from it using any dedicated procedure. Of course, the exact form of L^s depends on the scales that have been used for the attributes. The lattice of the scaled context *Human*^{d} may be seen in the Figure 1.

An interesting fact about scales is that they can be used for visualization (as in the Toscana system [34]). Moreover, the lattice L^s may be obtained directly from the set of lattices L_a^s corresponding to the various attributes of the initial context K using a slightly modified version of the Assembly algorithm described in [33]. To that end, an additional step would be necessary which amounts to translating scale concept extents, (these may be either values or scale attributes) into sets of formal objects from K. Thus, for each concept c_a^s of intent $Y_a \subseteq P_a$, the extent of c_a^s in K, say Y^I, is $\{o | \forall a_i^s \in Y, o.a \in a_i^{s\prime}\}$. where I is the incidence relation in K^s). If all scale concepts $c_a^s = (X, Y)$ in any scale lattice L_a^s are now replaced by $(c_a^s)^I = (Y^I, Y)$, then the lattice L^s is simply the apposition-based product of the modified scale lattices The reader is referred to [33] for details, however, due to the simplicity of our example, the links between the scale lattices in Figure 2 and the lattice in the Figure 1 are easy to notice.

The advantage of this second view on how lattices of scaled contexts emerge is that it explains why any combination of scale attributes from a particular scale S_a that are met in the intent of a concept c from L^s is necessary an intent of a scale concept in L_a^s. This basic fact means that there is always a unique concept from L_a^s that "describes" the set of values of a met in the objects of

Relations in contexts: Yet different attribute types represent links from an individual to other individuals. Such links may carry domain-relevant semantics, e.g., kinships or spatial relations, or just encode technical information, e.g., hyper-links in textual databases or on the Web. Inter-individual links represent important aspects of the data and therefore may help the formation of meaningful abstractions, provided that these aspects could be successfully integrated in the analysis process (see next section). Previous studies on relational data mining have underlined the difficulties arising with such datasets, mostly due to the fact that objects in the data have both their own structure, i.e., attributes, and additional relational part that extends to other objects [11]. More specifically, classical problems faced by a relational analysis include the processing of one-to-many relations and the resolution of circular dependencies among object descriptions.

Imagine now that the objects of our previous example are described by an additional feature that specifies their respective spouses. The table below presents a possible valuation of the relational attribute *spouse* that completes the *Human* context into *Human$_s$*.

	o_1	o_2	o_3	o_4	o_5	o_6
spouse	o_2	o_1	o_4	o_3	o_6	o_5

Taking into account the *spouse* relation for concept construction undoubtedly widens the range of potentially interesting generalizations from the raw data. For example, a target concept to look for might be described as: “27 years old or younger and being married to individuals with at most 5 years of work experience”. This description clearly covers the objects o_1 and o_4 of our example. However, the processing of relations in FCA has a price. Before presenting our framework for constructing concepts on formal objects with attributes and relational links, called relational concept analysis (RCA), we explore the extensive literature on processing data in richer-than-propositional formats within FCA.

2.3 Relations in formal concept analysis

Different research streams within FCA have attempted to overcome 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 function-free predicate-logic languages

(Datalog) [4] and conceptual graphs [24]. 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 3). 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 [23,21] 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 [30]. 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.

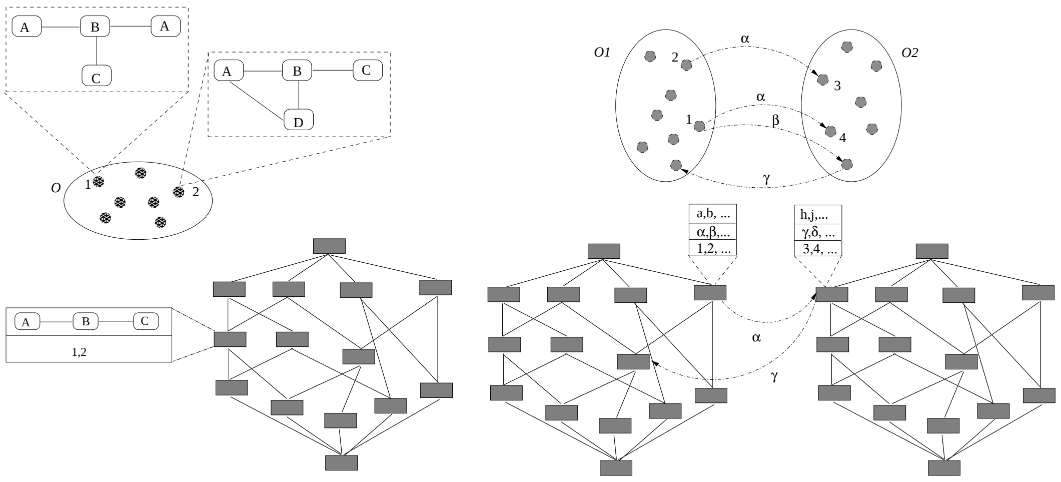


Fig. 3. **Left**: Classical approach for processing relational data in FCA: relations remain within concept descriptions. **Right**: In our approach, relations between individuals are reflected by the intentional descriptions of the discovered concepts.

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.* [12] (which generalizes earlier work of Wille [36]), 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 3 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.

In the sequel, we present an extension of the many-valued context framework to include relational information and show how the resulting object descriptions can be scaled back to binary attributes without a substantial loss of information.

3 Mining formal concepts out of objects with links

A first step to the processing of relational data is the definition of appropriate representations so that the necessary tools could be added. At a preliminary step we provide some practical arguments in favor of including relations into the analysis process and describe an intuitive way of doing that.

3.1 Integrating relations into concept intents: a step-by-step approach

The goal of the following development is to illustrate rather than to formally define the way concept relations are abstracted from inter-object links.

First, a basic fact from the object-oriented paradigm says that objects are identified within a software system, by means of a unique object identifier (OID). This identifier is typically used for link specification, via what is called an *object-valued* attribute. Thus, a flat set of objects offers tiny possibilities for abstraction: there are no commonalities among objects that are to be accessed by looking at their respective OIDs. Consequently, the only universally available comparison function for objects is the identity predicate. The work of [12] pushes the abstraction capacity of the identity to its extreme: objects are described by the OIDs of related objects, further called links, whereas the intersections between sets of OIDs constitute the only available generalization operators. The strategy clearly fails with a bijective mapping such as the one made out of the `spouse` links, but would also perform poorly on datasets where the sharing of linked objects is a rare phenomenon.

Instead of comparing objects with identity, a classical data mining method would look for a conceptual hierarchy that clusters the objects into meaningful groups which would further be addressed in the discovered patterns. The operation of replacing a concrete value or object by a generic entity, known as "climbing the generalization hierarchy", is close in spirit to conceptual scaling in that it allows the described objects to share features that are not in their initial description, but are inferred by inductive generalization. This necessarily pops up new and potentially interesting patterns that were inaccessible before: for instance, the concept #2 in Figure 1 could not have been discovered directly from the numerical data.

When applying the same reasoning to object-valued attributes, here the `spouse` links, one may observe that when they are left aside in the lattice construction not a single object couple can be grouped into a concept because of their respective links being similar. To illustrate that phenomenon, let us examine the objects o_2 and o_5 (objects of the context $Human^d$ or equivalently objects in Figure 1 with addition of the prefix o). Observe that these share the same encoding in the scaled context of the example, i.e., both are indistinguishable, a fact that actually brought them into the same object-concept #3. However, this fact is absolutely unaccounted for in the lattice in Figure 1 where, for example, the respective spouses do not share an attribute and thus join only at the extent of the top concept. However, being married to persons of such a similar profile should be enough to grant them a certain degree of similarity, at least more than o_3 and o_6 who not only share no attributes, but also are married to persons that are further completely incompatible (o_4 and o_5 have nothing in common). Conversely, the instance of being linked to objects of so much diverging characteristics would have permitted to separate o_2 and o_5, at least at the lowest level of the lattice.

The above observations point at the limitation of the classical FCA grouping mechanism that fails to "transmit" similarity/dissimilarity along the inter-object links. A remedy could come from a hierarchy of meaningful groups that are used to tag the link objects much like the way scale concepts tag attribute values. Following this analogy, the object will be tagged by a group if and only if it belongs to that group. With groups organized in a hierarchy, the more two objects are similar, the more "tags" they share, and *vice versa*. Furthermore, the tags of an object that come from its concepts are "transferred" to objects that point to the initial one, e.g., the tags of o_2 will be transferred to o_1 via the `spouse` link. Spelled differently, an object receives an encoding made up of all attributes representing the tag concepts of its link(s). For instance, Figure 1 shows the lattice of the example dataset which is obtained from the scaled context and without any consideration of the links. The lattice could be used as a basic conceptual hierarchy (what is actually is) to support generalization. Thus, the object o_1 would receive the tags for concepts to which o_2 belongs, i.e., #4, #2, #7, and #3.

At the concept level, i.e., when shared features are sought to form intents, this means that a concept receives a tag attribute whenever all of its member objects share that attribute. Or, equivalently, they have their link(s) in its extent. Thus, in the above example, o_1 would share the tag for #3 with o_6.

Here is the place to observe that there is no standard way of constructing a generalization hierarchy out of set of objects so that it could be later used for scaling purposes. However, formal concepts and their lattice are aimed at exactly providing a set of useful abstractions, so the concept lattice is the best candidate for a structure to support the scaling. Other interesting structures include various upper parts of the lattice (iceberg lattices) and the GSH. For our own study, we decided to focus on the complete lattice to simplify the presentation. However, there is still a technical difficulty that need to be adressed with constructed lattices serving as scaling structures. Indeed, in the case of Human, the recursive `spouse` links force the same object set to be taken both as the analysis set and the scaling basis. This propagates to the respective lattice which is to be used both as a conceptual structure and as a scaling hierarchy. The object scaling track thus seems to lead to a deadlock, but mutual dependencies actually resolve by a suitable iterative process.

To clarify this claim, let us first recall that our goal was to integrate spouse links into the features of the objects so that they can be "shared", at least at some abstraction level. As an inital step of the procedure, we consider the concepts of the lattice in Figure 1 and assign a tag attribute to each, except for the top and the bottom concepts[3] . To make our constructs more explicit, the names of the tag attributes are made out of a prefix "sp" indicating that the tag comes via the `spouse` link (as there may be several such links) and a suffix indicating the index of the tag concept in the target lattice. For example, the concept #7 of the lattice will be assigned the tag attribute $sp : 7$. Based on these new attributes, the description of the dataset objects is completed, by observing the rule which makes tags move from concept members to other object that point to them. Thus, the attribute $sp : 7$ will further go to the objects o_1, o_5, and o_6. Conversely, according to our previous remark, the object o_1 will "receive" the new attributes $sp : 2$, $sp : 3$ and $sp : 7$. The entire encoding is described in the following table.

[3] Universal and null attributes have no impact on the structure of the lattice.

tag	1	2	3	4	5	6
$sp:5$		X	X	X		
$sp:2$	X	X		X		X
$sp:8$			X		X	
$sp:7$	X				X	X
$sp:6$			X			
$sp:0$		X		X		
$sp:9$					X	
$sp:3$	X					X

The above table, when added to the initial proper features of the objects yields an extended context. The new context shares with the previous one the entire object dimension O, as well as the fixed part of the attribute dimension ($A_0 = a, b, c, d$). Thus, all the concepts from the initial context will remain valid concepts in the extended one, except for some additional, relation-based attributes that may need to be inserted in the intent. Therefore, to simplify the tracking of concept between lattices of various extensions, we shall further identify them with respect to their extents.

When we go back to concept discovery, after the encoding of objects with a scale corresponding to the initial lattice, another round of concept construction can be performed, this time factoring not only on attributes but also on shared tags, i.e., concepts to which links belong. The lattice of the new context, shown in Figure 4 is bigger than the initial one, with more than a dozen new concepts. To distinguish both lattices, they will be further denoted as L_H^0 and L_H^1, where superscripts follow the order of their construction.

It is interesting how the new concepts summarize the links with respect to the abstractions from the initial lattice. For instance, the new concept #23 encompassing o_1 and o_6 has only been created because of the shared tag attributes. Moreover, these are the attributes that point at all the concepts in L_H^0 including both the respective links o_2 and o_5. It is noteworthy, that whereas the new concepts, concepts whose extents are absent in L_H^0, necessarily include a tag attribute in their intent, they still may have some attributes from the initial set A. For instance, the concept #26 with an extent $\{3, 5\}$ has both c and $sp : 8$. In addition, tag attributes have allowed for all the objects to be separated even in the lowest level of the lattice.

The above process could easily go on with a second lattice-based scaling. Indeed, as the available information about the conceptual structure of the set O has evolved since the initial step, it is natural to look for an adjustment in

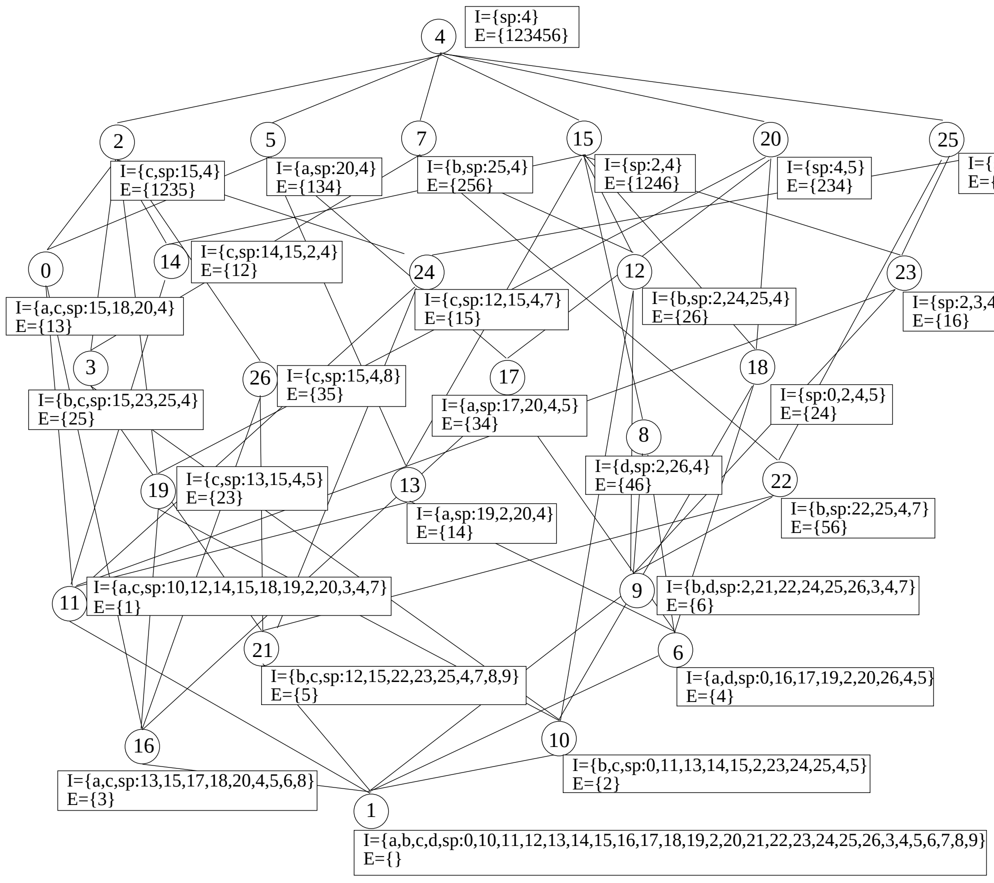


Fig. 4. The lattice L^1_H corresponding to the second step of Multi-Fca() on the RCF $R^1_H = (\{Human\}, \{spouse\})$. Concept intents in L^1_H include both conventional attributes and the inter-concept relations induced by *spouse*. $sp = i, j, ... \; k$ stands for $sp : i$,$sp : j$, ... $sp : k$.

the extracted lattice. Surprisingly enough, the encoding with respect to L^1_H does not lead to changes in the target lattice structure, i.e., to new concept extents. Of course, the second scaling step introduces a whole range of new tag attributes that actually replace those from the previous scaling. However, the new attributes fail to generate previously unseen closed set of objects. To summarize, the resulting third lattice L^2_H (not shown here for space reasons) is isomorphic to L^1_H and this halts the iterative construction process.

Another key aspect of relational attributes is the existing redundancy in concept descriptions where the presence of some attributes simply entails others. This phenomenon could be observed already in L^1_H: the intent of concept #18 includes the tags $sp : 0$, $sp : 2$, $sp : 4$ and $sp : 5$. However, the corresponding concepts are linked by generalization relationship: #0 $\leq^0$ #5 and #0 $\leq^0$ #2 and therefore, whenever a concept has $sp : 0$, it will necessarily have $sp : 5$ and $sp : 2$ as well. This fact is easily generalized to the rule saying that if an attribute with a target concept c is present in the intent of a concept c_1, then all the attributes associated to super-concepts of c will also be present. This follows trivially from the set-based semantics of the formal concepts: in the technical language of the domain, the former attribute is said to be "implied" by the latter one. This kind of redundancy may easily lead to an explosion in the size of concept intents due to the large number of tags that may be shared by the members of the most specific concepts. Therefore it requires further work on each concept in order to remove unnecessary tags, i.e., those associated to non-minimal concepts. In other terms, the concept descriptions need to be reduced to keep only a minimal set of tag attributes. For instance, the intent of the concept #18 after simplification would become $\{sp : 0\}$.

In the case of `Human` with `spouse`, the lattice has a nice property: there is only one non-minimal concept in all those sets and hence only one attribute remains in the intent after simplification. Thus, the semantics of the inter-concept relation `spouse` remains close to the semantics of relations in the E-R conceptual models, i.e., given two entities A and B, and a relation r, $A-r-B$ means that:

- every A element is linked via r to at least one from B (if cardinality constraints do not state otherwise),
- all links r of an individual a from A are in B.

The lattice that will be presented as a final result of the above relationally-aware concept analysis is a highly complex structure in which two sorts of relationships among concepts need to be visualized: the `sub-concept-of`, or specialization, relationship and what we call the inter-concept relations abstracting from inter-object links. In the above example, the relations connect, in a graphical sense, a source concept to a single destination concept, therefore they could be visualized as arrows. Figure 5 shows the final lattice where the `sub-concept-of` relationship has been skipped to avoid overloading the drawing.

Even if such a UML-like representation of relations seems natural, it is not always possible to enforce the existence of a minimal concept. In our example, such existence is the joint consequence of two important choices:

- to work with the entire lattice instead of taking substructures,
- to select a relationship that represents a one-to-one function.

Another important property of inter-concept tags, or *relations* as we shall call the reduced version of tag sets, is that relations respect specialization among concepts since they map a pair of comparable concepts to a pair of identically comparable ones. More technically speaking, relations are *monotonous* in the sense that if $c_1 \leq c_2$, $c_1 \rightarrow_r c_3$ and $c_2 \rightarrow_r c_4$, then $c_3 \leq c_4$.

This fact can be proven easily using the semantics of the `sub-concept-of` link, the status of c_3 and c_4 as minimal target concepts in c_1 and c_2, respectively.

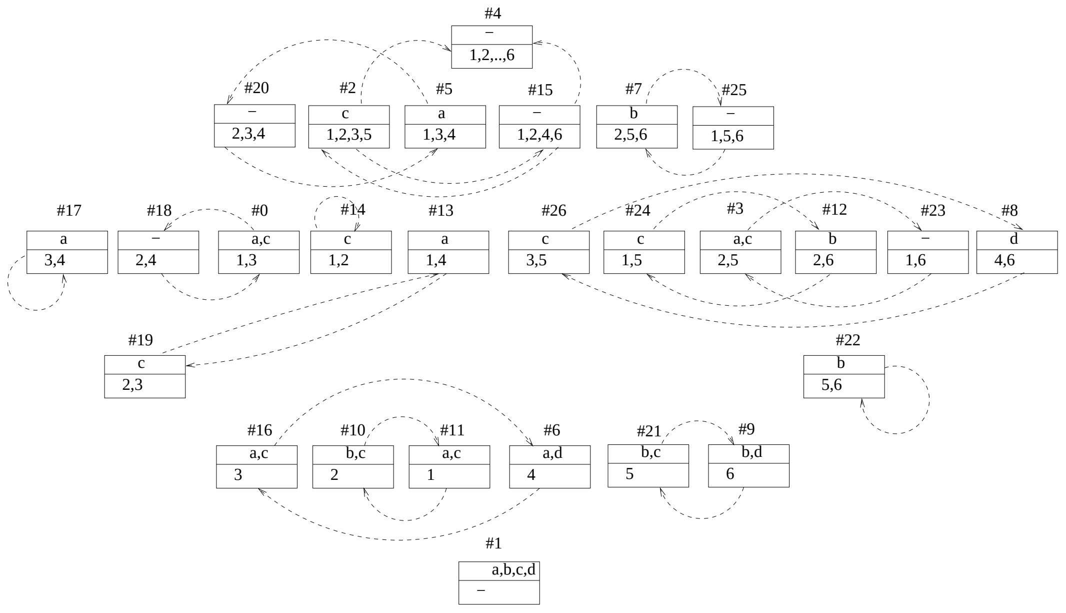


Fig. 5. The lattice L_H^1 without the lattice order. Concept intents include only initial attributes whereas the inter-concept relations induced by *spouse* are drawn as arrows. $sp : i, j, ..k$ in labels stands for $sp : i$, $sp : j$, ... $sp : k$.

Should one of the above constraints be relaxed, it may happen that several minima occur in the set of target concepts. We shall clarify these cases further in the text.

In the following, we formalize the above intuitive notions related to scaling on object dimensions and clarify the underlying computational mechanisms.

3.2 Combining formal contexts with relations

As relations usually link objects of different kinds, e.g., human beings versus owned vehicles, we introduce a data description framework that is of higher granularity than a single context. Thus, what will be further referred to as a *relational context family* (RCF) is a set of contexts provided with a set of relations. Each relation is binary and maps the objects of a particular type, i.e., of a specific context, to (sets of) objects from another type (context).

Definition 6 (Relational context family)
A relational context family R^s is a pair $(\mathbf{K}_R, A_R)$ where $\mathbf{K}_R$ is a set of s many-valued contexts $K_i = (O_i, A_i, V_i, J_i)$ and A_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 $\alpha_j : O_r \rightarrow 2^{O_q}$.

The mappings $dom : A_R \rightarrow \{O_i\}$ (domain) and $cod : A_R \rightarrow \{O_i\}$ (co-domain) are defined for a RCF R^s in the following way: for any $\alpha_j : O_r \rightarrow 2^{O_q}$,

$dom(\alpha_j) = O_r$ and $cod(\alpha_j) = O_q$. Moreover, the set of all relations of a given context within a RCF, is computed by the function $rel : \mathbf{K}_R \rightarrow 2^{A_R}$, with $rel(K_i) = \{\alpha | dom(\alpha) = O_i\}$. For instance, consider a RCF $R^1_H = (\{Human\}, \{\alpha\})$, where the relation $\alpha : O_{Human} \rightarrow 2^{Human}$ models the `spouse` links between persons (obviously, $\alpha : O_{Human} \rightarrow O_{Human}$ holds). In the above example, we set α to the following set of object-value pairs:
$\{(1, \{2\}), (2, \{1\}), (3, \{4\}), (4, \{3\}), (5, \{6\}), (6, \{5\})\}$.

It is noteworthy that a RCF offers data representation facilities similar to the entity-relationship (E-R) model [5] which is one of the most popular conceptual models nowadays. Indeed, contexts are comparable to entities, with their formal attributes corresponding to entity attributes, and objects to concrete entities. Relations in our framework are in turn the equivalent of relationships in the E-R model. Similarly, a parallel could be drawn with structural models in UML [2], i.e., the class/object diagrams, but at a level where each class corresponds to a context and each association to a relation. This is not to be mixed up with the specific modeling of an UML class diagram as an RCF that we use for refactoring purposes (see Section 4), in which UML classes and associations are seen at the meta-level of UML and thus become the formal objects of two distinct contexts.

Given a relational context family R^s, our aim will be to construct s lattices of formal concepts L_i, $(0 \leq i \leq s)$, one per context 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 relation α that maps O_r into 2^{O_q} for some r and q. Let us also limit α to singleton values, i.e., force it to be a one-to-one mapping. Intuitively, such α could be interpreted as a particular many-valued attribute and imagine a scale S_α, call it *relational scale*, where abstractions that summarize the properties of concrete objects from O_q are used as scale attributes. Recall that we look for a compact characterization of the values of α in a concept c from the target lattice L^s_r, i.e., an expression that summarizes their common properties. For convenience reasons, this set will be referred to as the image of c or the co-domain of α restricted to the extent of c, and it will be denoted, somewhat loosely, as: $\alpha(c) = \{o \in O_q | \exists \bar{o} \in O_r, o = \alpha(\bar{o})\}$. As with a conventional scale, $\alpha(c)$ will be described by the combination of scale attributes shared by the objects in $\alpha(c)$.

Another straightforward solution could be to use the attribute set A_q as encoding scale attributes. In this case, and with $cod(\alpha)$ being O_q, the scale lattice will be isomorphic to L^s_q, the lattice of the scaled context K^s_q. In this case, any combination of attributes that characterizes particular $\alpha(c)$ is, as we mentioned previously, an intent of a concept in L^s_q. Let c_q be such that for any a_q in $Intent(c_q)$, $\alpha(c) \subseteq a'_q$. This means that c can be "described" along the dimension α by the concept c_q, denoted $c \rightarrow_\alpha c_q$. When interpreting such a conceptual description, i.e., intent, this actually means that:

- (i) for any object o in the extent of c, the links of o of type α are all members of the extent of c_q ($\alpha(c) \subseteq Extent(c_q)$),
- (ii) c_q is the smallest concept in L^s_q that satisfies the previous property $\alpha(c)'' = Extent(c_q)$,
- (iii) all superconcepts of c_q satisfy the property (i) by extension inclusion.

For example, let us take our previous concept of "less-than-27-year-old with less-than-5-year-experienced spouses" and consider it as a scale concept for the relation *spouse* in the $Human^d_s$ context. As the description covers the objects o_1 and o_4, it will be linked by *spouse* to the concept over $Human^d_s$ whose extent is exactly the set of respective spouses, i.e., o_2 and o_3, whenever such concept exists. The situation is depicted in the following Figure 6.

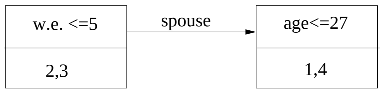


Fig. 6. Example of a concept linked by the *spouse* relation to another concept.

Hence, the scaling attributes may be replaced in the description of the concept c from L^s_r by a reference to the concept c_q (or, later on, by a symbolic name such as "potential clients" or "bad loan payers").

To sum up, in case of one-to-one relational mapping α the links between objects may be "summarized" by inter-concept relations much in the same way concepts summarize objects. Such representation is close to the conceptual models E-R and UML that were mentioned previously. In the sequel we shall examine extensions to more realistic relation cases, in particular, to one-to-many and/or circular relations.

3.3 Scaling arbitrary relations

Given a relation α, *a priori* one can use as scale attributes for α any set of properties A^s_q, possibly disconnected from A_q, the attributes of the context K_q. Independently from L^s_q, A^s_q may induce on its own a non trivial hierarchical structure on O_q ($O_q = dom(\alpha)$), i.e., a directed acyclic graph (DAG) of subsets of O_q that are ordered by inclusion. Call this graph the scale hierarchy $H = (A^s_q, \subseteq)$. For example, we may decide to use a scale for *spouse* that separates persons only with respect to their age, i.e., with $A^s_q = \{age\}$.

Another possibility, used in the application described in Section 4, is to use the so called *Galois sub-hierarchy* [9] of the context K^s_q which provides a compact representation of the entire information encoded by K_q^s. In this case, the construction of L_q^s may become redundant with respect to the domain structuring encoded in H (unless some additional constraints require this construction). Provided that this structuring is sufficient, then the only real scale concepts that will be considered are the closures of each scale attribute.

If a single scale concept needs to be associated to a concept c from the lattice L_r^s, this would force the hierarchy H to be a lattice (but not necessarily isomorphic to L_r^s). In all the remaining cases, H could be a general partial order, which means there will be concepts c from L_r^s whose descriptions include more than a single element for the dimension α. Nevertheless, according to a previous remark, there will always be a minimal set of scale concepts from H which characterize $\alpha(c)$ in a non-ambiguous way.

A suitable relational scale for α is any lattice L_α in which the set of objects in concept extents includes O_q (intuitively, the concepts of L_α represent useful abstractions over O_q). It is noteworthy that the lattice L_α can be the lattice L_q of the context K_q. Thus, L_α can be obtained by an explicit construction (as L_q) or be given beforehand, or even by combining a post-processing of L_q via a user interaction. However, in order to ensure the isomorphism between the partial order provided by the user and the lattice obtained through scaling (L_a^s), the manual order should be at least an upper semi-lattice (with bottom possibly missing).

When the relation α is one-to-one, the scaling replaces the unique value $\alpha(o)$ by a scale concept c_q, from the scale hierarchy H which is occasionally isomorphic to the lattice L_q^s. However, with one-to-many relations, also called many-valued attributes (e.g., a relation *friend-of*), $\alpha(o)$ is a subset of O_q. Thus, scaling requires an effective encoding scheme on $P(O_q)$, i.e., with encodings for sets as simple values and for sets of sets as images of concepts c from L_r^s.

If the single-scale-concept constraint holds, then the only possibility is to choose a conservative scheme that looks for scale concepts which include the entire set $\alpha(o)$ in their extent. Recall that in this case the scale hierarchy H should be a lattice, so that for a given o from O_r, the collection of all scale concepts including the entire set $\alpha(o)$ has a minimal element (the meet). For convenience, suppose that H is isomorphic to L_q^s. In this case, $\alpha(o)$ will be encoded by all the scale concepts c_q^s such that $\alpha(o) \subseteq Extent(c_q^s)$. Moreover, any family $\alpha(X) = \{\alpha(o) | o \in X\}$ will be encoded by the intersection of the sets $\alpha(o)$:

$$\bigcap_{o \in X} \{c_q^s | \alpha(o) \subseteq Extent(c_q^s)\}.$$

The resulting scale follows the same pattern as the one-to-one scaling over O_q and therefore guarantees the existence of a unique minimal scale concept that represents $\alpha(X)$:

$$Extent(c_q^s) = (\bigcap_{o \in X} \alpha(o))''.$$

The scheme replicates the way UML represents one-to-many associations where the target is a single class. In UML, the multiplicity is provided as a separate descriptor, a principle that can easily be adopted in our framework.

Alternatively, when the structure H is a general partial order, less conservative encoding schemes may be applied. In them, α values for objects or for sets of objects need only a non-empty intersection with the extent of a scale concepts. One of the possibilities, used in the application from Section 4, is as follows:

- $\alpha(o)$ is encoded by $\{c_q^s \in H \mid Extent(c_q^s) \cap \alpha(o) \neq \emptyset\}$,
- $\alpha(X)$ is encoded by $\bigcap_{o \in X} \{c_q^s | \alpha(o) \cap Extent(c_q^s) \neq \emptyset\}$.

Clearly, the whole scaling procedure amounts to a gradual refinement of object descriptions. Relational scaling brings new attributes which are nothing more than new closed object sets in the Moore family on O_r (the domain of the relations). These sets are computed as the images by α^{-1} of the scale concepts. In the following, we extend the above single-relation principle to the case of a RCF where a set of relations connects objects from various types.

3.4 Processing a RCF

Relational scaling may be applied for several different purposes and under various circumstances. In particular, the set of scale concepts may be the entire concept lattice of the underlying context or, alternatively, a restricted subset of it, or even a completely independent hierarchy of object sets. In our development, we focus on the most general way of defining the problem of lattice construction within a RCF. Thus, without any *a priori* knowledge on scaling concepts, all the necessary scales are assimilated to the concept lattices of the underlying contexts and therefore built from scratch.

3.4.1 Problem statement

In order to reflect the multiple and inter-related tasks of lattice construction that are necessary in order to analyze an entire RCF, we suggest a generalization of the classical lattice construction problem, called MULTI-LATTICE:

Given : A relational context family $R^s = (\mathbf{K}_R, A_R)$ with s contexts $K_i = (O_i, A_i, V_i, J_i)$.
Find : A set of s lattices $\mathbf{L} = \{L_i\}_{0 \leq i \leq s}$ such that each L_i corresponds to a binary context K_i^d derived from K_i whereby the relational scale used for an

attribute α in $rel(K_i)$ is isomorphic to the lattice L_j where $cod(\alpha) = O_j$. Formally, the target lattice L_i is such that for any concept c from L_j, the set $\alpha^{-1}(c) = \{o \in O_i | \alpha(o) \subseteq Extent(c)\}$ is a valid extent in the lattice L_i.

It is noteworthy that in the case of general partial orders as target hierarchies, the above condition must be combined with any additional condition on the structure of these orders. Technically speaking, for the set of contexts of a RCF, the necessary relational scaling imposes some constraints on the order in which contexts can be processed. Indeed, given a relation $\alpha : O_r \rightarrow P(O_q)$, the scale concepts on K_q must be discovered before the processing of K_r starts, at least prior to the scaling with respect to α. Globally, this means that the RCF should be processed in an order that respects dependencies among contexts induced by relational attributes. To that end, the RCF might be seen as a graph with contexts as vertices and relations as oriented edges. Successful order may then be discovered by a topological sort.

3.4.2 Coping with circularity

Obviously, with some circuits in the graph structure, this straightforward strategy would not work. Intuitively, any strongly connected component of the graph should be dealt with as a whole since no reasonable order can be established within it. The underlying difficulties have been illustrated in the section 3.1, in particular the need for constructing concepts that are later on used as abstractions on the same object set in order to find further and more precise concepts.

We stick to our simplified example of a RCF with single context K and single relation α. A simple solution of the problem seems to be to use the formal attributes in A as scale attributes for the scale S_α, i.e., to ignore relations at first and feed them in later. The resulting scale concepts will be first-class concepts from the target lattice L^s we are looking for since their extents are closed sets of objects from O. However, translating an extent X via α to its counterpart $\alpha^{-1}(X)$ may well result in a set that is not closed in the first scaled version of K. This means that the lattice corresponding to K scaled this time with α is not isomorphic to the initial lattice where α is ignored. It is easy to see that new concepts will be added by the introduction of α in the scaling and that they represent a finer version of the structuring of O into abstractions. Unfortunately, they are not reflected by the scaling itself, since they emerge as the product of the process. Taking them into account requires a further step of re-scaling with the new and richer set of scaling attributes, a process that may well go on indefinitely.

Fortunately, this is not the case. To clarify the termination point, suppose the target lattice L^s is available. Observe that L^s is both the lattice of the scaled context K^s (including a part scaled over α) and the scale lattice of the scale S_α on K. In other words, any further scaling of K with L^s would not change the structure of the lattice corresponding to the resulting context, i.e., it will remain isomorphic to L^s. Alternatively, the Moore family on O will not evolve under further scalings. Notice now that L^s is a fixed point of the (implicit) lattice operator that maps the argument lattice to its "extension" with the scale S_α where the same argument lattice is the scale hierarchy and therefore is filtered backward through α^{-1}.

3.4.3 Multiple lattice constructions within a RCF

Following the above reasoning, one can easily design a procedure that constructs the lattice L^s in a step-wise manner. The method proceeds by successive lattice constructions and scalings. It starts with an initial lattice associated to K and goes on until the result evolves no more, i.e., a fixed point is reached. The procedure may be summarized as follows: at step (0), L^0 is constructed from K^s_0 which is scaled on all attributes but α; at step $(i+1)$, L^i is used as a set of scale attributes for α to produce K^s_{i+1} and further construct L^{i+1}. The procedure stops whenever L^{n+1} is isomorphic to L^n. Obviously, the lattice L^* reached this way is the smallest fixed point of the underlying lattice operator. The following condition is characteristic for L^*: $\forall\, c \in L^*, \alpha^{-1}(Extent(c))$ is a valid extent in L^*.

```
proc MULTI-FCA( In: $R^s = (\mathbf{K}_R, A_R)$ a RCF,
                Out: $\mathbf{L}$ array of lattices )
$i \leftarrow 0$ ; halt $\leftarrow$ false
for $j$ from 1 to $s$ do
  $K^d_j \leftarrow$ SCALE-BIN($K_j$) -- Binary scaling
  $\mathbf{L}^i[j] \leftarrow$ FCA($K^d_j$) -- Lattice construction
while not halt do
  $i++$
  for $j$ from 1 to $s$ do
    $K^d_j \leftarrow$ SCALE-REL($K^d_j$, $\mathbf{L}^{i-1}$) -- relational scaling
    $\mathbf{L}^i[j] \leftarrow$ FCA($K^d_j$) -- Lattice construction
  halt $\leftarrow \bigwedge_{j=1,s}(\mathbf{L}^i[j] = \mathbf{L}^{i-1}[j])$
```

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

The iterative construction principle may be applied to the entire set of lattices within a RCF. To that end, one may first detect the strongly connected components of the RCF induced by the relational dependencies. However, for simplicity reasons, we give below a very general form of our method, called MULTI-FCA, that integrates both the resolution of circularities and non-circular relational scalings into a single global loop that follows the evolution of the lattice vector associated to the RCF. Algorithm 1 provides a summary of our method.

To deal with loops, our method applies the previously presented iterative computation strategy: starting from a rough approximation of the final solution and refining it at each step. The result is the minimal fixed point of the complex lattice vector function. A single iteration consists in subsequent visits of all contexts and respective lattices whereby existing lattices are used as scaling hierarchies in order to refine the object descriptions in related contexts (primitive EXTEND-REL). These are further used to replace current lattices with more detailed structures (primitive FCA). The entire iterative process halts when a construction step does not lead to the discovery of previously unseen formal concepts.

When applied to the example RCF, R_H^1, the above method halts in three steps, as we saw in Section 3.1: the initial step yields the lattice L_H^0 which is visualized in Figure 1, whereas the process ends up with a structure isomorphic to the lattice in Figure 4. Further work is necessary to transform the latter into an intelligible concept hierarchy with simplified concept descriptions and effective inter-concept relations.

To illustrate the power of the relational FCA approach, we zoom in on a pair of mutually related concepts (see Figure 7). The mutual definition of both classes in the figure is to be interpreted as a strong correlation between persons aged 27 or less and those married to persons that have worked for 5 years or less. The inverse is also true, i.e., 5 years or less of work, is correlated to living with somebody aged 27 or less. Although the concrete example we have chosen is of little practical value, the possibility of finding such classes in larger sets might well be crucial. And this is where our framework, i.e., definition of RCF, problem definition and algorithmic scheme, outperforms any straightforward relational extensions of classical method.

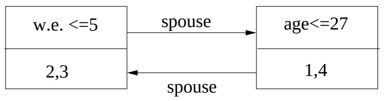


Fig. 7. Sample of two concepts with circular definitions.

3.4.4 Properties of MULTI-FCA

The MULTI-FCA method has been proved correct and its convergence has been formally established. The same holds for its reduced version called MULTI-GSH that yields a set of Galois sub-hierarchies instead of complete lattices. The main focus in current theoretical investigations is the definition of the target set of lattices in a declarative way, i.e., independently from the iterative algorithm used to their construction.

The efficiency is an unavoidable bottleneck of the method. Indeed, maintaining a set of lattices and performing constructions in a row may prove to be very expensive when data starts growing. Therefore, the current work on the subject tends to minimize the overhead due to lattice reconstruction. One improvement point is to use incremental algorithms that avoid reconstruction from scratch upon every addition of an attribute. Moreover, further efficiency gains may result from the replacement of lattice maintenance by a direct assembly of the current lattices and the scale lattices.

3.5 Tool support for mining of relational concepts

The processing of RCFs and the MULTI-FCA algorithm are currently under development within the GALICIA platform[4]. This platform is intended to support the entire process for FCA, from data acquisition to navigation and pruning of the resulting lattices and derived structure. Among other functions, GALICIA features several advanced lattice construction techniques, both batch and incremental, computation of various implication rule bases and 2D and 3D visualization of lattice diagrams. Complex data types are also provided in the extensible architecture of the platform, in particular XML-based descriptions of structured documents.

GALICIA is developed by an international consortium of research teams, led by the team at Montreal University. A more detailed description of the platform can be found in [31].

Notwithstanding the yet incomplete understanding of the behavior of MULTI-FCA method, the related construction techniques have already found interesting applications in software engineering as it is explained by the next section.

3.6 Open issues with relational concept analysis

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.

[4] See the website of the project at: `http://www.iro.umontreal.ca/~galicia`.

The result about the convergence of the MULTI-FCA method (see Section 3.4.4) 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 context evolves only by (possibly) extending its attribute set while the object set remains steady. Therefore, at iteration $i + 1$, instead of constructing the lattice L_{i+1} from scratch, one may use the available structure in 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).

4 Application to UML

We illustrate the benefits of the entire relational framework and the scaling of relations described in the previous sections through an application to class hierarchy reorganization from UML models. Hierarchy optimization through Galois/concept lattices and FCA is not new [15,9,37,16], but previous works have ignored relational information, like the information encoded in attribute/method description or in associations. Our proposal therefore offers richer possibilities for hierarchy reconstruction as interesting abstractions, impossible to reach by the known methods, are discovered. First, UML class diagrams are introduced (Section 4.1) and a motivating example is proposed (Section 4.2). Then, iterative abstraction of classes and associations is detailed and applied to the example (Section 4.3). We conclude the section discussing practical aspects (Section 4.4).

UML (Unified Modeling Language) [26,3] is a popular language in the software engineering 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 mainly made up of classes, specialization/generalization relationships 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 8, 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`. Specialization/generalization relationships are drawn between classes based on their instance (object) set inclusion, e.g., arrow going from `Chief_Officer` to `Manager` expresses that the chief-officer set is included into the manager set. As a consequence the features of the class `Manager` are inherited by class `Chief_Officer`, and in some cases refined.

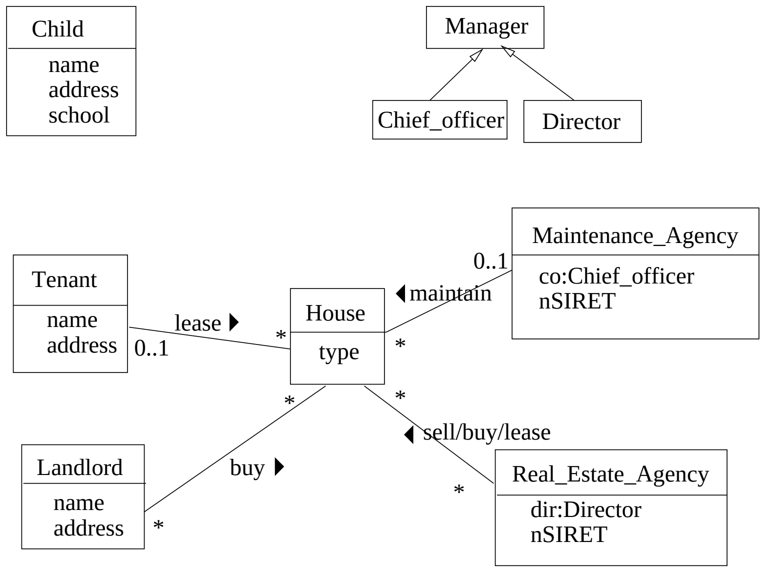


Fig. 8. An UML static diagram - House Transactions

Associations in UML model the relations among objects of the domain categories that are represented as classes. For example, in Figure 8, 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 4.3.1.

4.2 Motivating the reconstruction of the example

The example in Figure 8, which was adapted 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 go-between in various commercial transactions like `selling, buying` and `leasing` a house. 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 itself 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. Consequently, we claim that for an appropriate reconstruction of UML static diagrams, associations need to be dealt with in a direct manner, i.e., without a translation phase into the programming language model.

Our approach thus consists of processing two sorts of formal objects, classes and associations, which compile to two different formal contexts. The respective formal attributes not only describe their local properties, but also capture the relationships that hold among objects of opposed contexts. Such a separation, except for being natural for a FCA method, eases the interpretation task for the designer during the step-wise construction.

The result of the method we propose is presented in Figure 9. Classes $C'1$, $C'2$,

$C'3$, $C'4$, $C'11$ and associations $a'1$, $a'2$, $a'3$, $a'4$ have been discovered. Their meaning is detailed at the bottom of the figure, as well as correspondence with initial classes and associations. Most of the new "concepts" are of considerable interest, for instance, the concept of general house transaction, the concept of persons (resp. of the organizations) involved in a house transaction, etc. At a later step of the design, such concepts could support new methods or variables, like an insurance policy or regulations on specific transactions. As with other inductive methods, less relevant concepts (like `C’4` and `a’2`) can appear. Final reshaping is left to the designer who decides on whether to remove such irrelevant concepts or keep them. At this step, analysis tools can be provided, like those that have been proposed in the context of the `Macao` project (see section 4.4) and that help the designer to identify modifications.

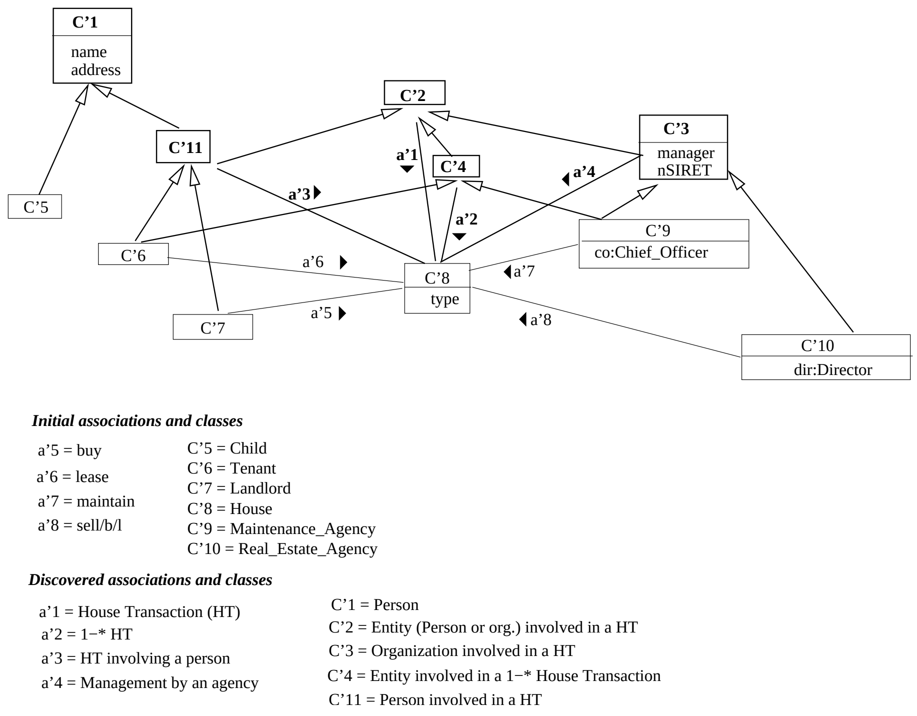

C’1
name
address
C’2
C’3
manager
nSIRET
C’11
C’4
a’1
a’2
a’3
a’4
C’5
C’6
C’7
C’8
type
C’9
co:Chief_Officer
C’10
dir:Director
a’5
a’6
a’7
a’8
Initial associations and classes
a’5 = buy
a’6 = lease
a’7 = maintain
a’8 = sell/b/l
C’5 = Child
C’6 = Tenant
C’7 = Landlord
C’8 = House
C’9 = Maintenance_Agency
C’10 = Real_Estate_Agency
Discovered associations and classes
a’1 = House Transaction (HT)
a’2 = 1−* HT
a’3 = HT involving a person
a’4 = Management by an agency
C’1 = Person
C’2 = Entity (Person or org.) involved in a HT
C’3 = Organization involved in a HT
C’4 = Entity involved in a 1−* House Transaction
C’11 = Person involved in a HT

Fig. 9. The UML diagram revisited - House Transactions

The process which realizes the GSH-based reconstruction of the above example by systematically applying the iterative method from Section 3.1 is described with greater details in Section 4.3.

4.3 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 4.3.1). Then the iterative process is described (Section 4.3.2) and applied to the running example (Section 4.3.3).

4.3.1 Many-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 [28].

A key step in our proposal is determining the right formal binary or many-valued attributes 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 [26]. In Figure 10, 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 end types of the association `place_order`;
- a visibility (or access control). When nothing is mentioned, the end is public;
- a multiplicity (number, interval, symbol '*').

Stronger links between objects, such as an aggregation or a composition, also called *part-of* (a `Document` is a composition of `paragraphs`) can be graphically distinguished, namely through diamond-shaped association ends. Further expressive means indicate that the links of an object are ordered, e.g., the paragraphs that form a document. An association end may have 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 10), an association end is sometimes provided with a role name which gives more accurate semantics to objects when they are involved in the link (*e.g.* role *employee* for a person in association *manage*). When the association is described with its own variables and methods, a specific class, called the *association class*, is assigned to it (`Access` is an association class that supports the variable `passwd`). The detailed description of the above constructs from the textual and visual languages in UML is far beyond the scope of this brief introduction. Most of them are depicted in Figure 10, while the respective syntactic and semantic details may be found in the extensive literature on UML and OO modeling.

Our method uses the relational context family (K_R, A_R) where $K_R = \{K_1 = (O_1, A_1, V_1, J_1), K_2 = (O_2, A_2, V_2, J_2)\}$. K_1 is the many-valued context for classes (O_1 is the class set) and K_2 is the many-valued context for associa-

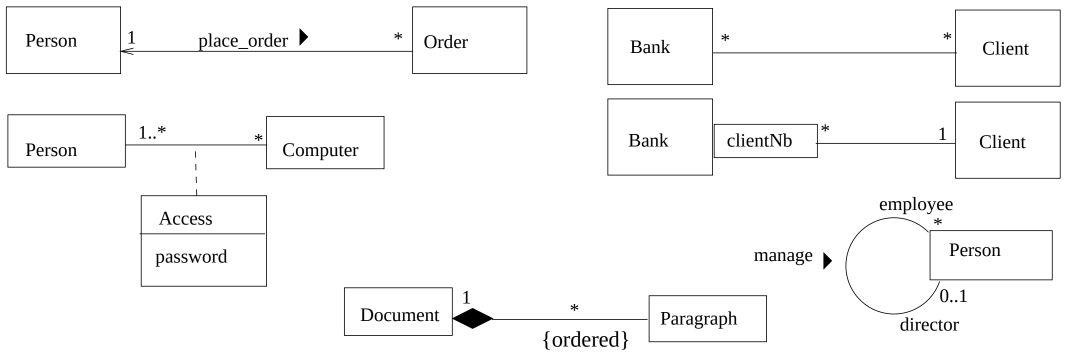


Fig. 10. Examples of UML associations

tions (O_2 is the association set). A_R will represent relational attributes linking classes and associations.

Local class attributes (A_1, V_1 and J_1) The features describing a class within the model considered here correspond to variables and methods while association ends constitute the relational attributes. Thus, given a class C, the formal attributes A_1 represent class members and the values V_1 are typically boolean. Furthermore, J_1 represents the "owns-property" relation: $(o\ a\ v) \in J_1$ if the class o owns the value v of formal attribute a. Thus, in the House Transaction example formal attributes are mostly binary except the variables `co:Chief_Officer` and `dir:Director`. In this case, a small amount of domain knowledge is introduced by the designer thus leading to the constitution of the many-valued formal attribute `resp` whose scale is given in Figure 11.

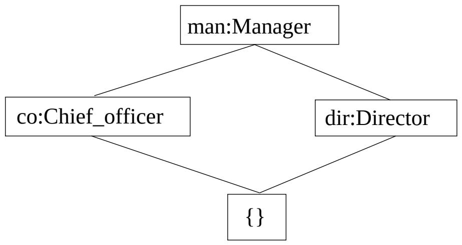


Fig. 11. Scale for formal many-valued attribute `resp`.

Further discussion on heuristic rules used for mining many-valued formal attributes from UML diagrams are provided in [7].

Relational class attributes ($rel(K_1)$) Information on the relationships between classes and UML associations is represented by three relational attributes in A_R:

- $origOf : O_1 \longrightarrow 2^{O_2}$, $a_i \in origOf(C)$ if all objects of C can be origins of the association a_i,

- $destOf : O_1 \longrightarrow 2^{O_2}$, $a_i \in destOf(C)$ if all objects of C can be destinations 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.

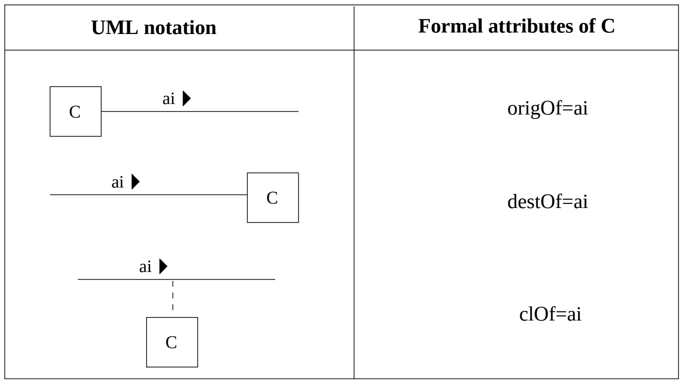


Fig. 12. Overview of formal attributes for classes.

Figure 12 gives a visual overview of these rules. In Figure 13 are presented local and relational attributes that describe the classes of the Figure 8 (names have been shorten).

		name	addr	sc	resp	nS	ty	origOf	destOf
Ch	Child	x	x	x					
Te	Tenant	x	x					lease	
La	Landlord	x	x					buy	
Ma	Maint_Ag				co:Chief_Off	x		maintain	
Re	RE_Ag				dir:Director	x		sell/buy/lease	
H	House						x		{lease,buy,maintain,sell/buy/lease}
		local attributes						relational attributes	

Fig. 13. Many-valued context K_1^1 composed of K_1 and $rel(K1)$

Local association attributes (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*).

Set inclusion is used for scaling multiplicities.

Relational association attributes ($rel(K_2)$) Information on the relationships between classes and UML associations is represented by three relational attributes in A_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.

Scales on relational attributes to, td and ca are deduced from UML Specialization/generalization relationships, which involves a partial order on classes[5].

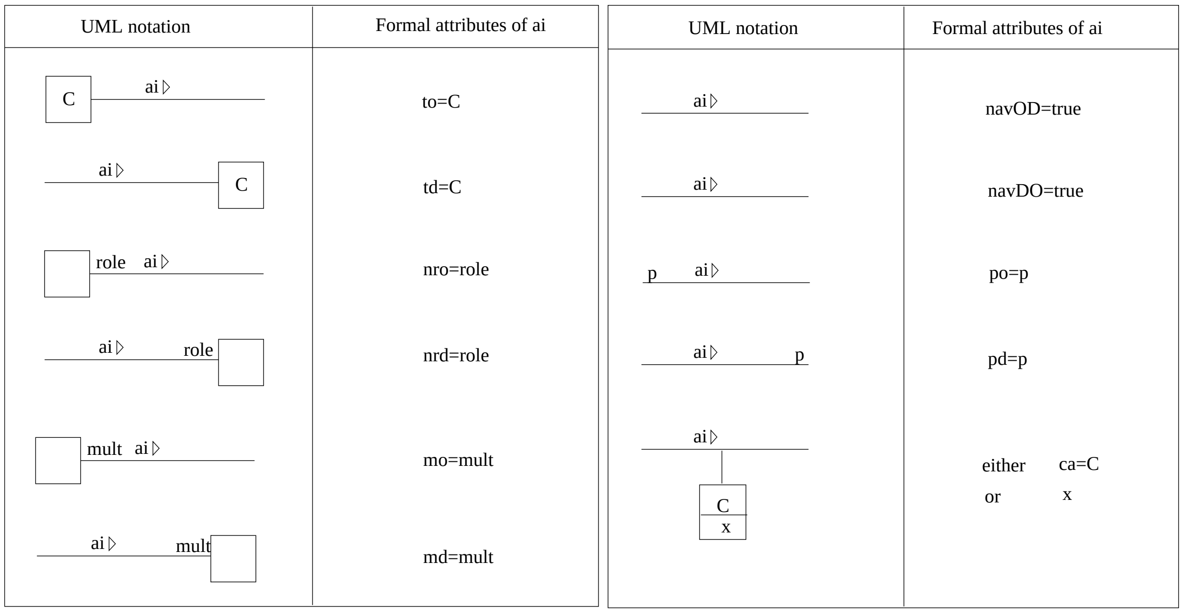


Fig. 14. Overview of formal attributes for associations.

An overview of formal attributes for associations appears in Figure 14 while description of associations of the house transaction example is given in Figure 15 (names have been shorten). The scaled context is presented in 16.

		mo	md	to	td
l	lease	0..1	*	Tenant	House
b	buy	*	*	Landlord	House
m	maintain	0..1	*	Maintenance_Agency	House
s	sell/buy/lease	*	*	Real_Estate_Agency	House
		local		relational attributes	

Fig. 15. Many-valued context K_2^1 composed of K_2 and $rel(K2)$

[5] In the running example, this scaling does not apply as no specialization/generalization relationship appears in the initial diagram.

4.3.2 The Iterative Process

This section introduces an instanciation of the MULTI-FCA algorithm for a relational context family (K_R, A_R) coming from an UML class diagram. Besides Galois sub-hierarchies rather than complete lattices are constructed. This limits the number of concepts produced while preserving those needed either for introducing variables, methods and association ends or for representing initial classes and associations.

		mo	mo	md	to	to	to	to	td
		0..1	*	*	Te	La	Ma	Re	H
l	lease	x	x	x	x				x
b	buy		x	x		x			x
m	maintain	x	x	x			x		x
s	sell/buy/lease		x	x				x	x
		local			relational attributes				

Fig. 16. The scaled many-valued context $K_2^{1\ s}$

This gives the generalization method ICG (for "Iterative Cross Generalization") which consists in iterating a pair of generalization methods:

- computation of the Galois sub-hierarchy associated with a many-valued context for associations (at step i, denoted by K_2^i)
- computation of the Galois sub-hierarchy associated with a many-valued context for classes (at step i, denoted by K_1^{i+1})

The input relations are obtained by associating to every formal object (class or association) its formal local and relational attributes. 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. A step of the process is developed below.

Step i. ($ICG(i)$)

a Computation of $GSH(K_2^i)$
b Computation of K_1^{i+1} using $GSH(K_2^i)$ for scaling relational attributes
c Computation of $GSH(K_1^{i+1})$
d Computation of K_2^{i+1} using $GSH(K_1^{i+1})$ for scaling relational attributes

4.3.3 The House transaction example (continued)

The iterative process is applied to the house transaction example.

Step $ICG(1)$.a Construction of $GSH(K_2^1)$, shown in the top of Figure 17 together with an UML interpretation (bottom), introduces two new associations:

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

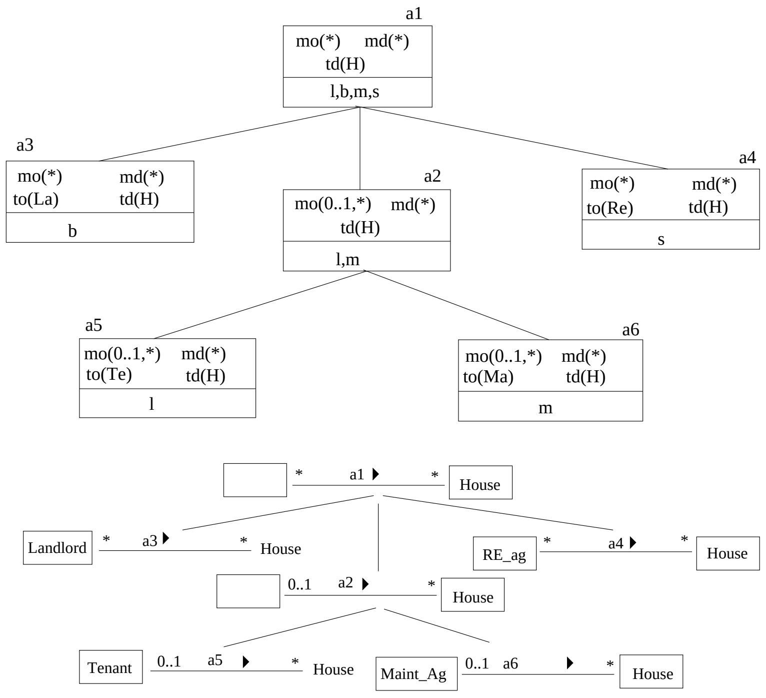


Fig. 17. The Galois sub-hierarchy $GSH(K_2^{1\ s})$ (top), an UML interpretation of the specialization order on associations (bottom)

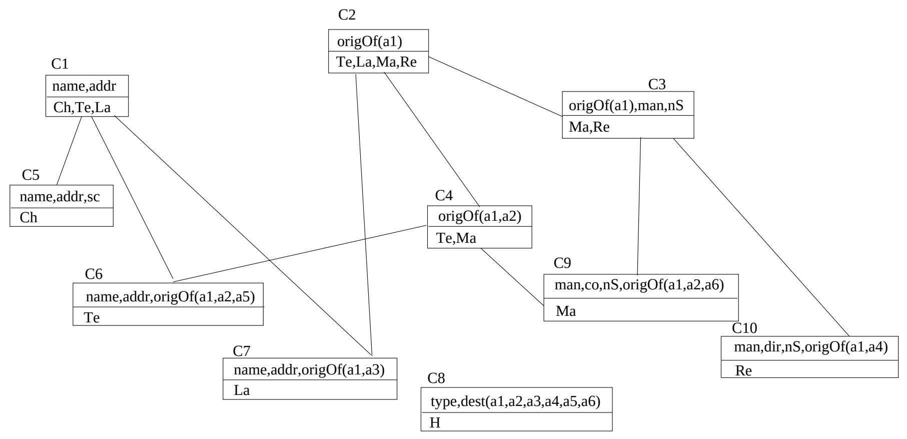
Step $ICG(1)$**.b** $GSH(K_2^1)$ is used as a scale to adjust the context K_1^1. Figure 18 (top) presents the resulting scaled context $K_1^{2\ s}$ and the associated constructions.

Step $ICG(1)$**.c** $GSH(K_1^{2\ s})$ is shown in Figure 18 (center), as well as an UML class diagram. The concept which currently is origin of an association a is the higher concept which owns $origOf = a$.

Step $ICG(1)$**.d** K_2^1 is now adjusted by a scaling on to et td. The resulting relation $K_2^{2\ s}$ appears in Figure 19 (top).

Step $ICG(2)$**.a** $GSH(K_2^{2\ s})$ is presented in Figure 19 (bottom). Concepts $C1$ and $C3$ have induced two properties $to = C1$ and $to = C3$ which determine

				resp					origOf						destOf					
	name	addr	sc	co	dir	man	nS	ty	a5	a3	a6	a4	a1	a2	a5	a3	a6	a4	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



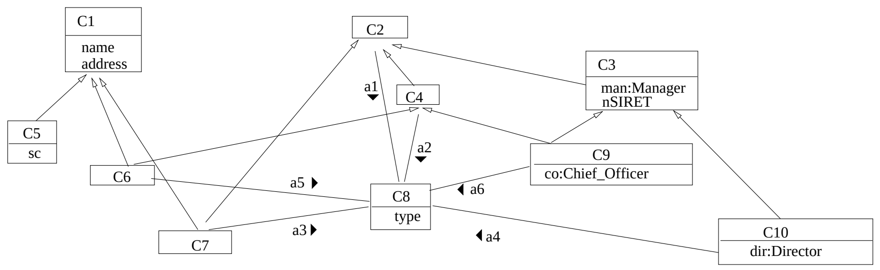


Fig. 18. Scaled many-valued context $K_1^{2\ s}$ (top), $GSH(K_1^{2\ s})$ (center), UML class diagram (bottom)

two new associations, $a'3$ which generalizes $(a'5, a'6)$ and $a'4$ which generalizes $(a'7, a'8)$.

Step $ICG(2)$**.b** These two new associations bring new information on classes, as shown in Figure 20. $K_1^{3\ s}$ is obtained from K_1^1 (without integrating concepts of $GSH(K_1^{2\ s})$) as some of them could be intermediate artifacts): K_1^1 is filled using a scaling of $origOf$ and $destOf$ based on $GSH(K_2^{2\ s})$.

Step $ICG(2)$**.c** In Figure 9 is shown the UML diagram coming from the interpretation of $GSH(K_1^{3\ s})$. When $GSH(K_1^{3\ s})$ is constructed, a new concept appears ($C'11$) which factorizes $origOf = a'3$, while $C'3$ is $C3$ completed for factorizing not only $nSIRET$ and $man : Manager$ as in the previous construction, but also $origOf = a'4$.

	mo		md	to								td
	0..1	*	*	C6	C7	C9	C10	C1	C2	C3	C4	C8
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

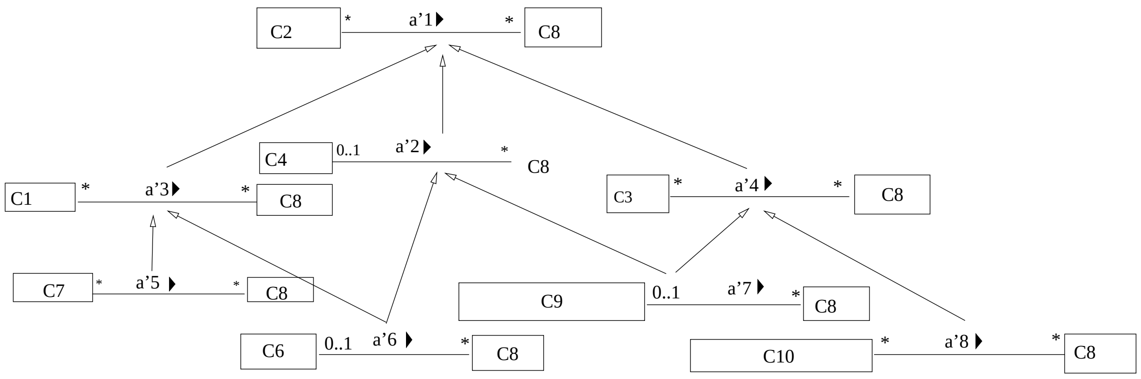


Fig. 19. Scaled many-valued context $K_2^{2\ s}$ (top), UML interpretation (bottom)

				resp					origOf								destOf							
	name	addr	sc	co	dir	man	nS	ty	a'6	a'5	a'7	a'8	a'1	a'2	a'3	a'4	a'6	a'5	a'7	a'8	a'1	a'2	a'3	a'4
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

Fig. 20. Scaled many-valued context $K_1^{3\ s}$

Step $ICG(2)$**.d** According to the same previous rules, the new concept $C'5$ modifies the description of associations giving $K_2^{3\ s}$ (Figure 21). $GSH(K_2^{3\ s})$ does not really contain new association concepts (there is no new association closed set) but $a'3$ intent is completed with $to = C'11$ to give $a3$. The process stabilizes since the class context will not be changed by scaling on $GSH(K_2^{3\ s})$.

4.4 Practical aspects

4.4.1 Implementation issues

The method ICG has been implemented in the framework of the Macao project[6], a joint project between Softeam, developper of the case tool Objecteering, France Telecom R&D and LIRMM. The project aims at proposing audit and construction assistance for UML models. It has been supported by the French Research Ministry within the rntl program. The

[6] http://www.lirmm.fr/˜macao

	mo		md	to									td
	0..1	*	*	C'6	C'7	C'9	C'10	C'1	C'2	C'3	C'4	C'11	C'8
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

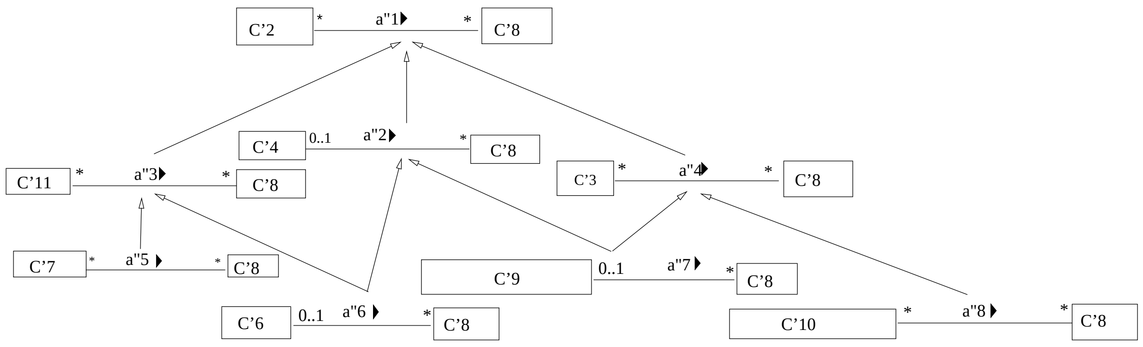


Fig. 21. $K_2^{3\ s}$ and UML interpretation of $GSH(K_2^{3\ s})$

implementation of *ICG* and its use by the Macao team is described in greater details in the project report [7].

Figure 22 illustrates the current implementation strategy. The case tool Objecteering is used as a UML class diagram editor as well as to give an interface where the designer can call and fine-tune *ICG*, choosing formal local and relational attributes. For example multiplicity and navigability can be omitted in local association attributes to avoid too many generalizations to emerge. When the designer starts diagram reconstruction, data are exported for use in Galicia (see Section 3.5). This latter tool is a general platform for formal concept analysis which, among others, proposes an implementation of *ICG* whereas the developments of the general Multi-Fca method is well advanced. At the end of the *ICG* algorithm, results are imported in Objecteering and interpreted for giving to the designer a usual UML class diagram. Initial class and association names are especially preserved while new artifacts are automatically named.

Objecteering offers a programming language (J) with which we could have implemented *ICG* but after a first implementation of ICG in J, efficiency reasons have dictate our choice of exporting data and processing them in Java. The use of Galicia is entirely hidden to the UML designer.

ICG has been applied to several UML frameworks of France Telecom R&D under the direction of M. Dao [6]. Results are really encouraging. The reconstructed UML diagrams help designers to discover better generalizations that greatly simplify their design (after removal of useless artifacts). Another concrete result is that, when results are uninterpretable or inadequate, often designers reconsider their initial design with a fresh eye and find inconsistent or

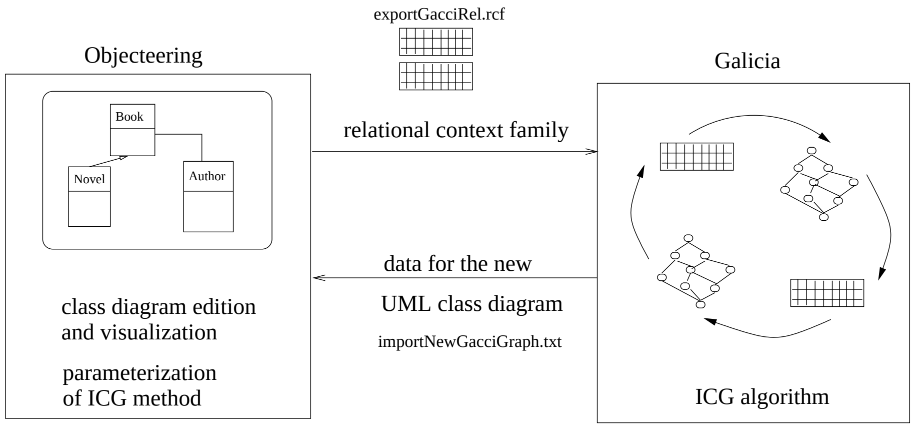


Fig. 22. Tools

inappropriate naming, description or organization of classes and associations. In this last case, *ICG* has to be applied after modifications of the initial diagram have been done to fix the problems.

4.4.2 Open questions

Scenarios of use From the designer point of view, defining good scenarios for using such reconstruction methods is crucial. We believe that step-by-step methods like *ICG*, where generalizations emerge little by little, can help the designer to understand and interact with the process. A natural extension of our work would be identifying actions a designer could do, as accepting or not some of the generalizations, stopping the process before it converges, applying some simplifications on the diagram, etc.

Semantic issues Capturing the semantics of modeling through the rather syntactical aspect of UML diagrams is an actual challenge. Problems raised by semantics include:

- *naming*: names used in diagrams are a source of many problems of conflict and synonymy that we think can be solved only by a designer, perhaps guided by an ontology of the application domain;
- *end ordering*: the association name and its attached black triangle explicitly order the classes that are at the two ends of the association [26]. We have preserved this asymmetry by identifying an origin and a destination in the association description. In the case of very symmetrical associations, this ordering could not be suitable. Using UML 1.4 metaclass `EndAssociation` (or UML 2.0 metaclass `Property`) as an additional context is a way of solving this problem but that complicates the process;
- *association class status*: as said in [26], "the association class and the association are the same semantic entity". This could indicate that variables

and methods of the association class could be used as formal attributes of the association, and the association class would not appear as a class. This question is not yet closed;

- *level of importance of formal attributes*: as reported in [27] with a reference going back to Aristotle's work, some descriptions are essential (tell what the subject is) while other are accidental (tell something else about the subject). Distinguishing between these kinds of descriptions would improve the generalization process, but UML notation does not actually give keys to do that. Generalizations based only on same arity multiplicity for example are somewhat questionable and could be avoided or considered with less importance than generalizations based on aspects like same name variables.

Towards a unified approach The iterative construction schema could be extended considering variables and methods as formal objects. Variables as well as methods can be described by their name, signature (type for variable, parameter list plus return type for methods), multiplicity for attributes, thrown exceptions and code for methods, etc. From each of the four many-valued context is computed the associated Galois sub-hierarchy. In practice this is justified because generalizing one of the four kinds of formal objects (class, association, variable, method) can have consequences on the others, for example new generalizations on variables or methods can determine new generalizations on classes.

5 Conclusion

We presented an approach which allows the classical FCA to deal with a relational context family, i.e., a set of formal contexts whose objects are related by links. Relational datasets characterize a large number of practical situations ranging from relational databases to software artifacts such as models or source code. Therefore it is of prime importance to develop tools for the analysis of such data.

Our own approach is the first one to enable the discovery of formal concepts characterized by both the shared features of member objects and by their relations to other formal concepts. Its core mechanism is a high-level algorithmic method that constructs several lattices on top of a relational context family. In case of circular dependencies between contexts of the family, the construction proceeds by subsequent lattice construction and relational extensions of contexts. The final structures represent the fixed point of a complex and yet implicit lattice (vector) function. Our method is therefore capable of discovering inter-concept relations despite the loops that may exist in object structure.

The construction of the appropriate lattices for an entire RCF is a challenging problem that conveys aspects tightly related to key trends in modern data analysis and, even more so, machine learning. Indeed, relational learning and relational data mining, two disciplines that study the analysis of relational data, are very dynamic nowadays (see [11]). Besides being affiliated to these state-of-the-art trends, the resolution of the above problem presents features of constructive learning as the concepts discovered on some contexts will be used to discover other concepts on related contexts. In more specific terms, the language used to describe the discovered abstractions is refined during the discovery process.

Although the approach has put the emphasis on the theoretical framework, effective algorithmic and software tools implementing the theoretical developments have been designed and tested. The practical impact of the framework and its respective set of tools is clearly illustrated by the current achievements in the industrial application on analysis of UML structural models. As a matter of fact, the *ICG* method for constructing GSHs on top of a RCF drawn from a UML class diagram, is a key part of the MACAO project involving France Télécom and Softeam. Thus, it was implemented in the OBJECTEERING^{TM} CASE Tool and has shown encouraging effectiveness in its working environment.

In the near future, we will thoroughly investigate the behavior of the current methods and carry out a complete set of performance tests on them. Further algorithmic design and tool development will be necessary to demonstrate the entire capacity of the relational paradigm in FCA. Another research track involves experiments on various scenarios of human interactions with the CASE tool where designers could guide the reconstruction algorithm.

Acknowledgments

The authors want to thank their colleagues for thoughtful dicussions and critical remarks, in particular Michel Dao, Thérèse Libourel and Robert Godin. The former two get credits for the experimental results as well.

References

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

[2] G. Booch, J. Rumbaugh, and I. Jacobson. *Unified Modeling Language User Guide.* Addison-Wesley, 1999.

[3] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Object Technology. Addison-Wesley, 2000.

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

[5] P. Chen. The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[6] M. Dao. Validation sur de grands projets, Projet MACAO (RNTL). Technical Report sous-projet MACAO 5.1, France Télécom R&D, december 2003.

[7] M. Dao, M. Huchard, T. Libourel, and C. Roume. Spécification de la prise en compte plus détaillée des eléments du modèle objet UML. Technical report, Projet MACAO. Réseau RNTL, 2001.

[8] B. A. Davey and H. A. Priestley. *Introduction to lattices and order*. Cambridge University Press, 1992.

[9] H. Dicky, C. Dony, M. Huchard, and T. Libourel. On Automatic Class Insertion with Overloading. In *Special issue of Sigplan Notice - Proceedings of ACM OOPSLA'96*, pages 251–267, 1996.

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

[11] S. Džeroski and N. Lavrač. *Relational Data Mining*. Springer, 2001.

[12] M. Faid, R. Missaoui, and R. Godin. Knowledge discovery in complex objects. 15(1):28–49, 1999.

[13] 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, pages 139–167. Springer-Verlag, New York, 1989.

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

[15] R. Godin and H. Mili. Building and maintaining analysis-level class hierarchies using Galois lattices. In *Proceedings of OOPSLA'93, Washington (DC), USA*, pages 394–410, 1993.

[16] 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.

[17] 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.

[18] G.Polaillon and E.Diday. Symbolic galois lattices of multivariate and interval tables. In *Proceedings of the International Conference on Ordinal and Symbolic Data Analysis (OSDA 97)*, Darmstadt, 1997.

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

[21] S. Kuznetsov. Learning of simple conceptual graphs from positive and negative examples. In J. Zytkow and J. Rauch, editors, *Proceedings of the Third European Conference PKDD'99, Prague, Czech Republic*, volume 1704, pages 384–391, 1999.

[22] S. Kuznetsov and S. Ob'edkov. Algorithms for the Construction of the Set of All Concept and Their Line Diagram. preprint MATH-AL-05-2000, Technische Universität, Dresden, June 2000.

[23] M. Liquiere and J. Sallantin. Structural Machine Learning with Galois Lattice and Graphs. In *Proceedings of the 15th International Conference on Machine Learning*, pages 305–313, 1998.

[24] G. W. Mineau, G. Stumme, and R. Wille. Conceptual structures represented by conceptual graphs and formal concept analysis. In W. M. Tepfenhart and W. Cyre, editors, *Conceptual structures: Standards and Practices*, volume 1640, pages 423–441, 1999.

[25] 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.

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

[27] D. Rayside and G. T. Campbell. An aristotelian understanding of object-oriented programming. In Doug Lea, editor, *Proceedings of OOPSLA'00, Minneapolis (Minesota), USA*, pages 337–353, october 2000.

[28] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object Oriented Modeling and Design.* Prentice Hall, 1991.

[29] G. Snelting and F. Tip. Reengineering Class Hierarchies Using Concept Analysis. In *Proceedings of ACM SIGPLAN/SIGSOFT Symposium on Foundations of Software Engineering*, pages 99–110, Orlando, FL, 1998.

[30] P. Valtchev. Building Classes in Object-Based Languages by Automatic Clustering. In D. Hand, J. Kok, and M. Berthold, editors, *Proceedings of the 3rd International Symposium on Intelligent Data Analysis.*, volume 1642, pages 303–314, 1999.

[31] P. Valtchev, D. Grosser, C. Roume, and M. Rouane Hacene. GALICIA: an open platform for lattices. In A. de Moor B. Ganter, editor, *Using Conceptual Structures: Contributions to 11th Intl. Conference on Conceptual Structures (ICCS'03)*, pages 241–254, Aachen (DE), 2003. Shaker Verlag.

[32] P. Valtchev, M. Rouane Hacene, M. Huchard, and C. Roume. Extracting Formal Concepts out of Relational Data. In E. SanJuan, A. Berry, A. Sigayret, and A. Napoli, editors, *Proceedings of the 4th Intl. Conference Journées de l'Informatique Messine (JIM'03): Knowledge Discovery and Discrete Mathematics, Metz (FR), 3-6 September*, pages 37–49. INRIA, 2003.

[33] P. Valtchev, R. Missaoui, and P. Lebrun. A partition-based approach towards building Galois (concept) lattices. *Discrete Mathematics*, 256(3):801–829, 2002.

[34] F. Vogt and R. Wille. TOSCANA – a graphical tool for analyzing and exploring data. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing*, volume 894, pages 226–233, 1994.

[35] R. Wille. Restructuring lattice theory: An approach based on hierarchies of concepts. In I. Rival, editor, *Ordered sets*, pages 445–470, Dordrecht-Boston, 1982. Reidel.

[36] R. Wille. Conceptual structures of multicontexts. In P. W. Eklund, G. Ellis, and G. Mann, editors, *Proceedings of the 4th ICCS 1996, Sidney (AU)*, volume 1115, pages 23–39, August 1996.

[37] 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, pages 422–437, 1996.