

# Similarity-based Clustering versus Galois lattice building: Strengths and Weaknesses

Petko Valtchev, Rokia Missaoui  
Département d'Informatique, UQAM, C.P. 8888, succ. "Centre Ville",  
Montréal, Québec, Canada, H3C 3P8  
e-mail : valtchev@info.uqam.ca, missaoui.rokia@uqam.ca

## Abstract

In many real-world applications, designers tend towards building classes of objects such as concepts, chunks and clusters according to some similarity criteria. In this paper, we first compare two approaches to clustering: the Galois lattice approach [14] and a similarity-based clustering approach [27]. Then, we sketch the possible ways each approach can benefit from the other in refining the process of building a hierarchy of classes out of a set of instances.

## 1 Introduction

Building class hierarchies is a very sought and common task in many computer science areas such as software engineering (including object-oriented models and systems), data management and knowledge representation. It has been mainly used for data and process modeling, system design, and performance optimization purposes. For instance, it has been successfully applied for software reuse (e.g., code reuse, class reorganization), reverse-engineering, model refactoring in databases and knowledge bases, and physical clustering of objects, to name a few.

In this short position paper, we compare the strengths and limitations of two distinct clustering techniques: the Galois lattice (GL) [28, 16, 14] and the similarity-based clustering (SBC) as described in [19, 6, 27]. While both of them allow the construction of a hierarchy of classes with both an extension and an intension, each one presents its own and peculiar features that make it better fit to particular applications.

The paper is organized as follows. Section 2 gives a background on the two conceptual clustering techniques. In Section 3, we put in contrast GL with SBC. Section 4 describes some possible combinations of these techniques. Finally, a conclusion is provided in Section 5.

## 2 Background

In the following we present the two clustering techniques which both take as an input a set of individuals (objects)  $O$  and a set of attributes  $A$ . Both of them divide the set  $O$  into sub-sets (classes) of highly similar objects. However, each clustering technique considers different sorts of attributes, uses its own similarity criterion and organizes the classes in a distinct hierarchical structure in the form of a tree or a lattice.

## 2.1 An illustrating example

In the following, we consider a set of six objects representing people in a hypothetical database. The objects are described by two attributes (see Table 1) : **age**, of integer type in the range of  $[20, 35]$ , and **salary**, a float in  $[1.8, 3.3]$  indicating person’s monthly income in thousands of euros.

Table 1: A set of objects representing human beings

attribute	$o_1$	$o_2$	$o_3$	$o_4$	$o_5$	$o_6$
age	26	23	27	30	32	35
salary	3	2.3	2.2	2.7	2.9	3.2

## 2.2 Galois (concept) lattices

The paradigm first occurred within the lattice theory [4]. Originally, only binary attributes are considered, i.e. with only two possible values: *present* or *absent*. Thus, objects in  $O$  can be seen as binary vectors and the dataset as a binary table with an incidence relation  $I$  ( $oIa$  means that the object  $o$  has the attribute  $a$ ).

### 2.2.1 Basic method

The table  $(O, A, I)$ , further called *formal context* or simply context, may be processed to extract the inherent lattice structure defining natural groupings and relationships among the objects and their attributes. This structure is known as a *formal concept lattice* [14]. Each *concept* of the lattice  $L$  derived from the context  $\mathcal{K} = (O, A, I)$  is a couple  $(X, Y)$  composed of an object set  $X \in \mathcal{P}(O)$  and an attribute set  $Y \in \mathcal{P}(A)$ . Concepts are complete couples with respect to  $I$ , which means that the following two conditions are satisfied:

- $Y = f(X) = \{a \in A \mid \forall o \in X, oIa\}$
- $X = g(Y) = \{o \in O \mid \forall a \in Y, oIa\}$

The functions  $f$  and  $g$  constitute a Galois connection [4] between  $\mathcal{P}(O)$  and  $\mathcal{P}(A)$ . The concept lattice  $L$  on  $\mathcal{K} = (O, A, I)$  is made up of all complete couples with the partial order  $\leq$ , where:

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

*Join* and *meet* in the concept lattice are given by:

- $(X_1, Y_1) \wedge (X_2, Y_2) = (X_1 \cap X_2, f \circ g(Y_1 \cup Y_2))$
- $(X_1, Y_1) \vee (X_2, Y_2) = (g \circ f(X_1 \cup X_2), Y_1 \cap Y_2)$

In the original settings of [14], the Galois lattice (GL) approach handles numerical data, as those of the previous section, through a preliminary encoding called *scaling*. Scaling results in a binary context, called *derived*, where each numerical attribute is replaced by a set of binary ones. Each derived attribute represents a predicate corresponding to a suitable part of the original attribute range. For instance, the context on the left side of Figure 1 has been extracted from the data table of Section 2.1 by splitting the attribute domains and assigning binary attributes to them. Thus, **age** is split into two attributes,  $p$  (**age** in  $[20, 27]$ ) and  $q$  (**age** in  $[28, 35]$ ), whereas **salary** has been sliced into three distinct parts:  $[1.8, 2.3]$  for  $r$ ,  $[2.4, 2.9]$  for  $s$  and  $[3.0, 3.3]$  for  $t$ .

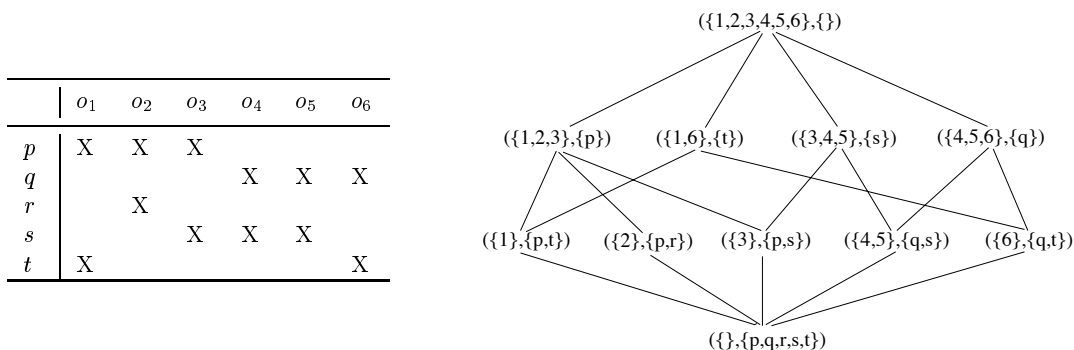


Figure 1: A binary relation expressed as a formal context and its Galois lattice

### 2.2.2 Further developments

With the work of Diday et al. [11] on the symbolic data analysis, a significant enlargement of the GL scope is reached by the following fact: a Galois connection may be directly established on an arbitrary set of attributes  $A$  (not necessary binary) provided that the value domain of each  $a \in A$  could be structured as a lattice of sub-domains [7]. In particular, this allows numeric data to be dealt with in a direct way: instead of fixing intervals in a scaling step, they are determined dynamically, i.e. during the lattice formation. However, the resulting lattice may grow exponentially to the size of the dataset. For example, the set of all complete couples over the set  $\{o_1, \dots, o_4\}$  of our dataset corresponds to its power-set  $\mathcal{P}\{o_1, \dots, o_4\}$ <sup>1</sup>.

Further developments concerned the relaxation of the crisp concept definitions in the original settings: fuzzy [15], probabilistic [11] and rough [20] formal concepts have been investigated. Yet different research stream looked to overcome the inherent limits of pure attribute value description formalisms: GL methods have been defined on structured terms [9], on function-free predicate-logic languages (Datalog) [8] and on object-oriented formalisms [25]. Efficiency requirements motivated research on suitable scalings of structured formalisms like conceptual graphs [23] and composite objects [12].

## 2.3 Similarity-based clustering

The similarity paradigm first occurred within the statistical data analysis [1]. In this framework, attributes in  $A$  take exclusively real values, i.e. they represent maps from  $O$  to  $\mathbb{R}$ . Thus, objects in  $O$  can be seen as real vectors, or equivalently, as points in  $\mathbb{R}^n$  where  $n$  is the size of  $A$ . In the sequel, the classical object notation  $o.a$  stand for the value of  $a$  for  $o$ .

### 2.3.1 Basic method

A proximity measure  $d$  is defined over a couple of sets  $(O, A)$  in order to quantify object likeness. It maps object pairs into non-negative real numbers,  $d : O \times O \rightarrow \mathbb{R}_+$ . We only consider distance-like proximity measures, further called *dissimilarities*, the dual *similarity* notion being ignored. They share a common metaphor, the Euclidean distance in  $\mathbb{R}^n$ . In a similar way, the value of such a measure is made up of elementary distances on each "dimension", i.e. on each object attribute. Among the most

<sup>1</sup>Fortunately, this is not always the case (e.g., the entire set  $\{o_1, \dots, o_6\}$ ).

popular dissimilarity functions, consider the Minkowski family:

$$d_p(o_1, o_2) = \left( \sum_{i=1}^n |o_1.a_i - o_2.a_i|^p \right)^{1/p}$$

Different values of  $p$  in the above formula lead to distinct functions: with  $p$  of 1, 2 and  $\infty$  the result is the so-called *Manhattan*, the Euclidean and the *Chebichev* distance respectively. Furthermore, the contribution of different attributes to the global value may be fine-tuned by attribute weights:

$$(1) \quad d_p(o_1, o_2) = \left( \sum_{i=1}^n \lambda_i |o_1.a_i - o_2.a_i|^p \right)^{1/p}$$

Usually, the sum of the weights is set to one ( $\sum_{i=1}^n \lambda_i = 1$ ) so that a global normalization remains possible. Using normalized functions, i.e. ones taking their values between zero and one, insures sensible comparison of objects having quite diverging characteristics. The normalization is achieved in two steps: (i) subtract the attribute mean from each attribute value and (ii) divide the result by the standard deviation of the attribute. Alternatively, one may simply divide each attribute contribution in Formula 1 by the attribute’s maximal range.

Specific constraints lay on the way the measures behave, which provide for sensible clusterings. In the following,  $o_1, o_2$  and  $o_3$  are objects in  $O$ :

1)	$d(o_1, o_1) = 0$	(minimalness)
2)	$d(o_1, o_2) = d(o_2, o_1)$	(symmetry)
3)	$d(o_1, o_2) = 0$ implies $o_1 = o_2$	(definiteness)
4)	$d(o_1, o_2) \leq d(o_2, o_3) + d(o_1, o_3)$	(triangular inequality)

Functions that satisfy properties 1) and 2) are called *dissimilarity indexes*, those satisfying 1) to 3) *semi-metrics* and 1) to 4) *metrics*. The intuition behind a metric is that two objects that are similar to a third one should not be too dissimilar between them.

To exemplify the computation of a dissimilarity, we take a Manhattan distance over the object set of Table 1 with equal weights for both attributes and the original ranges. With these settings, the dissimilarity  $d_p$  of a couple of objects, say  $(o_1, o_4)$ , becomes:

$$d_p(o_1, o_4) = 0.5 |26 - 30| / (35 - 20) + 0.5 |3 - 2.7| / (3.3 - 1.8) = 0.28$$

Similar computations help transform the entire object-attribute table into a square matrix of object-to-object dissimilarities (see the left part of Figure 2). The matrix is fed into a clustering procedure whose output is a set of classes, possibly organized into a hierarchy. For instance, a common family of clustering methods, the SAHN<sup>2</sup> algorithms, produce a *dendrogram*, a binary tree whose nodes are classes of objects indexed by dissimilarity values. The generic SAHN process starts with the set of singleton classes and proceeds by successive merges until a unique class is reached. At each step, the most similar classes are merged, whereby each concrete algorithm applies its own class similarity criterion. For example, the single-linkage method whose result is shown in the right of Figure 2, merges the two classes having the minimal distance between two member objects.

### 2.3.2 Further developments

Like Galois connections for the GL approach, the definitions of sensible (dis)similarities is a key topic for SBC. Once the appropriate transformation of a dataset into a distance matrix is devised,

<sup>2</sup>Stands for Sequential Agglomerative Hierarchical Non-overlapping

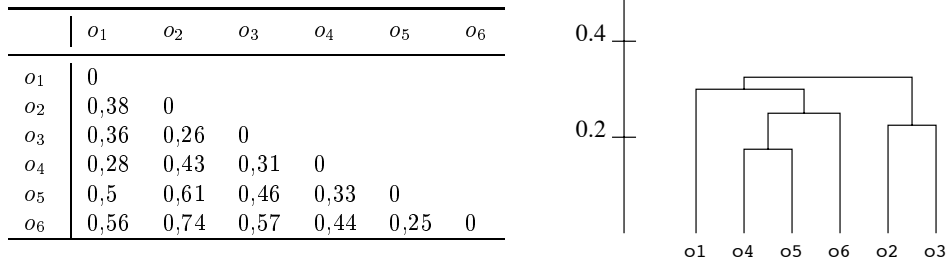


Figure 2: The dissimilarity matrix for the initial dataset and a *single-linkage* dendrogram

classes could be produced by (almost) any clustering technique. Similarity models for non-numerical homogeneous data descriptions, inclusive binary and categorical features, have been developed simultaneously, when not primary, to the all-numerical current [24]. Mixed attribute sets, i.e. with more than one of the above types, have been largely studied within the data analysis and machine learning fields [29]. Recent works on similarity for richer description languages include conjunctions of function-free predicates [5], conceptual graphs [21] and object-oriented formalisms [6, 27].

In its basic version, the hierarchical clustering does not provide an intensional description of the discovered classes. However, a characterization of a class can be easily obtained from its member objects. For this purpose, a generalization could be performed to yield a value restriction on each of the attributes shared by the class members. For example, let the classes  $c_1$  and  $c_2$  have member sets (extensions)  $\{o_4, o_5\}$  and  $\{o_4, o_5, o_6\}$  respectively. The respective sets of **age** values within the class are  $\{30, 32\}$  and  $\{30, 32, 35\}$  which generalize to the intervals  $[30, 32]$  and  $[30, 35]$ . Similar generalizations may be carried out on the entire set of common attributes whereby each data type that is used would require its own generalization operator<sup>3</sup>. The set of value restrictions produced in this way constitute a description for the class in the sense that each member object satisfies the all the restrictions. Here are some descriptions of the classes in Figure 1:

	$c_1 = \{o_4, o_5\}$	$c_2 = \{o_4, o_5, o_6\}$
age	$[30, 32]$	$[30, 35]$
salary	$[2.7, 2.9]$	$[2.7, 3.2]$

Class descriptions as presented above are very similar to concept intents in the GL framework as both could be seen as conjunctions of predicates that constrain (initial) attribute values. However, two important distinctions exist between them. First, the description of a class discovered by clustering provide only a *necessary* condition for class membership whereas a concept intent in the GL approach provides both *necessary and sufficient* conditions. Next, value restrictions in concept intents are fixed *a priori*, i.e., during the scaling step, and therefore they determine the shape of the concept lattice, whereas class descriptions depend on clustering results. Actually, the characterization task could be carried out either simultaneously with clustering [5] or after it, i.e., as a separate post-processing step [27].

### 3 Comparison of the two approaches

The two approaches can be compared on a variety of aspects related to the input and the output of the clustering as well as to the specific steps of the process. We selected a limited set of criteria which

<sup>3</sup>Suitable operators for the most frequent data types (e.g., numbers, symbols, etc.) may be found in [22].

discriminate them in the best way. These are: the kind of similarity the classes/concepts express, the output uniqueness, the structure of the clustering (partitioning versus overlapping), the specialization between classes/concepts, the use of attribute weights, and the ability to handle different kinds of data.

First, the notion of similarity in SBC is more precise than the one in GL. In fact, two objects in GL are similar if they share some identical properties, whereas in SBC they are alike to a degree that is quantified by a proximity measure over the entire attribute set. The same measure quantifies the dissimilarity between objects, a mechanism that is missing in GL. However, in both frameworks two similar objects tend to be grouped into concepts/classes that are specific, i.e., lay low in the classification structure.

Given a dataset, there are various ways to transform it into a working table, matrix or context, both in GL and in SBC. However, GL turns a context into a unique and complete lattice of concepts. In contrast, SBC produces a tree structure (i.e., recursive partitioning) over a small subset of all possible classes. Such a tree depends on the exact clustering algorithm used. Moreover, classes at each level of the tree are disjoint, whereas concepts in the lattice may overlap.

Furthermore, the generalization relation in both structures has distinct scopes. On one hand, in the concept lattice, generalization implies attribute set inclusion (e.g., concepts  $\{\{4,5\},\{q,s\}\}$  and  $\{\{4,5,6\},\{s\}\}$  in Figure 1). On the other hand, in the clustering tree, classes share all attributes but refine the respective value restrictions (e.g. classes  $c_1$  and  $c_2$  in Sec. 2.3.2).

While GL deals with all the attributes in a uniform manner, i.e. with a same prevalence, SBC allows the assignment of weights to attributes. Different sets of weights lead to distinct dissimilarity matrices which in turn may give rise to quite diverging class hierarchies. It is noteworthy, that discretization of a continuous attribute for scaling represents a particular kind of weighting values within the attribute domain: some couples of otherwise different values are considered as identical while other couples remain distinct.

Finally, a valuable feature of SBC is its ability to handle in a direct and uniform way different kinds of data such as qualitative (e.g., sex), continuous (e.g., salary) and discrete (e.g., number of owned cars) data. In contrast, GL needs a preprocessing (e.g., conceptual scaling [13]) of the input to keep the lattice in reasonable size, a step which introduces some bias. Moreover, SBC techniques are more easily adapted to structured and relational data than GL, especially in case of cyclic data (e.g., spouse relationship [27]).

The table below summarizes the main features.

Criteria	Galois Lattice	Similarity-based clustering
Similarity	Equality-based	Proximity-based
Uniqueness of the result	Y	N
Completeness of the final structure	Y	N
Specialization	Attribute set inclusion	Attribute range restriction
Attribute Weight	N	Y
Continuous value management	Hard	Y

## 4 Towards an Integration of the Two Approaches

The strengths and limitations of each approach discussed earlier hint at their complementarity. Therefore, a joint application of both may be foreseen that helps refine the clustering process over large sets of objects with a rich collection of attribute types (numerical, relational, etc.). We could suggest at least three plausible scenarios, although others are possible.

First, a preliminary SBC step may help successful processing of continuous data in a GL. The clustering tree may be used in two different ways to reduce the size of the GL context, and hence of the lattice to produce. On the one hand, instead of processing the entire object set in a direct manner, one may extract a set of highly cohesive classes from the tree to act as generalized individuals for the GL step. For example, the lattice on the right of Figure 3 has been obtained from the set of clusters at dissimilarity level 0.25 of the dendrogram in Figure 2 ( $\{1\}$ ,  $\{2,3\}$ ,  $\{4,5\}$  and  $\{6\}$ ). Each cluster is first characterized in the way described in Section 2.3.1, which gives rise to four generalized individuals (concepts  $c_1$  to  $c_4$  of the table on the left of Figure 3). Then, the Galois connection between the set of clusters and the set of all possible intents, i.e. couples of intervals, is used to compute the formal concepts. These are given on the left of Figure 3 both with their intents and their extents (in terms of *initial* objects).

concept	extent	intent	
		age	salary
$c_0 (\perp)$	$\emptyset$	$\emptyset$	$\emptyset$
$c_1$	$\{1\}$	$[26, 26]$	$[3., 3.]$
$c_2$	$\{2, 3\}$	$[23, 27]$	$[2.2, 2.3]$
$c_3$	$\{4, 5\}$	$[30, 32]$	$[2.7, 2.9]$
$c_4$	$\{6\}$	$[35, 35]$	$[3.2, 3.2]$
$c_5$	$\{1, 2, 3\}$	$[23, 27]$	$[2.2, 3.]$
$c_6$	$\{1, 4, 5\}$	$[26, 32]$	$[2.7, 3.]$
$c_7$	$\{2, 3, 4, 5\}$	$[23, 32]$	$[2.2, 2.9]$
$c_8$	$\{1, 6\}$	$[26, 35]$	$[3., 3.2]$
$c_9$	$\{4, 5, 6\}$	$[30, 35]$	$[2.7, 3.2]$
$c_{10}$	$\{1, 2, 3, 4, 5\}$	$[23, 32]$	$[2.2, 3.]$
$c_{11}$	$\{1, 4, 5, 6\}$	$[26, 35]$	$[2.7, 3.2]$
$c_{12} (\top)$	$\{1, 2, 3, 4, 5, 6\}$	$[23, 35]$	$[2.2, 3.2]$

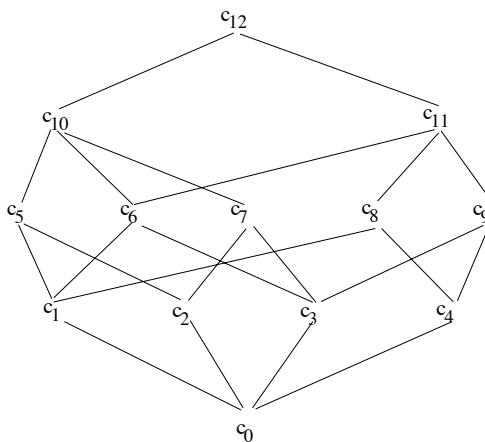


Figure 3: The set of concepts obtained by FCA with a straight Galois correspondence between individuals and descriptions. The individuals for the GL construction are intensionally described object groups that have been obtained by a preliminary conceptual clustering. The resulting lattice is shown on the right.

The concepts of the lattice built on top of the selected class set have their intents made up of value restrictions over the same attribute set as in the initial class descriptions. Therefore the new lattice represents a classification on the initial objects, i.e. the members of the selected classes. It is noteworthy that the specific nature of the Galois connection used, makes the concept intents fit perfectly to the extents. Indeed, if they are seen as sets of value restrictions in the sense of Section 2.3.2, then the resulting description is the most specific one that covers exactly the extent. In contrast, the descriptions provided by classical concept intents are not the most specific ones. To illustrate this phenomenon, let us take the concepts of extent  $\{1, 6\}$  in both lattices of Figure 1 and Figure 3 and compare their intents. In fact, the first intent indicates that both member objects share the property **salary** between 3.0 and 3.3 whereas the second one adds more restrictions to this: **salary** between 3.0 and 3.2 and **age** between 26 and 35. Observe that the new sub-concept relation is no more based on attribute set inclusion as all intents involve the whole set of attributes. The relation is rather based on value restrictions or, more precisely, on the inclusion of the set of admissible values for each attribute. Thus, the intent of the bottom concept is not the whole attribute set, but the intersection of all intents of object concepts which is usually void.

On the other hand, the clustering tree may be used in establishing the appropriate thresholds for scaling. For example, the following thresholds may be drawn from the classes of Figure 2.

age	[0, 29], [30, $\infty$ )
salary	[0, 2.6], [2.7, $\infty$ )

Conversely, SBC may be applied on an already available concept lattice in order to obtain a simplified structure in which only concepts that group similar objects appear. For that purpose, one may imagine a metrics on concepts that helps to group them into more general concepts. The procedure is identical to the hierarchical clustering, however, in addition to the merge operation over concept extents, a generalization on concept intents is performed. In case of binary object descriptions, as in the classical GL approach, the generalization is simply the set of common attributes<sup>4</sup>. This simplification procedure does not necessary yield a tree of concepts since problems may arise both with the completeness (in the sense of the Galois connection) of the obtained object clusters and with the existence of additional inclusion links between clusters, which follow the lattice structure<sup>5</sup>. An algorithm that completes all clusters to full concepts and adds the missing links has been described in [2]. The suggested approach is interesting, but remains limited to classical contexts with binary descriptions for both objects and concepts since the similarity measure used relies heavily on them. With a numerical function, as suggested here, much greater part of the initial object resemblance may be caught. An example of a small-size conceptual structure drawn from the lattice in Figure 1 by a SBC procedure is shown on the left of Figure 4. The proximity values from Figure 2 have been used to group the individual concepts. After each merge, a closure has been performed over the cluster intent which led to additional merges (e.g., between  $\{o_2, o_3\}$  and  $\{o_1\}$ ).

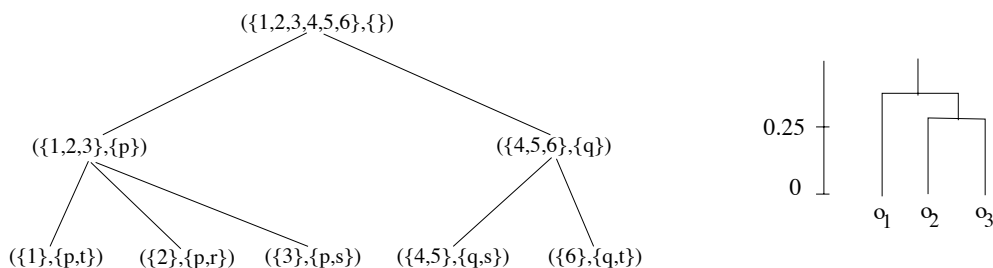


Figure 4: **Left:** a simplified concept hierarchy drawn from the concept lattice in Figure1 by a hierarchical clustering. The underlying proximity measure is computed on the individual objects (see the table in Figure 2). **Right:** a *single-linkage* dendrogram over the extent of the concept  $(\{1,2,3\},\{p\})$  based on the proximity matrix in Figure2.

A SBC may also be performed over the extent of a single concept of the lattice. In fact, concepts which stem from a derived context may provide quite a rough estimation of actual inter-object similarities. To catch those similarities, one could perform a SBC over the original object descriptions. For example, the dendrogram on the right of Figure 4 shows the result of a clustering over the extent of the concept  $(\{1,2,3\},\{p\})$  (see Figure 1) based on the measure in Figure 2. The discovered classes refine the initial concept but do not necessarily correspond to sub-concepts in the lattice. In this case, the result of the local clustering diverges<sup>6</sup> from the complete clustering tree in Figure 2 as

<sup>4</sup>Therefore, the generalization corresponds to the lattice *join* operator.

<sup>5</sup>Due to the sufficient condition for membership in the concept intents.

<sup>6</sup>In the sense that it does not represent a sub-tree.

well.

## 5 Conclusions and further research

Particular traits of the above approaches made them suitable for different kinds of applications. Within the object-oriented paradigm, both have been used to assist the design of class hierarchies. For instance, GL proved to be the right tool for designing the entire hierarchy from its leaf classes [18] and their features, methods and variables. The binary framework and the completeness of the lattice help insure properties like maximal factorization and minimal size of the entire class set. However, large parts of the lattice do not contribute effectively to the introduction of object features and may be ignored. Substantial work on simplification procedures for lattices has been done [17, 10]. In particular, hierarchical clustering of concepts based on binary similarity measures has been investigated in [2]. Alternative problems settings for GL class design include an optimization of existing hierarchies from documented uses [26]. SBC shows more flexibility and simplicity of the final hierarchy and, therefore, is better suited for domain model design where classes are built as clusters of domain entities. Thus, SBC has been essentially used within formalisms that allow such a loose class-instance relation, mainly object-based knowledge representations [6, 27]. However, some cases of SBC for class design have been reported [3] where simple class trees insured intelligible domain models and therefore better reuse of the entire hierarchy.

Our current concern is to carefully study and deepen some alternatives to the integration of both approaches and analyze their potential in software engineering fields such as refactoring. We believe that each of the scenarios we suggested in the paper could fit to a particular situation. For example, with an object set described by a rich set of data types, it seems more convenient to start with an SBC and then try an GL clustering (SBC-driven clustering). Conversely, in case of a very large object set with binary attributes, it may be relevant to start by building the lattice and then use a SBC procedure to prune it (GL-driven clustering).

## References

- [1] M. Anderberg. *Cluster analysis for applications*. Academic Press, 1973.
- [2] N. Anquetil and J. Vaucher. Extracting hierarchical graphs of concepts from an object set: comparison of two methods. In *Knowledge Acquisition Workshop, ICCS'94*, 1994.
- [3] H. Astudillo. Maximizing object reuse with biological metaphor. *Theory and Practice of Object Systems*, 3(4):235–251, 1997.
- [4] B. Birkhoff. *Lattice Theory*, volume 25. American Mathematical Society Colloquium Publ., Providence, revised edition, 1973.
- [5] G. Bisson. Conceptual clustering in a first order logic representation. In *Proceedings of the 10th European Conference on Artificial Intelligence, Vienna, Austria*, pages 458–462, 1992.
- [6] G. Bisson. Why and how to define a similarity measure for object-based representation systems. In N.J.I. Mars, editor, *Towards Very Large Knowledge Bases*, pages 236–246, Amsterdam, 1995. IOS Press.
- [7] P. Brito. *Analyse de données symboliques. Pyramides d'héritage*. Thèse de doctorat, Université Paris IX Dauphine, 1991.
- [8] L. Chaudron and N. Maille. First order logic formal concept analysis: from logic programming to theory. *Computer and Information Science*, 13(3), 1998.
- [9] M.-C. Daniel-Vatonne. *Les termes : un modèle de représentation et structuration de données symboliques*. Thèse de doctorat, Université Montpellier II, 1993.
- [10] H. Dicky, C. Dony, M. Huchard, and T. Libourel. On automatic class insertion with overloading. In *Proceedings of OPPSLA'96*, pages 251–267, 1996.

- [11] E. Diday and R. Emillion. Treillis de galois maximaux et capacités de choquet. *C.R. Acad. Sci. Paris*, 325(1):261–266, 1997.
- [12] M. Faid, R. Missaoui, and R. Godin. Knowledge discovery in complex objects. *Computational Intelligence*, 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. Girard. *Classification Conceptuelle sur des Données Arborescentes et Imprécises*. Thèse de doctorat, Université de la Réunion, 1997.
- [16] R. Godin. *L'utilisation de treillis pour l'accès aux systèmes d'information*. PhD thesis, Université de Montréal, 1986.
- [17] R. Godin and H. Mili. Building and maintaining analysis-level class hierarchies using Galois lattices. In *Proceedings of OOPSLA'93, Washington (DC), USA*, special issue of ACM SIGPLAN Notices, 28(10), pages 394–410, 1993.
- [18] R. Godin, G. W. Mineau, R. Missaoui, M. St-Germain, and N. Faraj. Applying concept formation methods to software reuse. *Journal of Knowledge Engineering and Software Engineering*, 5(1):119–142, 1995.
- [19] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.
- [20] R. Kent. Rough concept analysis: A synthesis of rough sets and formal concept analysis. *Fundamenta Informaticae*, 27:169–181, 1996.
- [21] P. Maher. A similarity measure for conceptual graphs. *International Journal of Intelligent Systems*, 8:819–837, 1993.
- [22] R. S. Michalski. A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, volume I, pages 83–134, Palo Alto, CA, 1983. Tioga.
- [23] 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 of *Lecture Notes in Computer Science*, pages 423–441. Springer-Verlag, Berlin, 1999.
- [24] G. Saporta. *Probabilités, Analyse des Données et Statistique*. Technip, 1990.
- [25] A. Simon and A. Napoli. Building Viewpoints in an Object-Based Representation System for Knowledge Discovery in Databases. In *1st International Conference on Information Reuse and Integration, IRI-99*, 1999.
- [26] G. Snelling 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.
- [27] P. Valtchev. *Construction automatique de taxonomies pour l'aide à la représentation de connaissances par objets*. Thèse de doctorat, Université Joseph Fourier, Grenoble I, 1999.
- [28] R. Wille. Restructuring the lattice theory: An approach based on hierarchies of concepts. In I. Rival, editor, *Ordered sets*, pages 445–470, Dordrecht-Boston, 1982. Reidel.
- [29] D.R. Wilson and T.R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6, 1997.