5 Lattice-Based Retrieval Systems

*Have in mind the physical methods and mechanisms used
to instrument models.
(Calvin N. Mooers)*

This chapter describes the application of lattices in retrieval systems
(Mooers, FaIR, BR-Explorer, Rajapakse-Denham, FooCA). The use of lat-
tices for visualization or navigation is not considered, nor are programming
and implementation issues dealt with, as these fall outside our scope.

The goal of describing these systems is to present the way in which lat-
tices have been used to represent document structures, term relationships,
and term-document matrices in actual retrieval systems developed thus far.

Mathematical properties (with proofs) of the lattices applied are estab-
lished. The principal finding is that they are not modular.

Further, a method is given to transform a term-document matrix into a
Galois (concept) lattice.

The chapter ends with exercises and problems that are designed to en-
hance understanding of the properties of the lattices applied.

5.1 Mooers' Model

Mooers (1959) seems to have been the first individual to offer a detailed and comprehensive treatment of the application of the lattice concept in IR. His model has the merit of being able to capture relationships between terms (i.e., to formally express the fact that words are not necessarily independent of each other, which is a widely accepted hypothesis in many retrieval methods today). The model focuses on what Mooers calls symbols (known today as terms) that together form a query and on the relationship between the query and a subset of documents that can be selected from a collection of documents.

5.1.1 Lattice of Documents

The document subsets can be formed from, e.g., a library collection (books, articles, etc.). If one denotes a document subset by A and the entire library collection by $L = \{D_1,\ldots,D_n\}$, then the document subsets $A \subseteq D$ form a Boolean algebra ($\wp(L), \cap, \cup, \backslash$) with respect to set intersection $\cap$, set union $\cup$, and set complement $\backslash$, where $\wp(L)$ denotes the powerset of L, i.e., the set of all subsets of L. In other words, the structure ($\wp(L), \cap, \cup, \backslash$) is a complemented and distributive lattice.

5.1.2 Lattice of Unstructured Queries

A query Q is conceived as consisting of one or several terms (in the latter case constructed from one-term queries). For example, the one-term query $Q = A$ is modeled as the following lattice: $\{\mathbf{0}, A\}$ (**Fig. 5.1**):
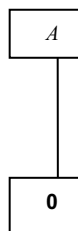

Fig. 5.1. One-term query $Q = A$ represented as a lattice $\{\mathbf{0}, A\}$, $\mathbf{0} \leq A$.

A new lattice P can be obtained by taking the product $\times$ of one-term lattices (**Fig. 5.2**):

1. Let A_i denote (or correspond to) the one-element lattice $\{\mathbf{0}, A_i\}$, $i = 1,\ldots,n$.

2. The product lattice $P = \underset{i=1}{\overset{n}{\times}} \{\mathbf{0}, A_i\}$ is given by the lattice $P=(\wp(T), \subseteq)$, where $T = \{A_1,\ldots,A_n\}$.
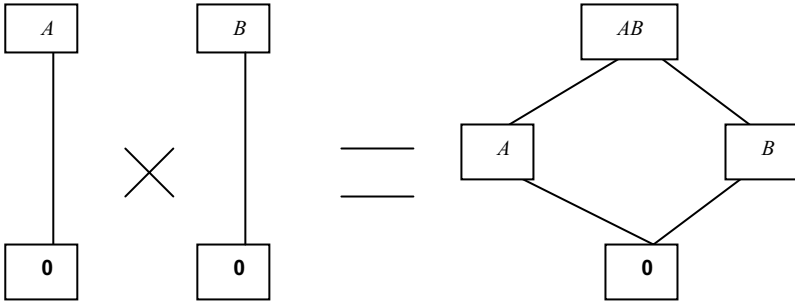

Fig. 5.2. Product P of two one-term lattices: $\{\mathbf{0}, A\}$ and $\{\mathbf{0}, B\}$.

The lattice P contains all the possible queries that can be formed using the given terms. It can be seen that the lattice P is a Boolean algebra and thus a complemented and distributive lattice.

Retrieval is formally viewed as follows. Given a query Q, i.e., an element of lattice P, there may be other elements in P preceded by Q, or elements that precede Q. Retrieval is some procedure that locates those elements of $\wp(T)$ that precede and are preceded by Q. However, the retrieval procedure is not described.

5.1.3 Lattice of Term Hierarchies

Mooers treats, very briefly, the case of term hierarchies. For example, the term "clothing" is broader in sense than the term "shoe." Thus, the following lattice (reflecting a hierarchy) can be constructed:
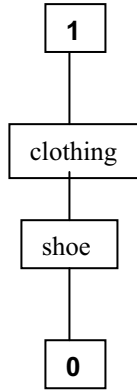
Taking into account hierarchical relations between terms should have an impact upon retrieval. In such lattices, some elements may be preceded by more than one element; this is referred to as a "system with weak hierarchy" (e.g., the U.S. Patent Office classification). After appropriate reductions, another lattice can be obtained from this one that does not allow for any element to be preceded by more than one element. Such a reduced lattice is referred to as a "system with strong hierarchy" (e.g., the Dewey Decimal classification).

5.1.4 Lattice of Boolean Queries and Documents

Mooers also outlines the case of Boolean (i.e., structured) queries (which he calls "characters with logic"). He starts by emphasizing that, "symbolic logic is a stylized view of things, and the symbolism or method which is found useful in that discipline need not necessarily be the most appropriate symbolism for information retrieval."

Given the terms $A_1,\ldots,A_n$, a Boolean query is conceived as being an element of a lattice $(L, \subseteq)$ obtained as follows:

1. The atoms are $A_1,\ldots,A_n$.
2. The elements greater than the atoms, and immediately above them, are given by the Cartesian product $\{A_1, \neg A_1\} \times \{A_2, \neg A_2\} \times \ldots \times \{A_n, \neg A_n\}$, where $\neg$ denotes negation.
3. The elements of point (2) are "topped" by the maximal element of the lattice.

If the query is a conjunction of terms (e.g., $A_1 \wedge A_2$), all terms are equally relevant to the subject matter of the document. When the query is a

disjunction of terms (e.g., $A_1 \vee A_2$), then—based on the interpretation of $\vee$ in mathematical logic—either one, or either two, and so on either all terms may be relevant to the subject matter of the document. "This is ridiculous," —Mooers says. He continues by asking: "For example, how good is a retrieval system that treats the query

$$red \vee square$$

as a logical expression?" Thus, he notes:

- Disjunction should not be a permissible operation in queries for retrieval; the only permissible operations should be conjunction and negation.
- On the other hand, a document should be represented as a lattice using only negation and disjunction of terms.

For example, if two terms are used, say A and B, then all possible documents (i.e., the 'space' whose element a document may be) are given by the lattice shown in **Fig. 5.3**.

The document lattice can be obtained as a product of two-element lattices. **Figure 5.4** shows the two 2-element lattices whose product lattice is shown in **Fig. 5.3**.
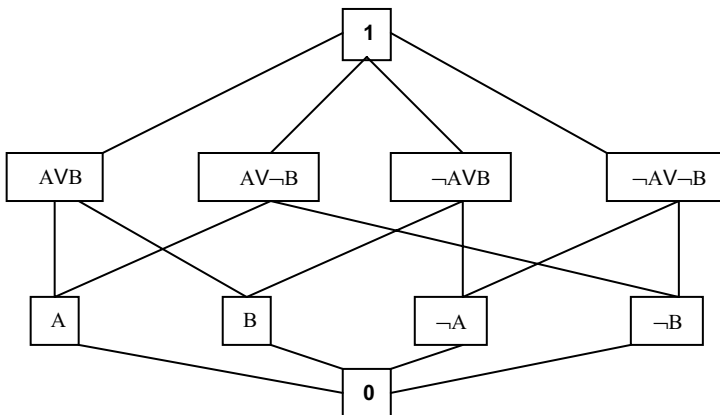

Fig. 5.3. Document lattice for Boolean queries using two terms A and B.
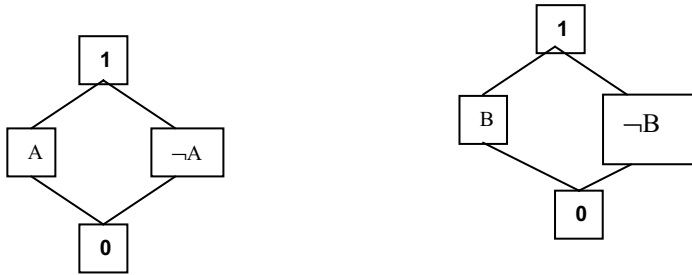
Fig. 5.4. The two 2-element lattices whose product is the lattice in Fig. 5.3.

The query lattice is similarly generated, but instead of disjunction we have conjunction. Mooers does not treat retrieval for such representations. At the same time, he makes a number of general and interesting observations:

- If a term A has not been used as an index term for a document, then the query "$\neg A$" should not retrieve that document merely because it does not contain A. In other words, in retrieval, absence is not necessarily synonymous with negation.
- It may be important to know the frequency with which terms are used. Thus, one can attach frequencies as "scalars" to lattice elements.
- In Boolean logic, the operations commute, e.g., $A \vee B$ is the same as $B \vee A$. If the information in a document is structured as a lattice, the ideas are commutative. But this is not always the case. The words that make up a term may form a sequence, but they do not always commute. For example, "street lamp" as a term may be modeled formally as the conjunction "street $\wedge$ lamp" but it is not equal, in general, to the commuted conjunction "lamp $\wedge$ street," as it should if taken as a Boolean expression of mathematical logic. In the term "street lamp," the words "street" and "lamp" do not form a hierarchy (and thus not a lattice); they form some other structure (e.g., a grammatical structure called a compound word).

5.2 The FaIR System

For a long time after Mooers's paper was published, the application of lattices in information retrieval was seen more as a theoretical possibility or curiosity. There were no further developments until Priss (2000) proposed a practical retrieval system, called FaIR, based on lattices, and thus showed that such a retrieval system could be built effectively.

The FaIR system uses domain knowledge (in the form of a thesaurus) to generate term lattices. The thesaurus consists of a set T of terms that is partitioned into classes (called facets). The facets are lattices. Every node in such a lattice represents a term (word or phrase), and the lattice expresses thesaurus relationships such as "broader than," "narrower than," etc. Every such lattice, i.e., facet, is conceptually complete (their terms express one concept).

Documents are represented by (or assigned to) as many terms as needed, but at most one term from one facet. Documents that contain only one of the terms of the facet are mapped to that concept. Documents that contain several terms of the facet are mapped to the join of the concepts. **Figure 5.5** illustrates an example of a facet lattice.
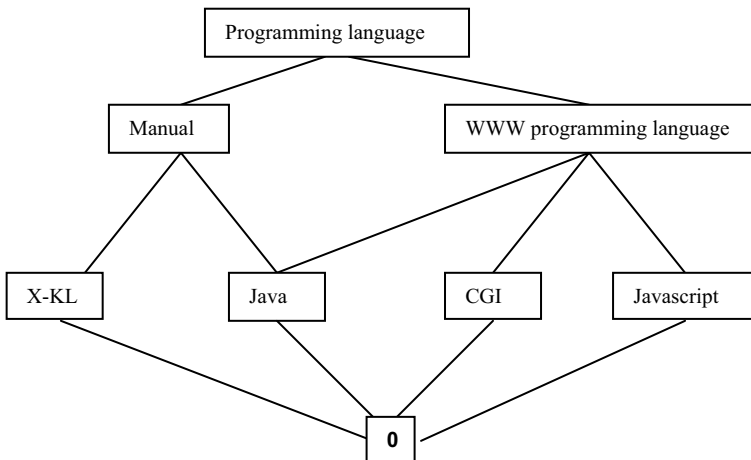

Fig. 5.5. Facet lattice "programming language." A document containing both CGI and Java is assigned to the node "WWW programming language," which is the join of these terms.

The elements of the facet lattice may be formally conceived as containing (i.e., being equal to the union of) the elements that are below it (equivalently, down to atoms). Every document is mapped to a single concept over all facets.

A query Q is a Boolean expression of terms. For example, the query $Q =$ "Java" retrieves the documents containing exactly and only the term "Java" (in the exclusive search) and the documents that also contain the more general term "WWW programming language" (in the inclusive search).

5.3 Galois (Concept) Lattice-Based Models

5.3.1 Galois (Concept) Lattice

Concepts are basic units of language and thought. The *extension G* of a concept consists of all objects that belong to it. The *intension M* of a concept consists of all attributes (properties) that apply to the elements of its extension. A formal context K is defined as a binary relation, i.e., as the set of relationships between objects and attributes to denote which object has a given property (Wolff 1993, Kim and Compton 2004, Wille 2005):

$$K = (G, M, I), I \subseteq G \times M. \qquad (5.1)$$

The following *derivation operations* are defined for arbitrary $X \subseteq G$ and $Y \subseteq M$:

$$X \rightarrow X^I = \{m \in M \,|\, g\,I\,m, \forall g \in X\}, \qquad (5.2)$$

i.e., the set of attributes common to all objects from X, and

$$Y \rightarrow Y^I = \{g \in G \,|\, g\,I\,m, \forall m \in Y\}, \qquad (5.3)$$

i.e., the set of objects described by at least one attribute from Y. A *formal concept* (A, B) is defined as

$$(A, B) \text{ is a formal concept} \Leftrightarrow$$
$$(A \subseteq G, B \subseteq M, A = B^I, B = A^I). \qquad (5.4)$$

The set A is the *extent* and the set B is the *intent* of the formal concept. The set of formal concepts becomes a poset [notation: $\mathfrak{R}(K)$] with the ordering relation

$$(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow$$
$$A_1 \subseteq A_2 (\Leftrightarrow B_2 \subseteq B_1). \qquad (5.5)$$

The poset $\mathfrak{R}(K)$ can be turned into a complete lattice, denoted by $\underline{\mathfrak{R}}(K)$, with the following definitions of infimum and supremum:

$$\text{infimum: } \bigwedge_j (A_j, B_j) = \left(\bigcap_j A_j, \left(\bigcup_j B_j \right)^{II} \right)$$
$$\text{supremum: } \bigvee_j (A_j, B_j) = \left(\left(\bigcup_j A_j \right)^{II}, \bigcap_j B_j \right). \qquad (5.6)$$

Concept lattices are useful for the representation of conceptual structure of data. There are efficient procedures for constructing formal concepts and the concept lattice from a given formal context (Kim and Compton 2004, Wille 2005).

5.3.2 Term-Document Matrix and Concept Lattice

Cheung and Vogel (2005) view the term-document matrix $TD_{n,m}$ (in its Boolean form, i.e., adjacency matrix, **Table 5.1**) as a formal context that is transformed into a concept lattice (**Fig. 5.6**).

Table 5.1. Term-Document Matrix

	D_1	D_2	D_3	D_4
T_1	1	1	0	0
T_2	1	0	1	0
T_3	0	1	0	1
T_4	0	0	1	1

Thus, term T_1 occurs in documents D_1 and D_2, term T_4 occurs in documents D_3 and D_4, and so on. The term-document matrix can be transformed into a concept lattice using the following method:

Generation of a Concept Lattice from the Term-Document Matrix

1. The least element, **0** (as well as the greatest element, **1**) of the concept lattice is introduced artificially.

2. The lattice is built in a bottom-up fashion (i.e., from **0** to **1**).

3. Every term T_i corresponds to an atom.

4. For every column j (j = 1,...,m), if $TD_{i,j} = TD_{k,j} = 1$, then document D_j is the meet (superconcept) of terms T_i and T_k.

This method can be applied even when the TD matrix is not Boolean, i.e., when it contains (nonbinary) weights. In this case, the condition in point 4 is rewritten as $TD_{i,j}$, $TD_{k,j} \neq 0$. (The superconcepts, as given by the TD matrix, may not be unique. However, in the concept lattice only one is allowed.)
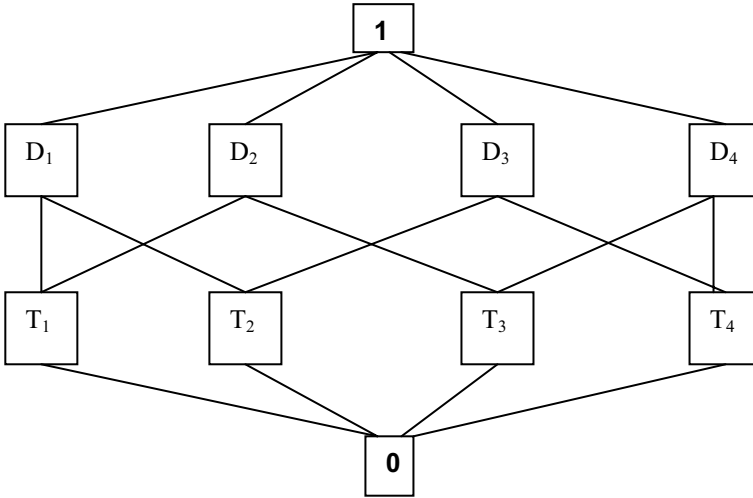
Fig. 5.6. Concept lattice obtained from the term-document matrix of Table 5.1.

A method to obtain a lattice from a term-document matrix is given in (Godin et al. 1989).

Each element of the lattice is a couple (d, t) such that

- d is the set of documents described by at least the terms in t.
- t is the set of terms common to all the documents in d.

Thus, t is the set of terms appearing in a conjunctive query retrieving exactly the documents in d. The set of all such couples is a lattice with the following partial order defined from the corresponding order on the term sets:

$$c_1 = (d_1, t_1) < c_2 = (d_2, t_2) \Leftrightarrow t_1 \subset t_2. \qquad (5.7)$$

Equivalently,

$$t_1 \subset t_2 \Leftrightarrow d_2 \subset d_1. \qquad (5.8)$$

The partial order is used to generate the Hasse diagram of the lattice. There is an edge (c_1, c_2) if $c_1 < c_2$, and there is no other element c_3 such that $c_1 < c_3 < c_2$.

5.3.3 BR-Explorer System

Messai et al. (2006) propose a retrieval system called BR-Explorer based on concept lattices. The term-document Boolean matrix (conceived as a formal context) is first transformed into a concept lattice L. A query Q is conceived as being a set of terms (attributes). In order to answer query Q, this is inserted into the concept lattice L first (e.g., by building L from scratch, or using some more efficient method as suggested by Messai et al.).

Relevance is defined as follows. The document d is relevant to query Q if they share at least one attribute. Messai et al. give a retrieval method (i.e., traversal of the concept lattice) that retrieves documents already ranked by their relevance degree.

5.3.4 Rajapakse-Denham System

Rajapakse and Denham (2006) proposed another application of lattices to IR. Documents and queries are represented as individual lattices. Concepts extracted from documents are used to construct a lattice. A document or query is conceived as a structure of objects, attributes, and their relationships from which a lattice is generated. The atoms are the elements consisting of objects that have identical attributes. On the next level, the elements are the objects that share most of their attributes, and so on. The smallest element of the lattice, at the bottom, is an artificial empty element, whereas the largest element, on the top, is the union of all the objects.

Retrieval is defined as follows. The relevance of a document to a query is determined on the basis of their common concepts. This is achieved by comparing nodes of the query lattice with the nodes of the document lattice. A partial match between the query lattice and the document lattice is defined as being the meet between their corresponding objects and attributes. When any of these two meets are empty, a keyword match is applied.

Rajapakse and Denham showed that their model worked by building an experimental system whose relevance effectiveness was measured on the Cranfield test collection. Moreover, the system was enhanced with a (personalized) learning strategy. In a relevance feedback process, all the terms of the query that are not present in a relevant document are added to that document. Weighting strategies were also used to refine the model.

5.3.5 The FooCA System

The FooCA system applies formal concepts to enhance Web retrieval (Koester 2006a,b, 2005). The system runs under Linux and is written in PERL. FooCA lets the user enter a query, which it sends to Google (other Web search engines can also be used).

The hit list returned by the search engine is used to construct a formal context and formal concepts as follows. The snippets (i.e., the short excerpts returned by the search engine) are used to extract terms. If there is no snippet, then the page address is used instead. The URLs of pages are viewed as objects, whereas its terms are viewed as attributes.

The hit list returned by the search engine is presented to the user as a table in which the rows correspond to objects (in the ranked order given by the search engine) and the columns to attributes. The table can be navigated using the mouse, and the corresponding row is highlighted. When the user clicks on a row, he/she is taken to the corresponding page.

The table can also be used for query refinement. If the user clicks on an attribute, then he/she can launch another search using the clicked attribute as the query or he/she can include or exclude that term into/from the original query.

FooCA can be used to visualize the hierarchy of formal concepts, which helps the user to assess the hits better and thus to know which hit to view first.

5.3.6 Query Refinement, Thesaurus Representation

Carpineto and Romano (2005) offer an excellent overview of other uses of concept lattices in retrieval. One such application of concept lattices is query refinement. Given a Boolean query (i.e., a Boolean expression of terms), the matching documents are found first. Then, the set of common terms in the retrieved documents is determined and used to build a concept lattice. The query can be refined by choosing the most general term (concept) that contains all the query terms.

Another use of lattices in retrieval is the representation of a thesaurus as a concept lattice by taking into account the ordering suggested by the thesaurus. The concept lattice of a document collection may be used as an underlying clustering structure. The query is merged into this lattice. Each document is ranked according to the shortest path between the query and the document concept.

Concept lattices can also be used to bound the search space or for navigational purposes. A possible application relates to Web searching. A set of pages returned by a search engine is parsed, and a concept lattice is built using the pages as objects and the terms as attributes. The user is presented with this lattice to initiate the next search interaction.

5.4 Properties of the Lattices Applied

As seen in the retrieval systems that have been described, lattices are used as mathematical models of the entities involved: objects and their relationships (document sets, structure of a document, within document-term relationships, queries, term relationships in general, and concepts). These lattices have different meanings (or roles) such as query, document, or concept. According to their role, they are subjected to appropriate processing. Retrieval is defined as a matching between a document and a query lattice, or between lattices (a query lattice and a document lattice). Different specific matching algorithms were proposed.

The lattices used are complex (albeit that there are attempts to find methods to reduce their complexity), and their construction is not an easy task.

Godin et al. (1998) showed that the number H of nodes in a concept lattice has linear complexity with the number n of documents:

$$H = \mathrm{O}(n \cdot 2^k), \tag{5.9}$$

where O denotes "big-Oh" (upper bound) from computational complexity, and k denotes the maximum number of terms/document. Experimental results showed that the ratio H/n was fairly constant and much lower than 2^k.

Cheung and Vogel (2005) applied SVD (singular value decomposition) to reduce the size of the term-document matrix, whereby the corresponding lattice became considerably smaller than the one corresponding to the original matrix.

From a purely formal point of view, the application of lattices in retrieval systems can be characterized as follows. Let

$$T = \{t_1, t_2, \ldots, t_n\} \tag{5.10}$$

denote a set of elements (e.g., terms, documents, etc.). Several types of lattices are defined over T, such as:

- Atomic, complete.
- Boolean algebra.
- Complete, atomic, complemented, nonmodular (hence not distributive).

- Complete, atomic, nonmodular (hence not distributive), not complemented.

We show now that, in Mooers's model, the following property holds:

Theorem 5.1. *The lattices of Boolean documents and queries are*
  1. Atomic and complete.
  2. Complemented.
  3. Not modular.

Proof (using **Fig. 5.3** as a model). Point (1) is straightforward. To prove point (2), note that the complement A^C of any atom A is equal to any expression of which A is not a member. The complement A^C of any nonatom A is equal to its negated counterpart. For point (3), we give a counterexample. By definition, a lattice L is modular if

$$(\forall A, B, C \in L \text{ for which } A \leq C) \Rightarrow$$
$$A \vee (B \wedge C) = (A \vee B) \wedge C.$$

For example, $A \subseteq C = \{A, B\}$, i.e., $A \leq (A \vee B)$ (recall that $\vee = \cup$ and $\wedge = \cap$). By taking $B = \{\neg A, \neg B\}$, we now have

$$A \cup (\{\neg A, \neg B\} \cap \{A, B\}) = A \cup \mathbf{0} = A,$$

which is not equal to

$$(A \cup \{\neg A, \neg B\}) \cap \{A, B\} = \mathbf{1} \cap \{A, B\} = \{A, B\}. \ \square$$

Further, we can prove that the facet lattices used in the FaIR system have the following property:

Theorem 5.2. *The facet lattice is*
  1. Atomic and complete.
  2. Not modular.

Proof. Point (1) is straightforward. For point (2), we give a counterexample (using **Fig. 5.5**). For $A = $ 'X-KL', $B = $ 'CGI' and $C = $ 'Manual', we have $A \leq C$, and

$$A \cup (B \cap C) = A \cup \mathbf{0} = \text{“X-KL”},$$

which is not equal to

$$(A \cup B) \cap C = \text{“Programming language”} \cap C = \text{“Manual.”} \ \square$$

Before proceeding with the analysis of the properties of lattices used in retrieval systems, we introduce further concepts related to lattices.

Definition 5.1. Two lattices $(L_1, \wedge_1, \vee_1)$ and $(L_2, \wedge_2, \vee_2)$ are *isomorphic* if there exists a bijective function $f: L_1 \rightarrow L_2$ such that

$$f(a \wedge_1 b) = f(a) \wedge_2 f(b),$$

$$f(a \vee_1 b) = f(a) \vee_2 f(b), \forall\ a, b \in L_1.\ \square$$

(5.11)

Definition 5.2. The structure $(L_1, \wedge, \vee)$, where $L_1 \subseteq L$ and $L_1 \neq \varnothing$, is a *sublattice* of the lattice $(L, \wedge, \vee)$ if

$$a \wedge b \in L_1,\ a \vee b \in L_1, \forall a, b \in L_1.\ \square$$

(5.12)

Definition 5.3. A lattice L_1 can be *embedded* into a lattice L_2 if there exists a sublattice of L_2 isomorphic to L_1. $\square$

There is an important relationship between the pentagon lattice and nonmodularity, namely:

Theorem 5.3. (Burris and Sankappanavar, 2000) *A lattice L is not modular if and only if the pentagon lattice can be embedded into L.*

Proof. It is clear that if the pentagon lattice can be embedded into a lattice L, then L is not modular (because the pentagon lattice itself is not modular; see Section 3.6). In order to prove the reverse, let us assume that L is not modular. This means that for $A, B, C \in L$ such that $A \leq C$ we have $A \vee (B \wedge C) < (A \vee B) \wedge C$. Let $D = A \vee (B \wedge C)$, then

$$B \vee D = B \vee (A \vee (B \wedge C)) =$$
$$B \vee ((B \wedge C) \vee A) =$$
$$(B \vee (B \wedge C)) \vee A =$$
$$B \vee A.$$

Now let $E = (A \vee B) \wedge C$; then

$$B \wedge E =$$
$$B \wedge ((A \vee B) \wedge C) =$$
$$(B \wedge (A \vee B)) \wedge C =$$
$$B \wedge C.$$

$D < E$ by assumption. Also $B \wedge C < A \vee (B \wedge C) = D$. Thus $B \wedge C < D < E$, and so

$$B \wedge C <$$
$$B \wedge D <$$
$$B \wedge E = B \wedge C.$$

Hence, $B \wedge D = B \wedge E = B \wedge C$. Likewise, $B \vee E = B \vee D = B \vee A$. **Figure 5.7** shows the copy of the pentagon lattice in L.
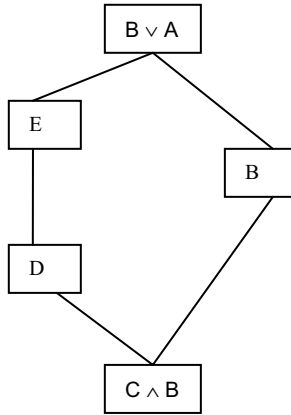

Fig. 5.7. Copy of the pentagon lattice as an embedded lattice.

Wille (2005) gives many examples of concept lattices, which model a wide range of different real situations:

- Geography (bodies of water).
- Sociology (economic concepts of young persons).
- Geometry (types of triangles, inversion of a circle).
- Medicine (examination of anorectic patient, functional rooms in a hospital, Ph-level of children with diabetes).
- Tourism (leisure activities).
- Information science (information and knowledge processing).
- Urbanism (town and traffic).
- Music (musical attributes).
- Biology (animals).

The concept lattice of the Ph-level of children with diabetes is shown in **Fig 5.8**. (This concept lattice was generated using the data from 111 children and 22 attributes in collaboration with medical experts.) It can be seen that this concept lattice is nothing other than the pentagon lattice, and hence it is not modular.
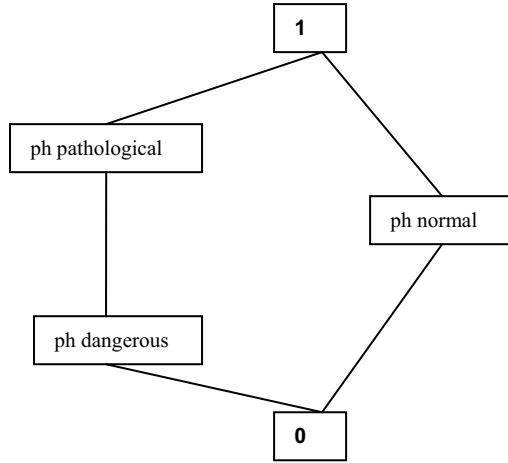
Fig. 5.8. Concept lattice of Ph-level of children with diabetes.

If one examines the other lattices given by Wille, it turns out that other concept lattices (e.g. those of economic concepts of young persons, of types of triangles, and of animals) also have sublattices isomorphic with the pentagon lattice, which means that they are not modular. Indeed, we can show that in general:

Theorem 5.4. *Galois (concept) lattices are not modular.*

Proof. A lattice L is, by definition, modular if

$$(\forall A, B, C \in L \text{ for which } A \leq C) \Rightarrow$$

$$A \vee (B \wedge C) = (A \vee B) \wedge C.$$

For Galois lattices, the definition of modularity becomes

$$(\forall (X_1, X_2), (Y_1, Y_2), (Z_1, Z_2) \in L \text{ such that } (X_1, X_2) \leq (Z_1, Z_2)) \Rightarrow$$

$$(X_1, X_2) \vee ((Y_1, Y_2) \wedge (Z_1, Z_2)) =$$

$$((X_1, X_2) \vee (Y_1, Y_2)) \wedge (Z_1, Z_2).$$

Using the definitions of join $\vee$ and meet $\wedge$ in concept lattices, and the hypothesis $(X_1, X_2) \leq (Z_1, Z_2)$, we rewrite the modularity condition as follows:

$$((X_1 \cup (Y_1 \cap Z_1))^{II}, X_2 \cap Y_2^{II}) =$$

$$((X_1 \cup Y_1)^{II} \cap Z_1, ((X_2 \cap Y_2) \cup Z_2)^{II}),$$

which is equivalent to

$$(((X_1 \cup Y_1) \cap Z_1)^{II}, X_2 \cap Y_2) =$$
$$((X_1 \cup Y_1)^{II} \cap Z_1, ((Z_2 \cap (Y_2 \cup Z_2)^{II}).$$

Let us take now $(X_1, X_2) \le (Y_1, Y_2)$, i.e., $X_1 \subseteq Y_1, Y_2 \subseteq X_2$. Then, the previous condition is rewritten as

$$((Y_1 \cap Z_1)^{II}, Y_2) =$$
$$(Y_1^{II} \cap Z_1, ((Z_2 \cap (Y_2 \cup Z_2)^{II}).$$

The equality holds when $(Y_1 \cap Z_1)^{II} = Y_1^{II} \cap Z_1$, which is true because $Y_1^{II} = Y_1$, and so we have:

$$(Y_1 \cap Z_1)^{II} = Y_1 \cap Z_1;$$

and when

$$Y_2 = (Z_2 \cap (Y_2 \cup Z_2)^{II} = Z_2 \cap (Y_2 \cup Z_2).$$

But if $Y_2 \subset Z_2$, then $Z_2 \cap (Y_2 \cup Z_2) = Z_2 \cap Z_2 = Z_2$, which is different from Y_2. $\square$

However, in nonmodular lattices certain pairs of elements may satisfy the modularity condition.

Definition 5.4. An ordered pair (A, B) of elements—i.e., in this order A is the first element of the pair and B is the second element of the pair—of a lattice L is referred to as a *modular pair* (notation: $A\mathsf{M}B$) if

$$(\forall\, C \in L \text{ such that } C \le B) \Rightarrow$$
$$C \vee (A \wedge B) = (C \vee A) \wedge B. \;\square \tag{5.13}$$

If A and B are not modular pairs, then this is denoted by $A\bar{\mathsf{M}}B$. Albeit that Galois (concept) lattices are not, in general, modular, in the concept lattices in Wille (2005), which all have a sublattice isomorphic to the pentagon lattice, there are modular pairs that correspond to the following modular pair in the pentagon lattice:

$$\mathbf{0} \le x \Rightarrow$$
$$\mathbf{0} \vee (x \wedge y) = \mathbf{0} \vee x = x =$$
$$(\mathbf{0} \vee x) \wedge y = x \wedge y \tag{5.14}$$
$$= x.$$

This means that y and x are a modular pair: $y\mathsf{M}x$. A Galois lattice has a sublattice isomorphic with the pentagon lattice. Hence, it also has a modular

pair of elements. We should note, however, that $x\mathbf{M}y$. We have $x \le y$, but $x \vee (z \wedge y) = x$, which is not equal to $(x \vee z) \wedge y = y$. This means that x and y are not a modular pair: $x\mathbf{M}y$.

It is known that the logical propositions of mathematical logic form a complemented and distributive lattice $(\{T, F\}, \vee, \wedge, \neg)$. As concept lattices are not modular (Theorem 5.4), they are not distributive either. Hence, the main difference between concept lattices and the lattice of propositions relates to the presence/absence of distributivity. In logic, distributivity is a property that connects conjunction and disjunction and is the expression of compatibility between any two propositions P and Q in the sense that

$$(P \wedge Q) \vee (P \wedge \neg Q) = P \vee (Q \wedge \neg Q) = P. \tag{5.15}$$

The fact that concept lattices are not distributive means that, in general, there are objects and/or properties that are not compatible, i.e., about which we cannot always reason in the sense of mathematical logic. For example, in the concept lattice of **Fig. 5.6**, we have

$$(D_1 \wedge D_2) \vee (D_1 \wedge \neg D_2{}^{C}) = T_1, \tag{5.16}$$

which means that reasoning with documents D_1 and D_2 does not result in some other document, but rather in a term (which is a different type of entity). In other words, in concept lattices, reasoning may lead to an object having a different nature or quality than the nature of objects on which reasoning has operated (a situation unimaginable in mathematical logic).

5.5 Exercises and Problems

1. Using a document collection of your choice, construct the corresponding concept lattice (using the term-document matrix).

2. Show that facet lattices are not distributive.

3. Are concept lattices uniquely complemented?

4. Is the Boolean algebra of documents uniquely complemented?

5. Are concept lattices orthomodular?