Machine Learning 24, 95–122 (1996)
© 1996 Kluwer Academic Publishers. Manufactured in The Netherlands.

A Lattice Conceptual Clustering System and Its Application to Browsing Retrieval

CLAUDIO CARPINETO carpinet@fub.it
GIOVANNI ROMANO romano@fub.it
Fondazione Ugo Bordoni, Via Baldassarre Castiglione 59, 00142 Rome, Italy

Editor: Douglas Fisher

Abstract. The theory of concept (or Galois) lattices provides a simple and formal approach to conceptual clustering. In this paper we present GALOIS, a system that automates and applies this theory. The algorithm utilized by GALOIS to build a concept lattice is incremental and efficient, each update being done in time at most quadratic in the number of objects in the lattice. Also, the algorithm may incorporate background information into the lattice, and through clustering, extend the scope of the theory. The application we present is concerned with information retrieval via browsing, for which we argue that concept lattices may represent major support structures. We describe a prototype user interface for browsing through the concept lattice of a document-term relation, possibly enriched with a thesaurus of terms. An experimental evaluation of the system performed on a medium-sized bibliographic database shows good retrieval performance and a significant improvement after the introduction of background knowledge.

Keywords: conceptual clustering, incremental learning, concept lattices, browsing, information retrieval

1. Introduction

Conceptual clustering is concerned with the problem of grouping unlabelled objects into classes. An important feature of conceptual clustering methods (Michalski & Stepp, 1983) is that any output class, in addition to being characterized by its extensional description (i.e., the set of objects covered by the class), is also characterized by an intensional (conceptual) description. A second feature that is not definitional of conceptual clustering, but which is often desirable, is that output classes (often referred to as concepts) are arranged into a hierarchy based on their generality/specificity. Another often desirable feature is that the process of class formation is incremental; that is, the processing of the nth object does not require extensive reprocessing of the $(n - 1)$ objects already seen.

Several incremental hierarchical conceptual clustering systems have been developed recently (Lebowitz, 1986; Fisher, 1987; Hanson & Bauer, 1989; Hadzikadic & Yun, 1989; McKusick & Langley, 1991; Martin & Billman, 1994). Although they differ in many respects (e.g., concept representation, hierarchy evaluation criteria, disjoint *versus* non-disjoint classes), they usually attack the problem by defining a set of hierarchy-change operators and carrying out a hill-climbing search through the space of possible concept hierarchies aimed at finding a “good” hierarchy (Gennari et al., 1989). These systems perform well in many experimental settings, but the criteria they use to modify and evaluate hierarchies are typically local and/or heuristic. In this paper we present an alternative approach to hierarchical conceptual clustering. Rather than finding and updating a particular subset of concepts (i.e., the output hierarchy), we keep and update *all* classes that can be generated in a restricted concept language. This is equivalent to building a virtually unbiased set of clusters, in much the same way as the version space approach (Mitchell, 1982) generates a set of alternative concept definitions during supervised concept induction.

Our approach relies on the theory of concept (or Galois) lattices (Wille, 1984; Davey & Priestley, 1990). This theory offers a formal and natural tool for restricting, representing, and ordering the set of concepts that can be induced over a collection of objects described by a set of properties. However, there has been relatively little work on the automation of the theory and its application (Godin et al., 1991; Carpineto & Romano, 1993a). In the paper we address these issues. We define an algorithm for incremental construction of a concept lattice, implemented in a system named GALOIS[1], and we report results showing that its time complexity grows at most quadratically with the number of objects in the lattice. The algorithm can also build a generalised version of a concept lattice, which we introduce to take into account background information about the properties describing the objects.

Of course, the possibility of automatically determining this kind of clustering structure raises the question of its application. Most hierarchical conceptual clustering systems have been seen as learning components of larger performance systems, such as systems for predicting the value of unknown attributes. GALOIS itself has been successfully used for predicting the class membership of unseen objects after training with classified objects (Carpineto & Romano, 1993a). However, there is another important, yet little explored, application of hierarchical conceptual clustering systems: knowledge browsing for information retrieval (Salton & McGill, 1983). Browsing through some kind of data space is one of the two principal techniques that have been developed to retrieve documents of interest from a bibliographic (or multimedia) database (Thompson & Croft, 1989; Marchionini & Shneiderman, 1988). Browsing can be used alone or in combination with querying, the other major paradigm of information retrieval; in fact, browsing allows a large spectrum of interactions with the database, ranging from casual inspection to query refinement. The typical data representation that supports browsing is a network in which nodes represent documents and edges represent relationships. This kind of structure is similar to that produced by most hierarchical conceptual clustering systems, with the important difference that, typically, it must be manually built.

In fact, conceptual clustering appears to be an attractive alternative for determining, indexing, and linking groups of documents automatically. In the paper we explore this issue. One basic research question we address is: what are the desirable features of a conceptual clustering system that supports browsing retrieval? We argue that the kind of cluster lattice produced by GALOIS satisfies most desiderata, thus making it a good candidate for supporting navigation through the content of a bibliographic database. To support this claim we experimentally evaluate the retrieval effectiveness of this approach. We have constructed a prototype user interface, GALOIS-Browser, which permits browsing as well as a limited query formulation through a document lattice. The retrieval interface is used to perform subject searching on a medium-sized bibliographic data base, including a thesaurus to build a generalised concept lattice. As there is a recent demostration that browsing an ungeneralised concept lattice may result in good absolute retrieval performance (Godin et al., 1993), we are particularly interested in evaluating the relative performance of GALOIS with or without background knowledge. The results are encouraging; in particular, they show a significant improvement in recall.

The rest of the paper is organized in the following manner. In Section 2 we introduce the basics of the theory of concept lattices. Section 3 is devoted to construction and analysis of concept lattices: in particular, we describe the algorithm for incrementally determining the lattice and we analyse its time and space complexity. In Section 4 we introduce a generalised version of concept lattices that can incorporate graph-structured information over the attributes describing the objects. In Section 5 we characterize conceptual clustering in general, and concept lattices in particular, as a support for browsing retrieval, then describe the browsing interface, and evaluate its retrieval effectiveness. Section 6 contains a discussion of related work, while Section 7 concludes the paper with a summary and some directions for future work.

2. Basics of the theory of concept lattices

We first give an informal characterization of a concept lattice and then describe it more formally. As an illustration we will refer to a very simple bibliographic database consisting of six documents described by eight index terms (see Table 1). The concept lattice relative to the first five documents of the table is illustrated in figure 1. Each node of the lattice is a pair, composed of a subset of the documents and a subset of the index terms; in each pair, the subset of terms contains just the terms shared by the subset of documents, and, similarly, the subset of documents contains just the documents sharing the subset of terms. The set of pairs is ordered by the standard set inclusion relation applied to the set of documents and terms that describe each pair. The partially ordered set is usually represented by a Hasse diagram, in which there is an edge between two nodes if and only if they are comparable and there is no other intermediate concept in the lattice (i.e., each node is linked to its maximally specific more general nodes and to its maximally general more specific nodes). The ascending paths represent the subclass/superclass relation; the bottom concept is defined by the set of all terms and contains no documents, the top concept contains all documents and is defined by their common terms (possibly none).

Table 1. A simple bibliographic database represented as a document by term matrix.

	Artificial intelligence	Expert systems	Information retrieval	Cataloguing	Indexing	Information science	Information retrieval systems	Knowledge-based systems
1	x	x	x					
2	x	x		x				
3	x	x			x			
4	x	x				x		
5	x	x		x			x	
6	x		x					x

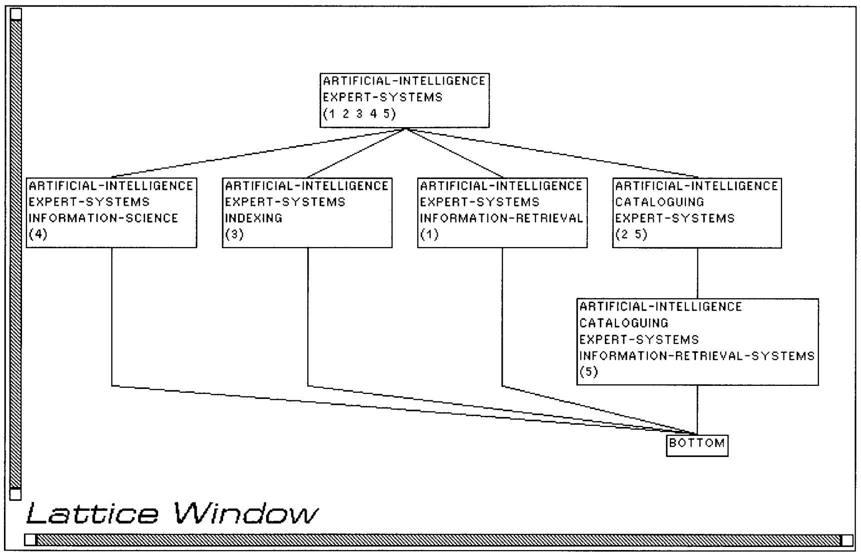


Figure 1. The concept lattice relative to the first five objects of the context in Table 1.

A more formal treatment of concept lattices begins by considering a binary relation between a set of objects (O) and a set of descriptors (D), called *context*. Therefore a context is a triple (O, D, I) where $I \subseteq O \times D$. We write oId, meaning the object o has the descriptor d. We may think of the set of descriptors associated with an object as a bit vector. Each bit corresponds to a possible descriptor and is on or off depending upon whether an object has that descriptor[2].

Formally, a concept of the context (O, D, I) is defined to be a pair (X, Y) where $X \subseteq O$, $Y \subseteq D$, and $X = \{o \in O \mid (\forall d \in Y) oId\}, Y = \{d \in D \mid (\forall o \in X) oId\}$; that is, X is the set of all objects possessing all the descriptors in Y and Y is the set of all descriptors common to all objects in X. X and Y are called the *extent* and the *intent* of the concept, respectively. Therefore only some pairs (X, Y)—i.e., the pairs that are complete with respect to I according to the given definition[3]—represent admissible concepts. For instance, in the lattice relative to the context in Table 1 there cannot be any pair having an intent equal to EXPERT-SYSTEMS, because all documents having EXPERT-SYSTEMS also have ARTIFICIAL-INTELLIGENCE.

The set of all concepts of the context (O, D, I) is denoted by $C(O, D, I)$. An ordering relation $(\leq)$ is easily defined on this set of concepts by

$$(X_1, Y_1) \leq (X_2, Y_2) \leftrightarrow X_1 \subseteq X_2 \tag{1}$$

or, equivalently, by

$$(X_1, Y_1) \leq (X_2, Y_2) \leftrightarrow Y_1 \supseteq Y_2. \tag{2}$$

$C(O, D, I)$ along with $\leq$ form a partially ordered set, that can be characterized by the Fundamental Theorem on Concept Lattices. The most important result stated by the fundamental theorem is that $(C(O, D, I); \leq)$ is a complete lattice[4] in which the intent of the least upper bound of a set of concepts is given by the intersection of the intents of the concepts, and, dually, the extent of the greatest lower bound of a set of concepts is given by the intersection of the extents of the concepts (see Davey & Priestley (1990) for the complete theorem and its proof).

3. Incremental determination of a concept lattice

Determining a concept lattice along with its Hasse diagram is a computationally difficult problem (Guénoche, 1990). Furthermore, the algorithms that have been proposed are—with one notable exception (Godin et al., 1991)—non-incremental, whereas an important requirement of many tasks is incrementality. In the first part of this section we describe the incremental algorithm used by GALOIS to build the concept lattice; then we study the space complexity of the problem and analyse the time complexity of our algorithm.

3.1. GALOIS: Description of the algorithm

We start by giving an informal characterization of the process of adding a new object to a given lattice. First we note that existing concepts are never removed from the lattice. An existing concept may contain the new object (i.e., its intent is more general than or equal to the new object's description), and therefore the concept's extent is augmented by the new object. If an existing concept does not contain the new object (i.e., its intent is incomparable to or more specific than the new object's description), then the concept is not affected at all by the object's introduction. In addition to modifying the extent of the current concepts in which an object is contained, the new object may cause new concepts to be added to the lattice. This happens whenever the intersection of the new object with any set of *objects* with which it shares some descriptor is not already present in the lattice. Any new concept introduced in the lattice must be consistently linked to the other concepts, including other new concepts created by an object's introduction. This will, in general, cause the elimination of some edges (i.e., the edges between all pairs of concepts C_1 and C_2 such that $C_1 > C_{\text{new}} > C_2$). The incremental determination of the concept lattice poses three computational problems: (a) generating all new nodes in the lattice, (b) avoiding generation of nodes that are already present in the lattice, and (c) updating the edges.

To generate all new nodes it is sufficient to consider the intersections of the new object with each concept in the current lattice. The main loop of our update algorithm adds such intersections to the lattice and places edges appropriately; while doing so the algorithm works only with intents (that is, logical conjunctions of descriptors) and exploits the basic result (see the fundamental theorem) that the intent of the least upper bound of two concepts is the intersection of the intents of the two concepts. However, the concepts in the current lattice are not examined independently of each other. Roughly, GALOIS compares the parents of each node to the intersection (of intents) of the node and the new object. If there is a parent with an intent that equals (=) the intersection, then the new concept is not created because it is already present in the lattice. If there is a parent with an intent that is a proper superset ($\supset$) of the intersection, then the new concept is not created since it will be generated when we intersect the new object with that parent of the node. Otherwise (i.e., if all parents have intents that are proper subsets ($\subset$) of the intersection or if all parents are incomparable to the intersection), then a new concept is actually added to the lattice.

For the problem of setting the appropriate edges there is no simple solution because any new node has to be consistently linked to other new nodes, and the new nodes to be added to the lattice cannot be generated in an ordered fashion. The LINK procedure that GALOIS uses operates on the lattice that exists at the time a new node is created (i.e., this lattice contains all nodes that existed prior to the introduction of a new object, and new nodes that have been created and linked in since the new object was first introduced). It determines, for each new node two boundary sets: the lower boundary set, S, contains the most general concepts that are more specific than the new node, and the upper boundary set, G, contains the most specific concepts that are more general than the new node. Then it links the new node to each element in S and G, and removes the edges between S and G (if any)[5].

Table 2 gives the complete description of the updating algorithm, which can be used in an incremental fashion to build the lattice from a set of objects. The invocation of the algorithm is preceded by an initialization step that adds the concept corresponding to the object to the

Table 2. The algorithm for updating a concept lattice.

```
UPDATE-LATTICE (lattice new-object)
updated-lattice ← lattice
**For** each concept in lattice
    **do**
    intersection ← concept ∩ new-object
    **unless** (**or** (intersection = empty)                          ;a
                (intersection = concept)                        ;b
                (there is parent(concept) = intersection)       ;c
                (there is parent(concept) ⊃ intersection))      ;d
          new-node ← (create-node :intent intersection)
          updated-lattice ← (LINK new-node updated-lattice)
  **finally** return the updated-lattice

LINK (node lattice)
Find lower boundary set of node in lattice
Find upper boundary set of node in lattice
Eliminate edges between lower and upper boundary sets
Add edges between node and each element in the lower boundary set
Add edges between node and each element in the upper boundary set
Return the updated-lattice
```

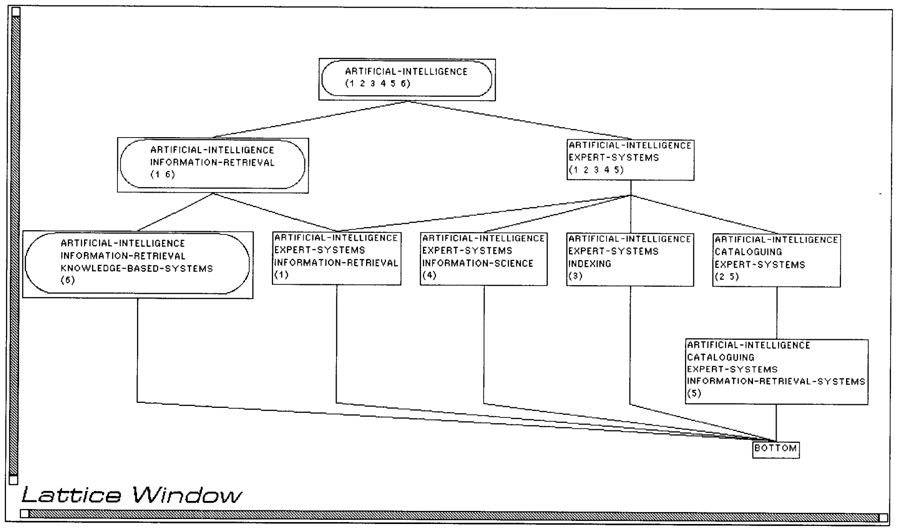


Figure 2. The concept lattice after the introduction of the sixth document.

lattice and sets the links; if the concept corresponding to the object is already contained in the lattice, then update does not take place. In Appendix A we give the formal proof of the correctness of the algorithm. It should be noted that this version of the algorithm is consistent with, but significantly more efficient than, the algorithm given by Carpineto & Romano (1993a) due to elimination of many unnecessary calls of function LINK.

To see the algorithm at work, consider the concept lattice shown in figure 1. Suppose we introduce the sixth document shown in Table 1. GALOIS returns the updated lattice shown in figure 2, where the new nodes are circled. Initialization causes a new node with intent AI-IR-KBS (i.e., Artificial-Intelligence ∩ Information-Retrieval ∩ Knowledge-Based-Systems) to be created and linked to the bottom. Then the algorithm iterates on the concepts in the old lattice. Examination of concepts AI-CAT-IRS-ES, AI-CAT-ES, AI-ES-IND, and AI-ES-IS leaves the lattice unchanged (case *d*). Concept AI-ES returns the node with intent AI, the sets $S =$ [AI-ES, AI-IR-KBS] and $G =$ [empty], and the edges (AI-ES, AI), (AI-IR-KBS, AI). Concept AI-ES-IR returns the node AI-IR, the sets $S =$ [AI-ES-IR, AI-IR-KBS] and $G =$ [AI], the edges (AI-ES-IR, AI-IR), (AI-IR-KBS, AI-IR), (AI-IR, AI), *and eliminates the edge (AI-IR-KBS, AI)*. Note that examining the concepts in a different order would produce different temporary edges, but the same final lattice.

In the next two sections we study the space complexity of concept lattices and evaluate the time complexity of the updating algorithm.

3.2. Space complexity

There is a theoretical upper bound on the number of concepts present in the lattice that depends on the size of the object-description space. In the following discussion we refer to

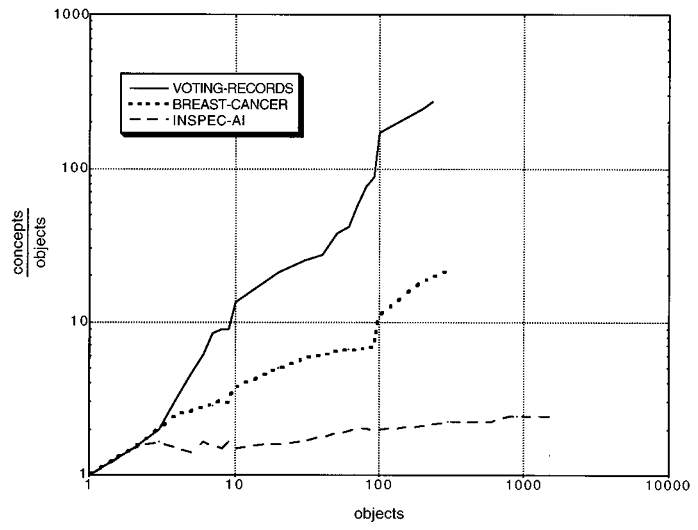


Figure 3. Experimental space complexity of concept lattices. Both scales are logarithmic.

a context in which objects are described by multi-valued attributes, rather than by single-valued attributes, as in the information-retrieval context. Assuming that there are m attributes with v values each, the conjunctive language used to describe the elements in a concept lattice contains $(v+1)^m$ possible concepts. When *all* possible v^m distinct objects described by all descriptors are seen, the corresponding lattice will contain indeed $(v+1)^m$ concepts (namely, $\binom{m}{0}v^m$ m-attribute concepts, $\binom{m}{1}v^{m-1}$ $(m-1)$-attribute concepts, etc). For practical values of v, m, and the number of objects, this theoretical bound is very high, so it is useful to see how the number of concepts empirically varies with the number of objects in some real domains.

We did an experimental study using the two natural data sets Voting Records and Breast Cancer, and the bibliographical database INSPEC-AI. For the first two data sets, found in the UCI data repository, a detailed description is given by Anderson & Matessa (1992); suffice it to say that they contain 435 and 286 instances, respectively, described by 16 and 9 nominal attributes. The database INSPEC-AI will be better illustrated in Section 5.3; it contains 1555 documents, where the number of descriptors per document varies, but is usually small. For the experiment, we ran GALOIS on each data set, computing the size of the corresponding concept lattice. The results are shown in figure 3. A lattice could not be fully constructed for the Voting Records data set because GALOIS reached the space limit of the hardware platform on which we performed the experiment (i.e., a SPARCstation LX equipped with 32 Mbytes of RAM). In particular, we observed that as the lattice grew to about 150,000 nodes, the time necessary to incorporate another object into the lattice started increasing dramatically due to memory fragmentation and garbage collection. Thus, to construct the curve in figure 3 for the Voting Records data set, we divided the data set into five randomly-selected, equally-sized subsets such that the size of each subset (360 instances) was less than that which would violate the hardware space constraint of 150,000 nodes. We then ran GALOIS on each subset to get an averaged growth curve. Curves for the other two data sets were obtained by running Galois once on the entire data set in each case. As shown in figure 3, the growth in the number of concepts ranges from linear to quadratic with respect to the number of objects[6]. The lower bound was obtained with the data set, INSPEC-AI, in which, as in most document collections, there are few descriptive terms per object, and thus the likelihood of intersections is low. The upper bound was obtained with the Voting Records data set, in which there are relatively many attributes and few values per attribute, and thus the likelihood and diversity of intersections is quite high.

These results provide some insight into the storage requirements of concept lattices. In particular, while they confirm earlier results obtained by Godin et al., (1991) for bibliographic databases, they also imply that for other domains, the size of the lattice may grow as a quadratic, rather than as a linear, function of the number of objects. These results, however, tell us little about the effect that two primary parameters governing the size of the descriptor space, namely number of attributes and number of values per attribute, have on the size of the lattice. Let us study this problem for the simple case in which the instances are uniformly distributed over the object description space.

We would like some formula for the expected number of nodes in the lattice (N) as a function of all three input variables—number of objects (p), number of attributes (m), number of values per attribute (v)—under the assumption of uniform distribution of the attribute values. Ideally, the formula can be obtained by summing up the probabilities of occurrence of each possible node. To make this computation feasible, it is convenient to think of each node as a pair formed by a subset of objects and a subset of attribute values, and to consider the subsets of nodes formed by fixed numbers of objects and of attribute values. There are $\binom{p}{i}\binom{m}{j}v^j$ possible nodes formed by i objects and j attribute values, where each node has the same probability of occurrence. Recalling the definition of concept in a concept lattice, this probability can be obtained as the product of three factors, namely the probability that the i objects share the j attribute values (i.e., $(\frac{1}{v})^{ji}$), the probability that the i objects do not share other attribute values (i.e., $[1-(\frac{1}{v})^{i-1}]^{m-j}$), and the probability that there exists no other object having the j attribute values (i.e., $[1-(\frac{1}{v})^{j}]^{p-i}$). This explains the first term in the following expression for the mean number of nodes in the lattice. To derive the complete formula we have to add the probability of the two spurious nodes that have been left out, namely the probability of the lattice top (second term) and the probability of the lattice bottom (third term):

$$N = \sum_{i=1}^{p}\sum_{j=1}^{m}\binom{p}{i}\binom{m}{j}v^j\left(\frac{1}{v}\right)^{ji}\left[1-\left(\frac{1}{v}\right)^{i-1}\right]^{m-j}\left[1-\left(\frac{1}{v}\right)^{j}\right]^{p-i} + \left(1-\frac{1}{v^{p-1}}\right)^{m} + \left(1-\frac{1}{v^{m(p-1)}}\right). \tag{3}$$

Figure 4 shows results plotted from formula (3) for four pairs of values for (m, v): (5, 5), (10, 10), (50, 10), (50, 50). As p grows to infinity, N/p tends to zero because,

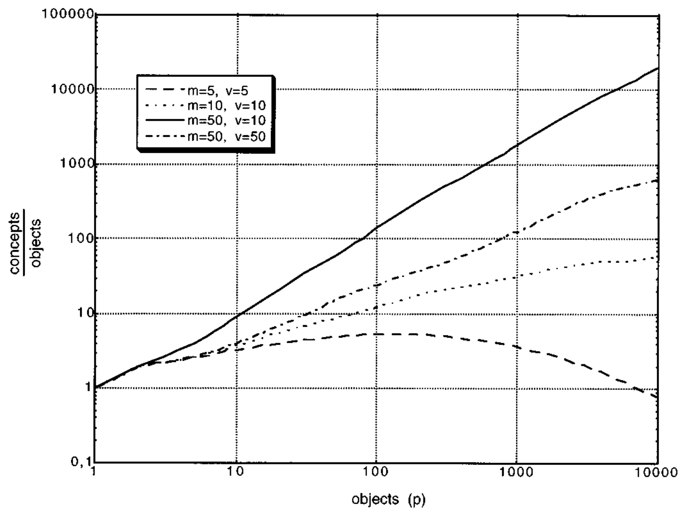


Figure 4. Theoretical space complexity of concept lattices as a function of data complexity. Both scales are logarithmic.

as said earlier, for fixed values of m and v, N has an upper bound of $(v+1)^m$. This phenomenon is apparent for the lower curve in figure 4. In this case the upper bound is given by $6^5 = 7776$ nodes; thus, after a certain point the lattice is saturated and N/p drops below one. Aside from this limit situation, the three other theoretical curves are consistent with the experimental curves reported in figure 3. In particular, while they confirm that the growth of the number of nodes varies from linear to quadratic with respect to the number of objects, they also show the two situations in which the space complexity moves towards the upper bound. This is when the ratio between m and v increases, or, less markedly, when both m and v grow by a same factor and the number of objects is not too small.

Before turning to time complexity, we should emphasize that we checked the correctness of the results derived from formula (3) experimentally. For all combinations of p, m, and v, for which the lattice could be easily built by GALOIS, we created corresponding artificial data sets with random assignment of values, then ran GALOIS on them. Comparisons between experimental and theoretical results showed remarkable similarity across parameter combinations.

3.3. *Time complexity*

For each object, GALOIS iterates on the concepts in the lattice. Assuming that there are m attributes with v values each, there are at most $(v+1)^m$ concepts in the lattice. Each iteration may have to apply the LINK function, which may require all concepts above the new object or the current iteration node ($2v^m$ concepts) be examined. Therefore the worst-case time complexity to build a lattice from N objects is proportional to

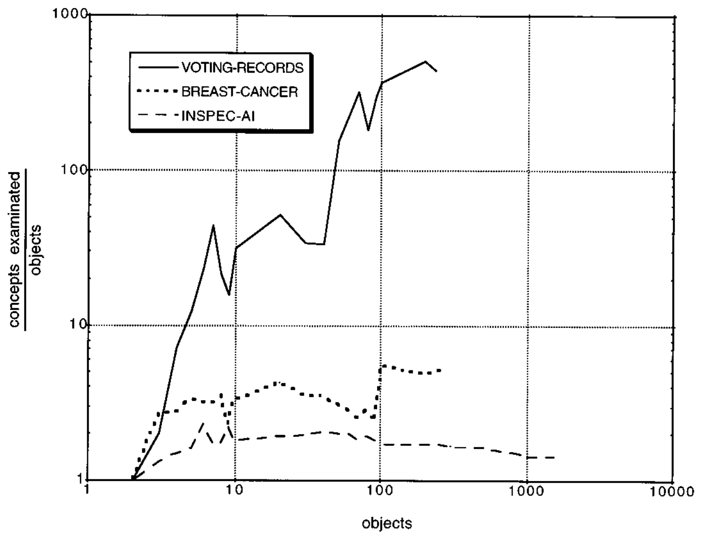


Figure 5. Experimental time complexity of GALOIS. Both scales are logarithmic.

$(v+1)^m 2v^m N < O((v+1)^{2m} N)$. If the number of descriptors has an upper bound, as usually happens, the asymptotic behavior is linear in the number of objects. In practice, however, the constant factor relative to each update may be very large. To evaluate GALOIS' experimental behavior, we plotted the number of nodes examined in each update by the key operation of GALOIS (i.e., determining the lower boundary set S within the function LINK) versus the number of objects. Figure 5 shows that the growth varied, with some oscillations, from linear to quadratic with respect to the number of objects processed; thus the time complexity was nearly linear *in the number of concepts* in the lattice. This result might seem surprising, in that, as noted at the beginning of this section, the time complexity should range from linear to quadratic with respect to the number of concepts. The fact that the experimental complexity was nearly linear can be explained by observing that (a) in each update the number of calls of function LINK never exceeds a small fraction of the number of concepts in the lattice, and (b) each application of function LINK requires examining only a small subset of the possible $2v^m$ concepts in the lattice that are above the new object or the current iteration node. In practice, the run time necessary to determine the *whole* lattice corresponding to INSPEC-AI and Breast Cancer (which has few attributes and a relatively large number of values per attribute) was less than 20 minutes, and the insertion time of one object never exceeded 15 seconds. For the data set Voting Records, as we noted earlier, GALOIS ran into computational barriers; thus the curve for this data set shown in figure 5 was obtained following the method explained in Section 3.2.

Admittedly, for large non-bibliographic databases the construction of the whole concept lattice may present computational problems, both of space and time. However, this problem may be alleviated in several ways. An alternative method to circumvent the kind of system crash we experienced, for instance, would be to keep the lattice on disk rather than in main memory, and to make each update on disk, as in Godin et al. (1991). Lattice construction would be much slower, of course, but we could try to improve its efficiency by appropriate caching; for instance, given that at any invocation of the function LINK all the nodes that are above the new object have to be examined, we could cache this subset of nodes in main memory. In addition, we can resort to general methods of concept lattice manipulation that may trade off time and memory in the determination of such structures. We might apply the methods that have been developed to compose or decompose concept lattices (Ganter, 1988), although they work only in certain cases.

4. Adding background knowledge to concept lattices

In this section we introduce a generalised definition of concept lattice that allows incorporation of background knowledge into the clustering structure. In particular, we will exploit a common source of background knowledge, namely possession of structured information over object descriptors (Thompson et al., 1991), to influence clustering. Our discussion will be restricted to those domains in which objects are described by single-valued attributes, such as in information retrieval; for the extension to domains in which objects are described by multi-valued attributes, such as databases, (see Carpineto & Romano (1993b)).

4.1. Concept lattices with graph-structured descriptors

We will assume that a partial-ordering relation ($\leq_{D^*}$) is defined over a set of descriptors D^*, where D^* is a superset of the descriptors (D) actually used to describe the objects, and that the partial-ordering relation can be represented by any directed acyclic graph (not just a tree). This kind of information is naturally available in many domains; in particular, it is often provided in most information retrieval tasks, where it is usually encoded in a thesaurus of the terms describing the documents. Figure 6 shows an example of such background information relevant to the data of Table 1. The intended meaning of the ordering relation is that any term implies any of its more general terms; for instance, CATALOGUING implies INFORMATION-ANALYSIS and COMPUTER-APPLICATIONS. Keywords that do not appear in the

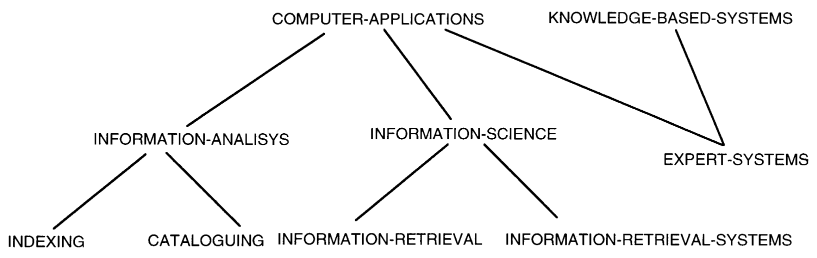


Figure 6. Graph-structured background information over the terms describing the database of Table 1.

graph, such as ARTIFICIAL-INTELLIGENCE, are incomparable according to the given ordering relation; they can be thought of as children of the most general term ANY. The graph also contains new terms, such as INFORMATION-ANALYSIS and COMPUTER-APPLICATIONS, which did not appear in the document descriptions.

The theory of concept lattices has been developed for unstructured descriptors, but it can be extended to cope with richer object description languages, including this kind of background information. One simple method for using structured descriptors in a concept lattice would be to expand the matrix describing the data, adding to the description of each document all the keywords that are implied by the original keywords. At this point one could treat the new context as if it contained unstructured descriptors, thus avoiding any special processing; however, the space necessary to store the lattice would increase considerably and the user would find it difficult to read and compare the relative concepts representations. In GALOIS we have taken an alternative approach. The idea is to work only with a minimal concept representation and to adapt the basic definitions of concept lattices to take into account the implicit keywords. GALOIS begins by removing from each object description all implied keywords, if any. Then it employs a modified definition of the ordering relation; definition (1) in Section 2 is left unchanged while definition (2) is replaced by

$$(X_1, Y_1) \leq (X_2, Y_2) \leftrightarrow \forall d_2 \in Y_2, \exists\, d_1 \in Y_1, d_1 \leq_{D^*} d_2 \tag{4}$$

The last step consists of changing the definition of intent intersection, for which the standard set intersection operator can no longer be used. The generalised operator can be best described operationally: the intersection of the intents of two concepts (X_1, Y_1) and (X_2, Y_2) is obtained by finding for each pair (d_1, d_2) $d_1 \in Y_1, d_2 \in Y_2$, the most specific keywords in D^* that are more general than d_1 and d_2, and then retaining only the most specific elements of the set of terms generated in this way[7]. With these new definitions GALOIS builds a generalised concept lattice which, besides preserving the basic properties of ungeneralised concept lattices, has the additional feature that more general terms index more general classes, whereas in an ungeneralised concept lattice the ordering relation over the classes is independent of the possible ordering relation over the terms.

A lattice with structured descriptors is a fair generalisation of a lattice with unstructured descriptors, in that unstructured descriptors can always be converted into structured descriptors using a dummy root ANY. However, we keep both the standard and the generalised version of the involved functions for reasons of efficiency. Thus, in practice GALOIS has two working modes, with and without background knowledge.

4.2. *The effect of background knowledge in the bibliographic context*

Consider again the bibliographical context of Table 1, but this time assume that GALOIS works with the background information displayed in figure 6. The new concept lattice (L^*) output by GALOIS is shown in figure 7. Due to definition (4), in this kind of lattice a node may be more specific than another node while having fewer keywords. For instance, the node whose extent is (1 2 3 4 5) is more specific than the node whose extent is (1 2 3 4 5 6),

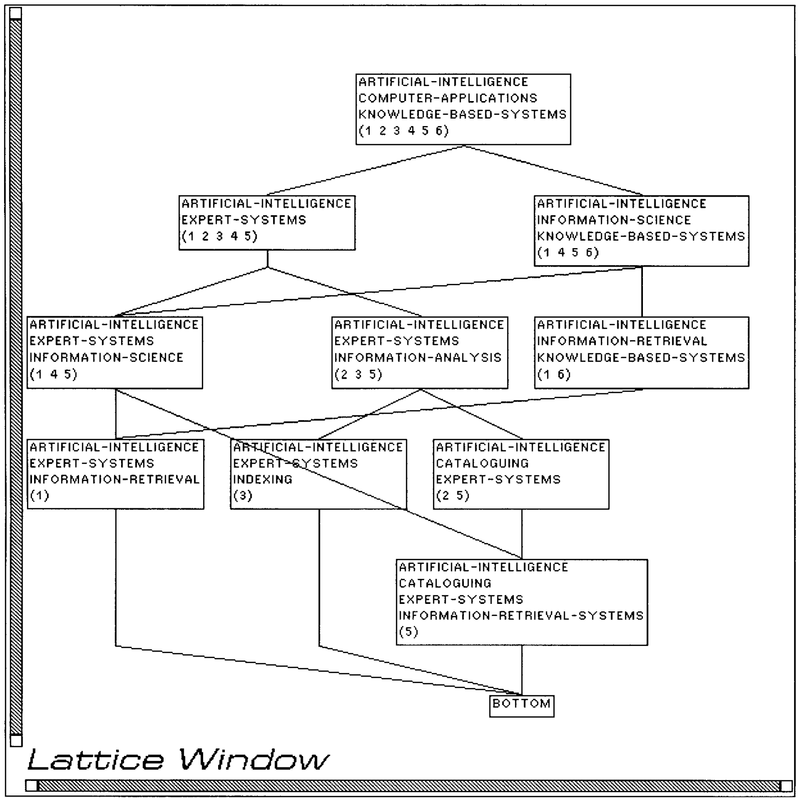


Figure 7. The concept lattice of the six documents with background knowledge.

but the former has fewer keywords; in fact (see the keyword graph in figure 6), the term EXPERT-SYSTEMS is more specific than both COMPUTER-APPLICATIONS and KNOWLEDGE-BASED-SYSTEMS.

The size of the lattice, when passing from an unstructured to a structured and possibly larger set of keywords, may decrease or it may increase. To explain this, consider that term structuring alone reduces the number of concepts: this will happen whenever two formerly distinct concepts come to have the same description after term expansion, thus collapsing them into a single concept. Term structuring may also cause two incomparable concepts to become comparable; in this case the set of concepts remains the same while the ordering relation changes. For instance, nodes (1), (4), (5) are incomparable in L whereas the intent of (4) is more general than that of both others in L^*. On the other hand, addition of new terms is likely to produce new concepts by revealing similarities between sets of objects that were formerly ungrouped (e.g., node (1 4 5 6) in L^*). The overall behavior depends on the specific data, on the new set of terms, and on the ordering structure over the terms, although we observed that the inclusion of background knowledge usually results in a noticeable increase in the lattice size.

4.3. Complexity of GALOIS with background knowledge

We should briefly consider how the time complexity varies when GALOIS works with structured descriptors. The theoretical bounds found when GALOIS runs with unstructured descriptors remain the same. However, those bounds refer to the number of comparisons between concepts and objects and comparisons between concepts, without considering the cost of the comparison itself. In the case of unstructured descriptors, each such comparison requires matching the corresponding bits of each descriptor, making the time necessary proportional to the number of descriptors. In the case of structured descriptors, for each descriptor in one concept, GALOIS must check its descendants in the descriptor-structuring graph to see if it can find some descriptor of the other concept. In fact, to solve this problem GALOIS employs an efficient procedure based on marker propagation, which requires that for each descriptor the algorithm visit all the elements below the descriptor only once (Carpineto & Romano, 1993b). Therefore, the time complexity in the structured case is greater than the unstructured case by a factor that is, at most, equal to the number of descriptors, but usually much less.

With this section we have completed the analysis of the system. Now we can turn to its application to browsing retrieval.

5. Conceptual clustering for supporting browsing retrieval

In this section we discuss the role conceptual clustering may play to support browsing. Browsing through a data space is a useful and familiar information retrieval technique (Marchionini & Shneiderman, 1988; Thompson & Croft, 1989; Cutting et al., 1992), as well as a major paradigm of recent hypertext and hypermedia technology (Nielsen, 1990). Browsing can meet a variety of the information needs of a user interacting with a database; it is suitable for making the non-expert user aware of the content of the database, for exploring new domains, and for helping the user refine early stages of direct query specification. The main problem in browsing is creating atomic pieces of information and linking them. Automatic methods of knowledge summarization and structuring may be very useful to achieve this goal without resorting to subjective and time-consuming decisions. In fact, browsing appears to be an important, yet little explored, challenge for conceptual clustering techniques.

5.1. Concept lattices as retrieval support structures

In the following we will refer to a typical browsing retrieval task in which there is a collection of documents described by a set of keywords and the user wants to find documents of interest by navigating through the content of the collection. Hierarchical conceptual clustering presents three basic features for supporting this task automatically: classes are generated from the usual document-term relation, classes are indexed through a conceptual description, and classes are linked using a generality/specificity relation. However, the task in question suggests other desirable properties for supporting cluster structure. First, graph navigation is more flexible than tree navigation. While in a strict hierarchical clustering each class has exactly one parent, in a lattice clustering there are many paths to a particular class. This facilitates recovery from bad decision making while traversing the hierarchy in search of documents (Godin et al., 1993). Second, the same document is often relevant to two or more queries that happen to have incomparable descriptions. Therefore, the ability to deal with non-disjoint classes is an important feature of browsing retrieval systems; lattice conceptual clustering naturally supports this functionality, as opposed to hierarchical conceptual clustering. Third, in information retrieval domains there is usually an available body of background knowledge expressed as a thesaurus of terms. The ability to incorporate such knowledge into the conceptual clustering of documents may considerably improve retrieval performance.

Most clustering systems can handle some of these issues but not all of them. In particular, the statistical clustering methods that have been predominately used in information retrieval (see Willet (1988) for a thorough review) do not produce conceptual descriptions of the generated classes, which usually makes the resulting cluster structure difficult to navigate (e.g., van Rijsbergen & Croft, 1975) or inefficient to query by direct search methods, (e.g., Maarek et al., 1991; Crouch et al., 1989). By contrast, the kind of concept lattice produced by GALOIS meets all the desiderata mentioned above. Furthermore, it presents two useful additional properties. The first is that, in addition to supporting browsing, a concept lattice of documents also allows an easy form of direct query specification. Each node in the lattice can be seen as a query formed by a conjunction of terms (the intent) with the retrieved documents (the extent). Therefore the lattice supports both direct retrieval of conjunctive queries and navigation from specific to general (or general to specific) queries. A second characteristic is that the lattice allows gradual enlargement or refinement of a query. More precisely, following edges departing upward (downward) from a query produces all minimal conjunctive refinements (enlargements) of the query with respect to that particular database (Godin et al., 1989).

We now describe our design and implementation of a browsing retrieval interface operating on a generalised document lattice.

5.2. An interface for browsing and querying in a generalised concept lattice

To enable the interaction between the user and the document lattice we have implemented a visual interface on top of GALOIS, named GALOIS-Browser[8], which shows the retrieval space to the user and lets him or her choose the next query to select. On the conceptual level, the user interacts with the system by jumping from one query to another in the lattice; the transition to other queries is driven either by the Hasse diagram or by direct term specification by the user. The main problem in the interface design is the visualisation of the retrieval space. As the structure is too large for limited display resolution, GALOIS-Browser shows only a small portion of the structure. The display window is centered on the current node and contains all nodes adjacent to it; for each such node, the terms and number of the documents are displayed[9].

GALOIS-Browser allows two basic interaction modes. In pure browsing mode a user selects a neighboring node by direct graphical manipulation (Shneiderman, 1987). The other way to make a jump to a node is through query formulation by menu selection. The new query can be formulated in two ways: either the user specifies the new terms from scratch, or modifies the current query. In the latter case, the user can remove a term, add a new term, or specialise/generalise a term using the information contained in the thesaurus. Query specification by term specialisation/generalisation lets the user make large jumps through the lattice driven by the semantics of terms. When a new query is input to GALOIS-Browser, it first checks that the query contains only minimal terms according to the term graph, and then jumps to the node of the lattice equal to the query; if there is no such node it jumps to the most general node(s) containing all the terms of the query, provided that the lattice contains some more specific node than the query. One of the most interesting features of GALOIS-Browser is that, since the underlying retrieval space is unique, all these interaction modes can be naturally combined in a hybrid search strategy. For instance, the user can start with a keyword search followed by term specialisation to locate a subspace of interest, and then browse through the subspace. The interface, which consists of about 70 Kilobytes of Lisp code, takes advantage of the many standard interface tools offered in the Symbolics Lisp Machine environment, such as dynamic windows, pointing and clicking devices, and graphical display routines.

In Appendix B, we illustrate the interface by showing a simple browsing session on the generalized concept lattice of figure 7. It is worth noting that the interface can also work on an ungeneralised lattice; in this case the set of available commands is restricted accordingly.

5.3. *Experimental evaluation of GALOIS as a browsing retrieval system*

In this section we evaluate GALOIS as a browsing system on a typical retrieval task: subject searching. Recent results (Godin et al., 1993) show that an information retrieval method similar to our approach, but without background knowledge, performs favourably on this task. In particular, Godin et al. show that its performance is comparable to that of more conventional information retrieval methods such as hierarchical browsing and Boolean querying, at least for small databases. In our experiment we evaluate how the retrieval effectiveness of this method changes when it is enhanced with a thesaurus component.

We conducted an experiment on a collection of 1555 documents, of which the documents in Table 1 represent a tiny sample extracted from INSPEC, a commonly used large computer-engineering collection. We queried INSPEC by questioning “artificial intelligence”, and selected the 1555 most recent (as of January 1994) elements out of some 10,000 documents retrieved. We chose this size because it is large enough for the test to be considered significant, and small enough for the relative lattice to be computed, stored, and accessed easily. The documents were described by a title, an abstract, and a set of terms. In order to deal with a controlled and compact vocabulary, we used only the terms labelled as *preferred*. With this choice, there were 926 distinct keywords, with an average of 5.10 terms per document. The corresponding concept lattice consisted of 3763 nodes with an average of 2.88 parents per node and path lengths ranging from 3 to 11 edges from the lattice’s top to bottom node. This was the lattice used to test browsing retrieval without knowledge. By contrast, building the lattice to test browsing retrieval with knowledge required some more pre-processing. It was first necessary to encode the background knowledge. From the broader/narrower relation among preferred terms given in the 1991 INSPEC thesaurus, we extracted the sub-graph containing all terms equal to or broader than the 926 terms used to describe the documents. The resulting graph, of which figure 5 represents a small portion, contained 1136 terms. With this enlarged set of keywords each document was described by 10.15 terms on average. The generalised lattice contained 8769 nodes, with an average of 3.11 parents per node and a depth ranging from 2 to 15 edges.

The experimental protocol was as follows. Retrieval using the generalized and ungeneralized lattice was evaluated on 20 queries by three users. As there is no standard methodology to determine a representative set of queries, we tried to mix different types of queries, such as queries containing general and specific subjects, known and unknown words, conjunctive and disjunctive concepts. The queries are shown in Appendix C. For each query we manually produced its relevance judgements, i.e., the associated set of relevant documents. The average number of relevant documents for the 20 queries was 30.5. The three users were researchers at our institute; they had a computer science background, but had little knowledge of the document domain and did not know the system. We provided them with a tutorial session on a small training database. For assigning queries to the two retrieval methods (i.e., generalized and ungeneralized lattice) we used a repeated-measures design, in which each user searched each query using each method. To minimize sequence effects (Tague-Sutcliffe, 1992), we varied the order of the two methods over the query set, and, in addition, we introduced a time delay between the test of one method and the test of the other method. In practice, the set of queries was randomly split into two subsets of 10 queries, and each user searched one subset using generalized lattice and the other subset using ungeneralized lattice. After a week, each user searched each subset using the alternative method. During each search the user, who was not asked to finish within a certain time period, could see the abstracts of the documents associated with the visited nodes. The documents judged to be relevant by the user, as well as those scanned during the search, were noted as retrieved[10]. For each search we considered four measures: recall, precision[11], number of nodes visited, and search time (i.e., the time taken by the user to perform his task). The results are displayed in Table 3.

The table shows that searching with knowledge (i.e., a generalized lattice) obtained better evaluation scores for each measure, except for precision. To see if these differences can be considered statistically significant (Hull, 1993) we performed a paired t-test for each measure. The test revealed no effect of the method on precision ($p = 0.590$), and it did not reveal a significant difference in number of nodes visited ($p = 0.241$). However, it did reveal the superiority of the method with knowledge with respect to recall ($p = 0.020$) and, to a smaller extent, with respect to search time ($p = 0.055$). Thus, these results provide

Table 3. Average values of retrieval performance measures.

Method	Recall	Precision	Number of nodes visited	Search time (sec)
Browsing without knowlege	0.494 ($\sigma = 0.163$)	0.489 ($\sigma = 0.115$)	22.4 ($\sigma = 12.2$)	2415 ($\sigma = 846$)
Browsing with knowledge	0.647 ($\sigma = 0.212$)	0.453 ($\sigma = 0.159$)	19.8 ($\sigma = 13.4$)	1968 ($\sigma = 396$)

further evidence (e.g. (Furnas, 1985; Yu et al., 1989)) that the integration of the information contained in a thesaurus can, especially for cluster-based or interactive retrieval methods, improve recall without sacrificing precision. In addition, the results seem to suggest that the use of background knowledge may reduce user search time without reducing retrieval effectiveness. In our experiment this phenomenon can be explained in the following way. In an ungeneralised lattice the proximity of two nodes is independent of the meaning of the terms describing the nodes, so that two semantically similar queries are often very distant within the lattice. In fact, we observed that in the method without knowledge, the browsing was interrupted more often, compared to the method with knowledge. This meant a more frequent use of the query interaction mode, entailing a time consuming scanning of the vocabulary window. In contrast, the method with knowledge made it easier for the user to find the nodes containing semantically related queries, in that the region to search is usually a small subgraph of the whole lattice; furthermore, the system can be queried fast and more effectively by specialising/generalizing some of the terms in the current node, which is not allowed in the method without knowledge.

6. Related work on clustering and browsing

This research is closely related to work done in three areas: incremental hierarchical conceptual clustering, automatic determination of concept lattices, and browsing retrieval. In this section we examine the relation to each of them in turn.

Most incremental hierarchical conceptual clustering systems (Lebowitz, 1986; Fisher, 1987; Hanson & Bauer, 1987; Hadzikadic & Yun 1989; McKusick & Langley, 1991; Martin & Billman, 1994) rely on hierarchy modification operators that perform local changes to the current hierarchy in order to accommodate new objects, where the operator application is usually driven by some heuristic evaluation criterion. Therefore, the result is crucially biased by the choice of operators and evaluation criteria. In fact, the main problem with this approach seems to be the lack of a clear semantics by which to characterize the *whole* hierarchy in terms of the object description. It is easy to see (Carpineto & Romano, 1993a) that even for a very simple set of objects, a system like COBWEB (Fisher, 1987) may produce certain classes while failing to produce other equally good classes; furthermore, the classes produced may change when the objects are processed in a different order.

By contrast, a concept lattice represents a virtually unbiased set of clusters, as it contains all *complete* classes of objects having a conjunctive description. The fact that a concept lattice enjoys this kind of completeness property, and that it is not sensitive to object ordering, helps expand the range of its applications beyond pure predictive tasks (Carpineto & Romano, 1993a). Browsing retrieval is one case in point; another is discovery of implication rules between database attribute values (Ganascia, 1987; Oosthuizen & McGregor, 1988; Godin et al., 1991), which is relevant to work on database discovery (for instance, see Piatetsky-Shapiro & Frawley (1992)). Another feature of concept lattices is that they can generate overlapping classes. Although this is not a new idea (e.g. (Levinson, 1984; Lebowitz, 1986)) the importance of this issue has been somewhat overlooked. On one hand, very little work has been done on acquiring formally characterizable classes of partially ordered sets of clusters; on the other hand, although learning overlapping classes may be an important practical requirement even for pure predictive tasks (Martin & Billman, 1994), data sets with non-disjoint classes have been surprisingly rare in the machine learning literature. Finally, GALOIS can easily accomodate background knowledge, while few other clustering systems (e.g., Thompson et al., 1991; Stepp & Michalski, 1986) have this capability.

These are clear advantages of GALOIS over other incremental hierarchical conceptual clustering systems, but there are other aspects in which the system compares less favourably. One of the disadvantages is its relative cost, especially when, as for the Voting Records data set, the number of concepts grows quadratically with the number of objects. Another difference concerns the concept description language. GALOIS' concepts are conjunctive descriptions defined on structured descriptors, while other systems employ different formalisms, such as probabilistic concepts (Fisher, 1987; McKusick & Langley, 1991; Martin & Billman, 1994), contingency tables (Hanson & Bauer, 1989), and prototype schemata (Hadzikadic & Yun, 1989). In this respect, the latter formalisms are more flexible in that their hypothesis space is larger than GALOIS', but we should note that the kind of queries GALOIS can handle can be easily extended (Carpineto & Romano, 1994) to incorporate more powerful operators, including the Boolean operator NOT.

Among the few learning approaches that advocate the use of a lattice structure similar to that of GALOIS, Ganascia's CHARADE (Ganascia, 1987) does not construct the lattice physically, and Oosthuizen and McGregors' approach (Oosthuizen & McGregor, 1988) focuses on the utilization of the lattice for rule-base normalisation, rather than on its construction. To our knowledge, the only algorithm for incremental determination of concept lattices other than GALOIS is reported by Godin et al. (1991). Like GALOIS, their system iterates on every node in the lattice; however, its updating strategy is substantially different, in that it is based on partitioning the concepts according to the cardinality of their intents and processing the obtained partitions in ascending cardinality order. Although the asymptotic behavior of their method's time complexity is the same as GALOIS (i.e., constant for each update), we expect that for practical applications Godin et al.'s system will be less efficient. To compare the two systems[12], it is convenient to consider their time complexity with respect to the number of concepts in the lattice to be updated (which is of course the same for both algorithms). For each iteration, assuming the database is described by m v-valued attributes, Godin et al.'s system must explore one of the m partitions P_i, which may be as large as $\max\|P_i\| = \max_i \binom{m}{i} v^{m-i} > v^m$, to check concept duplication; furthemore, at the end of this inner iteration it may have to explore a large part of the lattice to properly set the edges. Therefore, we expect that the time complexity of each update may grow nearly quadratically in the number of concepts in the lattice. By contrast, GALOIS will apply the function LINK, which requires exploring a small subset of $2v^m$ concepts, only for a small fraction of iterations (see Section 3.3). In fact, we have seen that the time complexity of GALOIS is in practice linear in the number of concepts in the lattice.

Aside from complexity issues, the algorithm presented in Godin et al. (1991) is more limited in scope than GALOIS, in that it deals with unstructured attributes and cannot be easily extended to accomodate structured attributes; in fact, its reliance on intent cardinality would seem to prevent it from incorporating this kind of knowledge (see Section 4.1).

In most information retrieval systems, browsing is seen either as an aid for conventional query-based retrieval or as an alternative search method. The first approach is described in

Frei & Jauslin (1983) and Pedersen (1993), where a search through a *term* space is used to improve query formulation in a distinct *document* space. An example of the second approach is Crouch et al. (1989), which describes an interactive browser supported by a cluster hierarchy, or Maarek et al. (1991), where a similar hierarchical browsing system supplements a Boolean query system. In these systems, the network that supports browsing, whether manually built, as in Frei & Jauslin (1983), and Pedersen (1993), or automatically built, as in Crouch et al. (1989), and Maarek et al. (1991), is usually developed and maintained as a distinct component. The main drawback of these retrieval architectures is that the user must map different concept representations and data spaces; in addition, the system has to maintain different support structures. GALOIS, on the other hand, integrates browsing and querying into a single term/document space. This has the advantage that the user may combine term-based query refinement with direct inspection of the document database. Among the hybrid approaches to browsing retrieval that have been proposed, the most similar to ours is Godin et al. (1989) and (1993), which is also based on a Galois document lattice. The major difference from GALOIS is the inability of Godin et al.'s system to use thesaurus knowledge, which results in a weaker expressive power and in more limited browsing facilities. In particular, as pointed out in Section 5.3, because an ungeneralised lattice is an inherently syntactic structure, it does not let the user track the semantic links between terms.

7. Conclusion and future work

This paper was roughly split into two parts. In the first, we presented GALOIS as a conceptual clustering system based on the theory of concept lattices. We have seen that a concept lattice represents a radical departure from most hierarchical conceptual clustering structures that can help overcome some of their well-known limitations. GALOIS implements a new incremental algorithm for determining the concept lattice corresponding to a set of objects. An experimental evaluation conducted on different types of data sets suggests that the time complexity of each update of the algorithm ranges from linear to quadratic in the number of objects to be clustered. The experiments also provide supporting evidence for a theoretical analysis that shows that the number of concepts in the lattice usually does not exceed the square of the number of objects to be clustered. These results in part confirm and in part extend earlier results found by Godin et al. (1991). In addition, the algorithm can use background knowledge to expand and order the set of properties describing the objects, thereby extending the scope of the theory.

In the second part of the paper, we took the view that conceptual facilitates browsing retrieval. In particular, we discussed the merits of the kind of lattice structure produced by GALOIS for supporting this task. To summarize, the lattice is built automatically from the usual document-term relation, it may use the information contained in a thesaurus of terms, it has its own indexing and linking structures, it provides multiple access to each class of documents, it supports both browsing retrieval and query retrieval, it allows minimal query refinement or enlargement, and it has limited storage requirements (proportional to the number of documents). In order for this structure to be used for actual information retrieval, it is necessary to have a user interface for browsing and querying a lattice of documents enriched with thesaurus knowledge. We described the design and implementation of a prototype interface of this kind, and we evaluated its retrieval effectiveness on three subjects searching a bibliographic database of reasonable size (1555 document abstracts, and 7929 index terms), including a thesaurus of terms. In particular, we evaluated the relative performance of the system with knowledge over the system without knowledge. Similar to the results obtained by Godin et al. (1993) on a much smaller database, our results show a good absolute retrieval performance; they also suggest that the integration of a thesaurus component into the system improves recall as well as search time.

This research can be extended in several directions. A basic improvement, for which a substantial amount of work has already been done (Carpineto & Romano, 1994), is the realization of a pruning mechanism whereby the user may dynamically bound the lattice from which he or she is retrieving information based on the feedback he or she gets from the structure. Another limitation of this approach concerns the visualisation of the lattice. As seen in Section 6, there are many ways in which the current display technique of GALOIS can be improved to provide better orientation for the user. The third planned research direction is investigation of the potential of language factorization for this approach. The idea is to work with a concept lattice in which the set of index terms may be restricted or enlarged by the user at browsing time, based on the user goal and other possible search constraints. We believe that this could be used to control the complexity of the lattice construction, as well as the complexity of the interaction between the user and the system.

Appendix A. Correctness of the algorithm

We will give the proof of the correctness of the algorithm for updating a concept lattice and its Hasse diagram with a new object. In our notation O stands for the (intent of the) new object, C for the (intent of the) current old concept examinated by the algorithm, N for the (intent of the) new concept given by the intersection of C and O (i.e., $N = C \cap O$).

Observation 1. *The lattice contains all possible intersections of the objects.*

A corollary of Observation 1 is that N is always a node of the updated lattice.

Lemma 1. *The algorithm generates all concepts of the updated lattice.*

Proof: The set of concepts of the updated lattice can be obtained by the union of the concepts in the initial lattice (old concepts) and the intersections between the new object and every old concept (new concepts). This is the way the algorithm works, except when (see test d in Table 2) there is a parent (F) of C such that $F < N$, in which case the algorithm does not add N to the lattice. However, we show that there exists some other old concept which returns the same intersection (i.e., $C \cap O$). By $F < N = C \cap O$ we derive $F \cap O \leq (C \cap O) \cap O = C \cap O$; by $F > C$ we derive $F \cap O \geq C \cap O$. Thus, $F \cap O = C \cap O$. F may be either an old concept or a new concept. If F is old it is the desired concept. If F is new then there is some old concept $\underline{C}$ such that $F = \underline{C} \cap O$; thus

$N = F \cap O = (\underline{C} \cap O) \cap O = \underline{C} \cap O$. Therefore, in the latter case the desired concept is $\underline{C}$. □

Lemma 2. *The updated lattice generated by the algorithm only contains distinct concepts.*

Proof: Since the initial lattice contains only distinct concepts, we must only show that the update does not introduce duplicates. Assuming there is some concept $\underline{C}$ in the lattice such that $\underline{C} = N = C \cap O \geq C$. If $\underline{C} = N = C$ then the generation of a duplicate is ruled out by test b in Table 2. If $\underline{C} = N > C$ then there is a parent F of C (i.e., $\underline{C}$ or one of its descendants) such that $F \leq N$; thus concept duplication is avoided by tests c and d. □

Lemma 3. *The updated lattice generated by the algorithm is a Hasse diagram.*

Proof: We prove the lemma by induction over the iteration steps of the algorithm. The initial lattice is a Hasse diagram by hypothesis; assume the lattice at step i is correct and consider step $i+1$. If no new concept is added to the lattice then the lattice remains correct, otherwise a new concept (N) is added to the lattice, which causes some edges to be created and some edges to be removed. The former edges are only concerned with the new concept; by definition, they are set correctly by procedure LINK, which links the new concept to all of and only its maximally specific more general concepts and its maximally general more specific concepts. For the latter edges, since LINK removes all edges between the lower and the upper boundary sets, we must prove that if an edge between two concepts C_1 and C_2 has to be removed (i.e., $C_1 < N < C_2$) then $C_1(C_2)$ does belong to the lower (upper) set. Indeed, if C_1 did not belong to the lower set then there would exist a concept $\underline{C}$ such that $C_1 < \underline{C} < N < C_2$; but this would imply that the lattice at step i was incorrect. □

Theorem 1. *The algorithm for updating a concept lattice and its Hasse diagram is correct.*

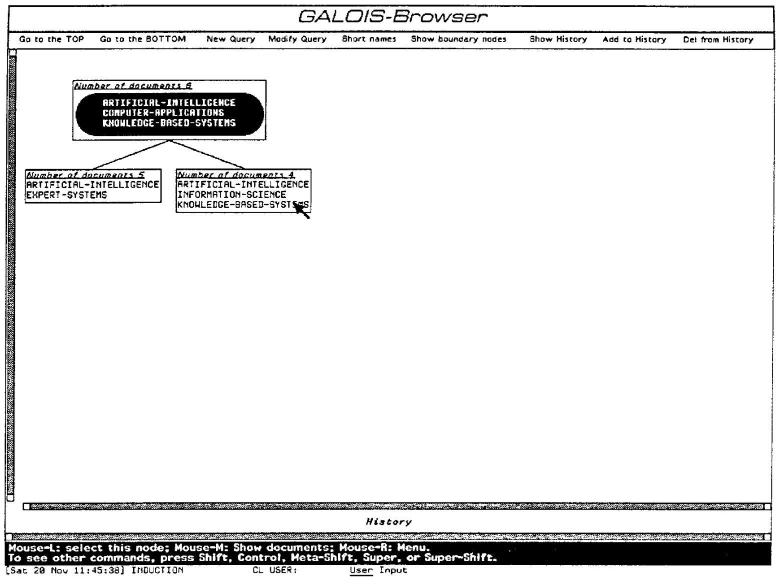
Proof: From Observation 1 and Lemma 1 it follows that the set of nodes generated by the algorithm contains all of and only the elements of the concept lattice. By Lemma 2, the algorithm generates no duplicates. By Lemma 3, the graph is a Hasse diagram. □

Appendix B. A simple browsing session

The session is relative to the lattice in figure 7. According to the visualization technique employed by GALOIS-Browser, each figure represents a screen displaying the current node (i.e., the black node) along with its parents and children. Simultaneously, the figure shows the action taken by the user to modify the current node, which results in the screen shown in the next figure.

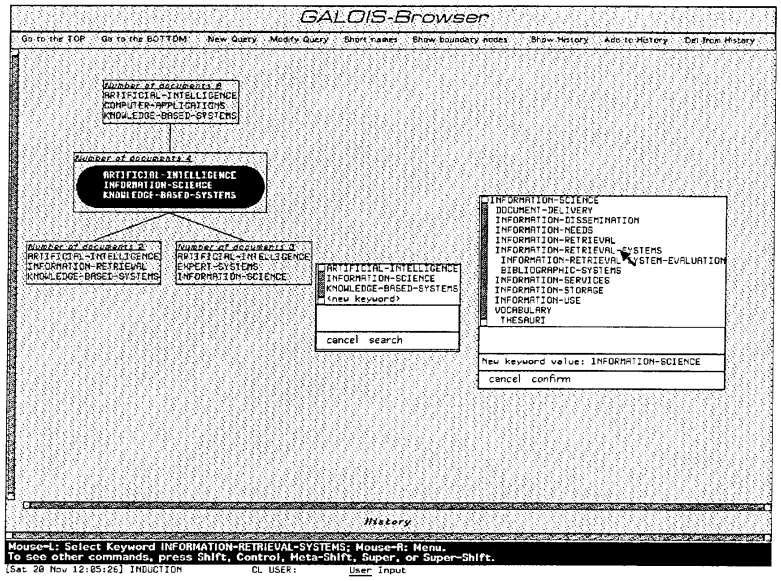
Appendix C. Experimentation query set

1. Rule induction in artificial intelligence systems.
2. Applications of speech recognition to telecommunications environments.



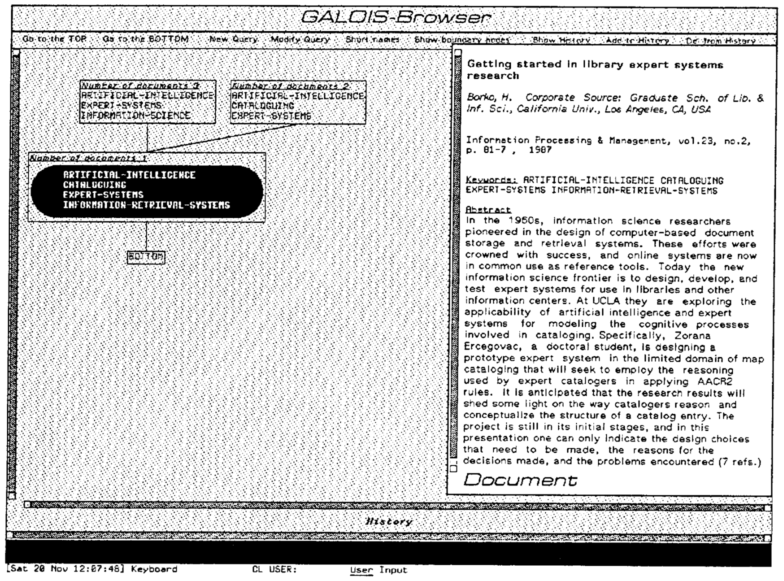
```
Number of documents 6
ARTIFICIAL-INTELLIGENCE
COMPUTER-APPLICATIONS
KNOWLEDGE-BASED-SYSTEMS

Number of documents 5
ARTIFICIAL-INTELLIGENCE
EXPERT-SYSTEMS

Number of documents 4
ARTIFICIAL-INTELLIGENCE
INFORMATION-SCIENCE
KNOWLEDGE-BASED-SYSTEMS

Mouse-L: select this node; Mouse-M: Show documents; Mouse-R: Menu.
To see other commands, press Shift, Control, Meta-Shift, Super, or Super-Shift.
[Sat 20 Nov 11:45:38] INDUCTION          CL USER:          User Input
```

Figure 8. Initial screen displaying the most general query (ARTIFICIAL-INTELLIGENCE, COMPUTER-APPLICATIONS, KNOWLEDGE-BASED-SYSTEMS) as current query. The user selects the more specific query ARTIFICIAL-INTELLIGENCE, INFORMATION-SCIENCE, KNOWLEDGE-BASED-SYSTEMS by pointing and clicking with the mouse on the right child of the current node.



```
Number of documents 6
ARTIFICIAL-INTELLIGENCE
COMPUTER-APPLICATIONS
KNOWLEDGE-BASED-SYSTEMS

Number of documents 4
ARTIFICIAL-INTELLIGENCE
INFORMATION-SCIENCE
KNOWLEDGE-BASED-SYSTEMS

Number of documents 2
ARTIFICIAL-INTELLIGENCE
INFORMATION-RETRIEVAL
KNOWLEDGE-BASED-SYSTEMS

Number of documents 3
ARTIFICIAL-INTELLIGENCE
EXPERT-SYSTEMS
INFORMATION-SCIENCE

ARTIFICIAL-INTELLIGENCE
INFORMATION-SCIENCE
KNOWLEDGE-BASED-SYSTEMS
<new keyword>

INFORMATION-SCIENCE
 DOCUMENT-DELIVERY
 INFORMATION-DISSEMINATION
 INFORMATION-NEEDS
 INFORMATION-RETRIEVAL
 INFORMATION-RETRIEVAL-SYSTEMS
  INFORMATION-RETRIEVAL-SYSTEM-EVALUATION
  BIBLIOGRAPHIC-SYSTEMS
 INFORMATION-SERVICES
 INFORMATION-STORAGE
 INFORMATION-USE
 VOCABULARY
  THESAURI

New keyword value: INFORMATION-SCIENCE

Mouse-L: Select Keyword INFORMATION-RETRIEVAL-SYSTEMS; Mouse-R: Menu.
To see other commands, press Shift, Control, Meta-Shift, Super, or Super-Shift.
[Sat 20 Nov 12:05:26] INDUCTION          CL USER:          User Input
```

Figure 9. The user modifies the current query specialising the term INFORMATION-SCIENCE by the term INFORMATION-RETRIEVAL-SYSTEMS.



Getting started in library expert systems research

Borko, H. Corporate Source: Graduate Sch. of Lib. & Inf. Sci., California Univ., Los Angeles, CA, USA

Information Processing & Management, vol.23, no.2, p. 81-7 , 1987

Keywords: ARTIFICIAL-INTELLIGENCE CATALOGUING EXPERT-SYSTEMS INFORMATION-RETRIEVAL-SYSTEMS

Abstract
In the 1960s, information science researchers pioneered in the design of computer-based document storage and retrieval systems. These efforts were crowned with success, and online systems are now in common use as reference tools. Today the new information science frontier is to design, develop, and test expert systems for use in libraries and other information centers. At UCLA they are exploring the applicability of artificial intelligence and expert systems for modeling the cognitive processes involved in cataloging. Specifically, Zorana Ercegovac, a doctoral student, is designing a prototype expert system in the limited domain of map cataloging that will seek to employ the reasoning used by expert catalogers in applying AACR2 rules. It is anticipated that the research results will shed some light on the way catalogers reason and conceptualize the structure of a catalog entry. The project is still in its initial stages, and in this presentation one can only indicate the design choices that need to be made, the reasons for the decisions made, and the problems encountered (7 refs.)

Figure 10. Since the query ARTIFICIAL-INTELLIGENCE, KNOWLEDGE-BASED-SYSTEMS, INFORMATION-RETRIEVAL-SYSTEMS is not in the lattice, the system returns its most general specialization (ARTIFICIAL-INTELLIGENCE, CATALOGUING, EXPERT-SYSTEMS, INFORMATION-RETRIEVAL-SYSTEMS). The figure shows also the document associated with it (i.e., the fifth document of the example context).

3. Effect of the advances in computer technology and artificial intelligence on information services.
4. Heuristic search: methods and applications.
5. Role played by artificial intelligence in advancing the field of robotics.
6. Criteria for the objective evaluation of knowledge-based systems.
7. Picture processing systems not based on neural network techniques
8. Effectiveness of inclusion of natural language processing techniques in information retrieval interfaces.
9. Methods for mapping 3D objects on 2D representations.
10. State of the art of automatic medical diagnosis.
11. Parsers for languages other than English.
12. Expert systems for designing or monitoring nuclear plants.
13. Automated construction of expert systems through machine learning techniques.
14. Artificial intelligence programming languages other than Lisp or Prolog.
15. Applications of machine learning technology to real world problems.
16. Image segmentation in picture analysis.
17. Integrated methods for combining symbolic and neural processing.
18. The cost and value of expert systems.
19. Unsupervised learning: methods and research issues.
20. Use of parallelism to improve efficiency of artificial intelligence languages or algorithms

Acknowledgments

We would like to thank Doug Fisher and three anonymous reviewers for many useful comments and suggestions. We would also like to thank Giacinto Matarazzo for help with the statistical tests and Vittorio Giannini for his support for the experiments on a SPARCstation LX. This work was carried out under the auspices of an agreement between the Italian PT Administration and the Fondazione Ugo Bordoni.

Notes

1. GALOIS has been implemented in Common Lisp.
2. Although this representation is very natural for unstructured domains, such as documents described by keywords in information retrieval, it can also be easily adapted to more structured domains, such as records described by a fixed number of attribute-value pairs in databases. In the latter case, each attribute is assigned a set of bits (one for each value of the attribute), no more than one of which will be on.
3. Y is a kind of maximally specific description of the objects in X (Michalski, 1983).
4. Recall that, given a non-empty ordered set P, if for all $S \subset P$ there exists a least upper bound and a greatest lower bound, then P is called a *complete lattice*.
5. In practice, it is sufficient to determine either of the two sets, and then derive the other from it (the LINK procedure determines the lower boundary set first). Furthermore, the two boundary sets are determined by considering only a subset of the whole current concept lattice; in fact, it can be easily seen that, as the new node is the intersection of the new object with an old concept, the concepts to which it may be linked must be more general than the new object or more general than the old concept.
6. This inference follows from the observation that, in figure 3, the growth of the ratio between the number of concepts and the number of objects varies from constant to linear with respect to the number of objects.
7. In practice GALOIS employs the generalised intersection operator ($\cap^*$) to assess the more-general-than relation (4): $(X_1, Y_1) < (X_2, Y_2) \leftrightarrow (X_1, Y_1) \cap^* (X_2, Y_2) = (X_2, Y_2), (X_1, Y_1) \neq (X_2, Y_2)$.
8. GALOIS-Browser has been implemented in CLIM on a Symbolics Lisp Machine.
9. Although this is a simple and widespread solution to graph visualisation, it has the disadvantage that the user may get disoriented because of the lack of global information; a more sophisticated approach would use techniques to produce concise visual representations of large graphs, such as a generalised fisheye view (Furnas, 1986) or a graphical compression of Hasse diagrams (Wille, 1984).
10. In evaluating the effectiveness of browsing retrieval systems, the definition of the retrieved set of documents is usually not obvious. One typical choice (Tague-Sutcliffe, 1992) is to rate a document as a retrieved document when its full description (the abstract, in our case) is recovered.
11. *Recall* is defined as the ratio of number of items retrieved and relevant to the number of items relevant; *precision* is the ratio of number of items retrieved and relevant to the number of items retrieved. Recall measures the ability to retrieve *all* relevant documents, while precision measures the ability to retrieve only relevant documents.
12. A direct experimental comparison between GALOIS and Godin et al.'s system is difficult because Godin et al. used different data sets in their experiments, and, as also mentioned in Section 3.3, they employed an alternative implementation strategy.

References

Anderson, J., & Matessa, M. (1992). Exploration of an incremental Bayesian algorithm for categorization. *Machine Learning*, 9:275–308.

Carpineto, C., & Romano, G. (1993a). GALOIS: An order-theoretic approach to conceptual clustering. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 33–40), Amherst, MA: Morgan Kaufmann.

Carpineto, C., & Romano, G. (1993b). Adding background knowledge to a Galois lattice classification structure (Technical Report 5T02093). Rome, Italy: Fondazione Ugo Bordoni.

Carpineto, C., & Romano, G. (1994). Dynamically bounding browsable retrieval spaces: an application to Galois lattices. *Proceedings of RIAO 94: Intelligent Multimedia Information Retrieval Systems and Management* (pp. 520–533). New York, N.Y.: Jouve.

Crouch, D., Crouch, C., & Andreas, G. (1989). The use of cluster hierarchies in hypertext information retrieval. *Proceedings of the ACM Hypertext '89 Conference* (pp. 225–237). Pittsburgh, PA: ACM.

Cutting, D., Karger, D., Pedersen, J., & Tukey, J. (1992). Scatter/Gather: A cluster-based approach to browsing large document collections. *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 318–329). New York, N.Y.: ACM SIGIR Forum.

Davey, B., & Priestley, H. (1990). *Introduction to Lattices and Order*, Cambridge University Press, Cambridge, Great Britain.

Fisher, D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172.

Frei, H., & Jauslin, J. (1983). Graphical presentation of information and services: a user oriented interface. *Information Technology: Research and Development*, 2:23–42.

Furnas, G. (1985). Experience with an adaptive indexing scheme. *Proceedings of ACM CHI'85 Conference on Human Factors in Computings Systems* (pp. 130–135). San Francisco, CA: ACM.

Furnas, G. (1986). Generalized fisheye views. *Proceedings of ACM CHI'86 Conference on Human Factors in Computing Systems* (pp. 16–23). Boston, MA: ACM.

Ganascia, J.-G. (1987). CHARADE: A rule system learning system. *Proceedings of the Tenth International Conference on Artificial Intelligence* (pp. 345–347). Milan, Italy: Morgan Kaufmann.

Ganter, B. (1988). Composition and decomposition of data in formal concept analysis. In H. Bock (Ed.), *Classification and Related Methods of Data Analysis*. North-Holland.

Gennari, J., Langley, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, 40:12–61.

Godin, R., Gecsei, J., & Pichet, C. (1989). Design of a browsing interface for information retrieval. *Proceedings of the Twelfth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 32–39). Cambridge, MA: ACM SIGIR Forum.

Godin, R., Missaoui, R., & Alaoui, H. (1991). Learning algorithms using a Galois lattice structure. *Proceedings of the 1991 IEEE International Conference on Tools for AI* (pp. 22–29). San Jose, CA: IEEE Computer Society Press.

Godin, R., Missaoui, R., & April, A. (1993). Experimental comparison of navigation in a Galois lattice with conventional information retrieval methods. *International Journal of Man-machine Studies*, 38:747–767.

Guénoche, A. (1990). Construction du treillis de Galois d'une relation binaire. *Mathḿatiques et Sciences Humaines*, 109:41–53.

Hadzikadic, M., & Yun, D. (1989). Concept formation by incremental conceptual clustering. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 831–836). Detroit, MI: Morgan Kaufmann.

Hanson, S., & Bauer, M. (1989). Conceptual clustering, categorization, and polymorphy. *Machine Learning*, 3:343–372.

Hull, D. (1993). Using statistical testing in the evaluation of retrieval experiments. *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 329–338). Pittsburgh, PA: ACM SIGIR Forum, 1993.

Lebowitz, M. (1986). Concept learning in a rich input domain: generalization-based memory. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Vol. 2). Morgan Kaufmann, San Mateo, CA.

Levinson, R. (1984). A self-organizing retrieval system for graphs. *Proceedings of the Fourth National Conference on Artificial Intelligence* (pp. 203–206). Austin, TX: Morgan Kaufmann.

Maarek, Y., Berry, D., & Kaiser, G. (1991). An information retrieval approach for automatically constructing software libraries. *IEEE Transactions on Software Engineering*, 17(8):800–813.

Marchionini, G., & Shneiderman, B. (1988). Finding facts vs. browsing knowledge in hypertext systems. *IEEE Computer*, 21:70–80.

Martin, J., & Billman, D. (1994). Acquiring and combining overlapping concepts. *Machine Learning*, 16(1–2):121–155.

McKusick, K., & Langley, P. (1991). Constraints on tree structure in concept formation. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (pp. 810–816). Sidney, Australia: Morgan Kaufmann.

Michalski, R.S. (1983). A theory and methodology of inductive learning. *Artificial Intelligence*, 20:111–161.

Michalski, R.S., Stepp, R. (1983). Learning from observation: conceptual clustering. In R.S. Michalski, J.G. Carbonell, & T. M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Vol. 1). Palo Alto, CA: Tioga Publishing.

Mitchell, T.M. (1982). Generalization as search. *Artificial Intelligence*, 18:203–226.

Nielsen, J. (1990). *Hypertext & Hypermedia*, Academic Press, San Diego, CA.

Oosthuizen, G., & McGregor, D. (1988). Induction through knowledge base normalisation. In *Proceedings of the Eight European Conference on Artificial Intelligence* (pp. 396–401). Munich, Germany: Pitman.

Pedersen, G. (1993). A browser for bibliographic information retrieval based on an application of lattice theory. *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 270–279). Pittsburgh, PA.

Piatetsky-Shapiro, G., & Frawley, W. (Eds.) (1992). *Knowledge Discovery in Databases*, AAAI Press, Palo Alto, CA.

Salton, G., & McGill, M. (1983). *Introduction to Modern Information Retrieval*, McGraw Hill, New York, N.Y.

Shneiderman, B. (1987). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Addison Wesley, Reading, MA.

Stepp, R., & Michalski, R.S. (1986). Conceptual clustering: inventing goal-oriented classifications of structured objects. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Vol. 2). Morgan Kaufmann, San Mateo, CA.

Tague-Sutcliffe, J. (1992). The pragmatics of information retrieval experimentation, revisited. *Information Processing & Management*, 28(4):467–490.

Thompson, K., Langley, P., & Iba, W. (1991). Using background knowledge in concept formation. *Proceedings of the Eight International Workshop on Machine Learning* (pp. 554–558). Evanston, IL: Morgan Kaufmann.

Thompson, R., & Croft, B. (1989). Support for browsing in an intelligent text retrieval system. *International Journal of Man-machine Studies*, 30:639–668.

van Rijsbergen, J., & Croft, B. (1975). Document clustering: An evaluation of some experiments with the Cranfield 1400 collection. *Information Processing & Management*, 11:171–182.

Wille, R. (1984). Line diagrams of hierarchical concept systems. *International Classification*, 2:77–86.

Willet, P. (1988). Recent trends in hierarchic document clustering: A critical review. *Information Processing & Management*, 24(5):577–597.

Yu, C., Meng, W., & Park, S. (1989). A framework for effective retrieval. *ACM Transactions on Database Systems*, 14(2):147–167.

Received December 9, 1993
Accepted October 9, 1995
Final Manuscript October 26, 1995