Chapter 6: Compressed pseudo-lattices

'Everything should be made as simple as possible, but not simpler.'

Albert Einstein

In this chapter we define the notion of a compressed pseudo-lattice. A compressed pseudo-lattice essentially consists of a sublattice embedded in a bipartite graph. This allows for the reduction of the size of the lattice but allows control over the amount of information that is lost in the process. The aim of this is to simplify the lattice but still retain the essence of the context it represents.

The discussion starts by introducing and developing an IR (Information Retrieval) problem and develops the problem domain into one where it is argued that a compressed pseudo-lattice plays a significant role. The properties and use of compressed pseudo-lattices are discussed as well as their interpretation. It is argued that this approach may have significant advantages over approaches using the complete lattice in particular areas.

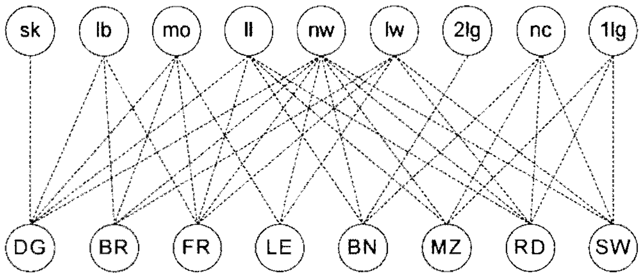
6.1 A BIPARTITE DATABASE AND QUERY OPERATION

We now sketch an IR problem domain and do not consider lattices until the next section. For this problem domain we define a database $D = \langle S, \leqslant \rangle$ related to a context $C = \langle O, A\rangle$ as consisting of a set, S, of concepts which are partially ordered by the relation $\leqslant$ (this set need not be a lattice although it is one of the possibilities). An incidence relation I can be derived from the partial order that describes which objects possesses which attributes. The database is restricted in that the maximal elements are the attribute concepts (representing A) in C and the minimal elements are the object concepts (representing O) in C. In addition D may contain any number of intermediate concepts M (i.e. S = attribute concepts $\cup$ intermediate concepts $\cup$ object concepts). The *upward closure* of any concept c, denoted by $\mathrm{UpwardClosure}(D, c)$, is the set of all concepts greater than or equal to c in terms of the partial order, $\leqslant$. The *downward closure* is the set of concepts that are less than or equal to c, and is denoted by $\mathrm{DownwardClosure}(D, c)$. The *extent* of a concept is defined as the set of objects in its downward closure. Similarly the *intent* of a concept is the set of attributes in its *upward closure*.

We consider the problem of retrieving a set of objects relevant (in some abstract and as yet undefined way) to a specific query. The query is formulated as a set of attributes in the form $Q = \{a_1, a_2, ..., a_m\}$ (i.e. $Q \subseteq A$). Different query operations, taking Q as a parameter, can be defined on D. The result of a specific query operation O based on database D with respect to query Q is denoted by $O(D, Q)$. A query operation may return any number of concepts from D, the objective being the identification of concepts relative to the query Q.

Notwithstanding the foregoing, we choose to interpret the final results in terms of objects, namely those objects that are in the union of the extents of the concepts returned by the query operation. As a consequence the results of a query can be interpreted as clusters of objects represented or referenced by the concepts returned by the query operation.

Different query operations may be evaluated in terms of the well-known information retrieval (IR) metrics, precision (the proportion of objects returned by $O(D, Q)$ that are relevant in relation to all objects returned by O(D, Q)) and recall (the proportion of relevant objects returned by O(D, Q) in relation to all relevant objects in the database, D) (Salton 1989). Conversely using the same query operation, different databases of the same context can be evaluated against each other in terms of these metrics. Clearly only concepts in a given database can be returned. Therefore it can be expected, for a given query, that a database containing more 'meaningful' concepts will return concepts that result in higher precision and recall values. We argue that a compressed pseudo-lattice (as defined later on in this chapter) is a versatile data structure to represent various databases of this type and could prove useful in researching information retrieval and machine learning strategies.



Incidence relation									
	nw	lw	ll	nc	1lg	2lg	mo	lb	sk
LE	✓	✓					✓		
BR	✓	✓					✓	✓	
FR	✓	✓	✓				✓	✓	
DG	✓		✓				✓	✓	✓
SW	✓	✓		✓	✓				
RD	✓	✓	✓	✓	✓				
BN	✓		✓	✓		✓			
MZ	✓		✓	✓	✓				

Figure 6.1: The Living Context and its associated bipartite graph

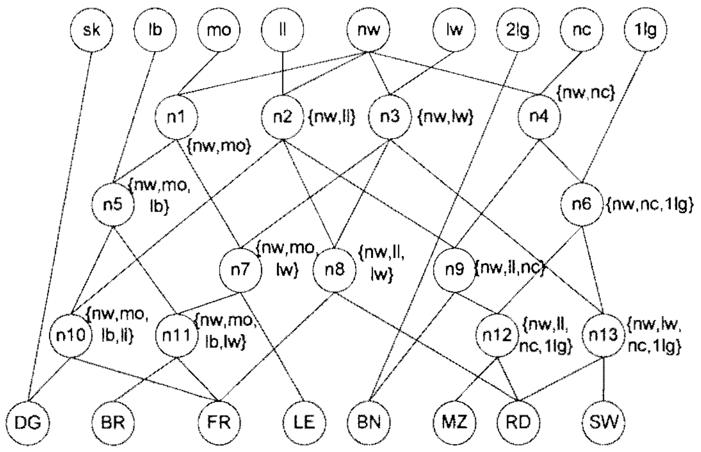
The simplest example of such a database is that of a bipartite graph (essentially representing the incidence relation of the context) with objects at the bottom, attributes at the top and arcs from each object to all the attributes it possesses in a specific context, as illustrated in figure 6.1.

Consider $O_{BP}(D, Q)$, a query operation on the bipartite database, D in figure 6.1. For a query Q we define $O_{BP}(D, Q) = Q$. As a trivial example we see that for Q = {mo, lw} (where the key of figure 6.1 indicates that mo means 'motile' and lw means 'lives in water', etc.) the query $O_{BP}(D, Q)$ would return {mo, lw}, effectively referencing the set of objects {LE, BR, FR, DG, SW, RD} (i.e. the extent of the concepts mo and lw). This can be verified by inspecting the line diagram in figure 6.1 of the database for DownwardClosure(D, mo) = {LE, BR, FR, DG} and DownwardClosure(D, lw) = {LE, BR, FR, SW, RD}.

A shortcoming of the bipartite database and of O_{BP} is the fact that the query operation returns a very general set of objects, each of which has any, but not all, of the attributes specified in the query Q. (Thus leech, bream, frogs, spike-weed and seeds are either motile and/or live in water.) In IR terms the query operation has low precision but high recall. One way of improving the precision is to introduce a new intermediate concept called 'mo_lw' that groups all objects that possesses both mo and lw into the database. This concept would be connected via upward arcs to mo and lw and all objects possessing mo and lw would have upward arcs to the new concept. In this way, a query operation might be able to use the new concept in arriving at the results of the query presumably yielding better results.

Continuing this line of thought and introducing new 'useful' or 'meaningful' intermediate concepts, the other end of the spectrum would be to use a formal concept lattice or EA-lattice as the database. Databases of this type are discussed in the next section.

6.2 A CONCEPT LATTICE DATABASE AND QUERY OPERATION



Incidence relation									
	nw	lw	ll	nc	1lg	2lg	mo	lb	sk
LE	✓	✓					✓		
BR	✓	✓					✓	✓	
FR	✓	✓	✓				✓	✓	
DG	✓		✓				✓	✓	✓
SW	✓	✓		✓	✓				
RD	✓	✓	✓	✓	✓				
BN	✓		✓	✓		✓			
MZ	✓		✓	✓	✓				

Entities/objects		Attributes	
LE	Leech	nw	Needs water
BR	Bream	lw	Lives in water
FR	Frog	ll	Lives on land
DG	Dog	nc	Needs chlorophyl
SW	Spike-weed	1lg	1 leaf germination
RD	Reed	2lg	2 leaf germination
BN	Bean	mo	Is motile
MZ	Maize	lb	Has limbs
		sk	Suckles young

Figure 6.2: The EA-lattice of the Living Context (the unit- and zero concepts are not shown)

Figure 6.2 shows an EA-lattice for the Living Context. It has clearly partitioned sets of attributes (top row of concepts), objects (bottom row) and intermediate concepts. The unit- and zero concepts of the lattice have been excluded from the database. (They are, however always implied.) The intents of some of the EA-formal concepts are shown as a guide.

$O_{Meet}(D, Q)$ is a query operation on a lattice database and is defined as the meet or infimum of the attributes of Q in the lattice (i.e. the concept $\langle Q', Q''\rangle$ corresponding to all the objects that have the attributes Q in common and all the attributes this set of objects have in common). For the query $Q = \{mo, lw\}$ the resulting objects are therefore $\{BR, FR, LE\}$ since the meet of $\{mo, lw\}$ is concept n_7 which has an extent of $\{BR, FR, LE\}$ (i.e. objects possessing all of the attributes in Q are returned). Note that it is now possible to obtain a result with a higher precision due to the fact that concepts lower down in the database discern between objects in a more granular way. (Thus leeches, bream and frogs are all both motile and live in water.)

6.3 AN ADAPTED SUBLATTICE DATABASE AND QUERY OPERATION

Assuming that we were looking for all the fish objects in the Living Context (in this case only BR qualifies) with query $Q = \{mo, lw\}$ (i.e. all the living objects that can move and live in water). The lattice-based query operation O_{Meet} has a higher precision and recall than O_{BP}. O_{Meet} has however the disadvantage that it is not tolerant of errors or ambiguity in either the context or formulation of the query terms as indicated in the following example.

Assume, for example, that we are looking for all edible plants in the context using the query $Q = \{nw, nc, 1lg, 2lg\}$. (The key of figure 6.1 indicates that nw means 'needs water', nc means 'needs chlorophyll', 1lg means 'one leaf generation' and 2lg means 'two leaf generation', all attributes being related to edible plants.) $O_{Meet}(D, Q)$ would not return any relevant objects since the meet of Q is not in the database – the meet is in fact the zero concept, 0_L, in the lattice. In this case the query was too specific and the query operation was unable to find a concept corresponding to exactly $Meet(D, Q)$. This is clearly not ideal since the database did contain objects relevant to Q and the query operation should ideally have coped with the situation.

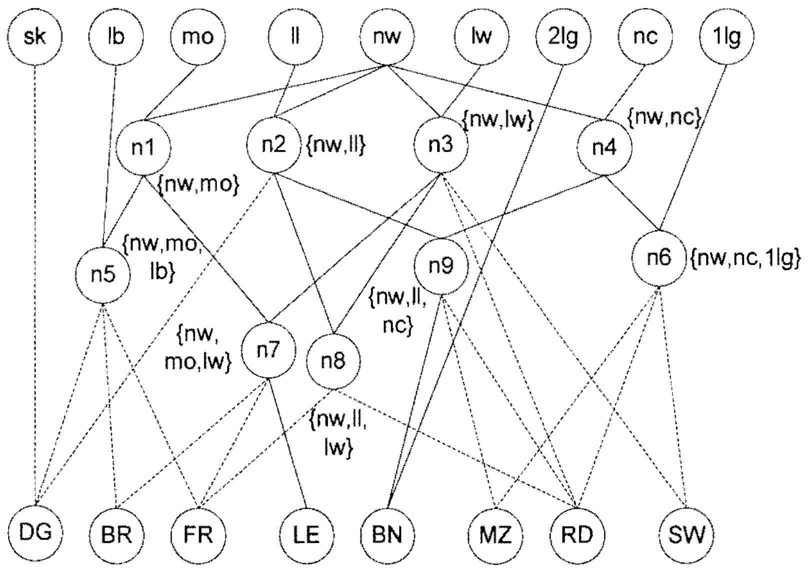


Figure 6.3: A database (compressed pseudo-lattice) with intermediate concepts with an intent that has more than three attributes removed

One strategy that could help in this case and increase the tolerance for errors is to specify a query operation for a context of n attributes that will return the minimal concepts that have at most $k \leq n$ attributes, all of which are in Q. Since the domain of discourse as defined requires that queries only be formulated in terms of concepts already contained in the database we can adopt one of two strategies. The first is to redefine the query operation to examine all concepts and return the appropriate minimal concepts. In this case, the database D is kept the same. A second option is to modify the query operation and the database. For reasons that will become clear, we will pursue the latter option.

Removing all intermediate concepts in the lattice in figure 6.2 that have more than $k = 3$ attributes creates the new database in figure 6.3. Where the original EA-lattice concepts have been removed, dashed arcs indicate successors defined by the partial ordering relation. Note that a subset, L, of the database namely all the concepts except DG, BR, FR, MZ, RD, SW and BN still forms a sublattice when the unit- and zero concepts are appropriately inserted (i.e. it forms a poset of which the supremum and infimum of all pairs of concepts exist and therefore a sublattice of the EA-lattice in figure 6.2 - this may be verified by inspection). This lattice (identified by all the concepts connected with solid arcs as well as all attribute concepts plus the implied unit- and zero concepts) does not correspond to either the formal concept lattice or EA-lattice lattice for the given context but is a sublattice of the EA-lattice of the Living Context. A query operation on the database in figure 6.3 now cannot discern between objects that have more than k attributes in common and would therefore presumably still return objects even when the query is too specific.

6.4 REMOVING CONCEPTS FROM A LATTICE

An EA-lattice as defined in chapter 2 has a fixed set of concepts for a fixed context. It is however possible to remove some of the concepts from the EA-lattice and still have a complete sublattice (albeit not a formal concept lattice or EA-lattice) that is based on the original partial ordering relationship[10]. For example, an EA-lattice can clearly be reduced

[10] Throughout this discussion, it is assumed that the same ordering relationship is used. Thus the ordering $x \leq_{EA} y$ either holds in all lattices in which concepts x and y appear, or it holds in none of them.

to a formal concept lattice by removing EA-formal concepts defined by EA-formal conditions 1 to 4 (section 2.5) if they are not generated by condition 5, but retaining EA-formal concepts defined by condition 5. Appropriately removing even more concepts will result in a set of concepts that constitute a lattice (i.e. a complete sublattice), but not a formal concept lattice (as per example in section 6.3).

In order to ensure the retention of lattice properties, when removing concepts from a lattice, steps must be taken to ensure that both the supremum and infimum of any two remaining concepts exist and are unique (as required by the definition of a lattice). Should this not be the case, the resulting set of concepts will not be a sublattice. Removing a concept from a lattice thus involves not only the removal of the concept from the lattice's set of concepts, but also the revision of the order relationship so that the parents and children of the removed concept are partially ordered as per the original $\leq_{EA}$ ordering relationship. Removing atoms or coatoms from an EA-lattice is a particular case where this is possible as formulated in the next theorem.

Theorem 6.1: Removing an atom or coatom from an EA-formal concept lattice or sublattice results in a set of remaining concepts that is a complete sublattice with respect to the partial ordering relation $\leq_{EA}$.

Proof: Let L_0 be the concept lattice and L_1 the set of concepts remaining after removing an atom or coatom concept. To prove that L_1 is a lattice we need to prove that two arbitrary concepts $x, y \in L_1$ have a unique supremum and infimum in relation to $\leq_{EA}$.

We first prove that x and y have a unique infimum. By implication $x, y \in L_0$. Let $q = \mathrm{Inf}(L_0, \{x, y\})$. Two alternatives exist depending on whether q is in L_1 (i.e. whether it was possibly an atom removed from L_0 or not). If q is indeed in L_1 then q is a lower bound of x and y because $q \leq_{EA} x$ and $q \leq_{EA} y$ in L_1 since q was the infimum of x and y in L_0 and the partial ordering relationship of L_1 was redefined to preserve all the transitive ordering relationships between the concepts of L_0 in L_1. Since q was also the unique greatest lower bound of x and y in L_0 and no other concepts have been added to L_1, q is therefore also the unique greatest lower bound of x and y in L_1. Thus, the infimum of x and y in L_1 exists and is unique.

If q is not in L_1 then q must be an atom that was removed from L_0. (q can not be a coatom since then $x = q$ or $y = q$ in which case either x or y will not be part of L_1.) Furthermore, any possible lower bound of x and y in L_1 must be a concept that is smaller than q in L_0, since q was the unique greatest of all lower bounds of x and y in L_0. But q is an atom in L_0 and therefore it has only one child concept in L_0 namely 0_L. The 0_L is a lower bound of all concepts in L_1 and L_0 including x and y. Since q do not exits in L_1, 0_L also is the only lower bound of x and y. The infimum of x and y in L_1 is therefore unique.

A similar argument can be used to prove that the supremum of x and y in L_1 is unique with q possibly being a coatom. Since the supremum and infimum of x and y in L_1 exist and are unique, L_1 is a sublattice.

Using this theorem it is therefore clear that atom or coatom concepts can be removed from a lattice (or sublattice) without violating the lattice property. Since this theorem is generic, it can be extended to apply to formal concept lattices. Removing any concept does however mean that the resulting lattice is not the EA-formal concept lattice (or formal concept lattice) of the specific context since all EA-formal concepts are not present. We follow the convention of calling these derived lattices, *sublattices* and only refer to a lattice as an EA-lattice (or formal concept lattice) when it contains all the EA-formal concepts (or formal concepts) of the context. Although some of the attributes or objects concepts may have been removed from the lattice (but not from the context – the context remains unchanged), the atoms and coatoms (i.e. concepts covering the zero concept or covered by the unit concept, respectively) of the resulting sublattice may effectively be regarded as its new objects and attributes respectively.

Theorem 6.1 can be generalised to removing whole areas of the lattice by progressively removing one atom or coatom at a time whether it involves object concepts, attribute concepts or intermediate concepts. The atoms and coatoms of the sublattice define it sufficiently since the resulting lattice still only contains EA-formal concepts. The set of atoms and coatoms of a sublattice is called a *cut* on the EA-lattice and consists of two (possibly overlapping) sets called the *atomic cut* and the *coatomic cut.* These latter sets correspond to the set of atoms or of coatoms of the sublattice respectively and both sets are therefore anti-chains. An EA-lattice can be reduced to a given sublattice by removing all concepts that are not comparable to elements of the atomic cut or the coatomic cut as discussed below. That is, if $y \in$ atomic cut of the sublattice and $x \in$ coatomic cut of the sublattice, then the following EA-formal concepts are retained in the sublattice: all EA-formal concepts c such that $y \leq_{EA} c \leq_{EA} x$.

There are five conditions under which EA-formal concepts (excluding 1_L and 0_L) are removed to create a sublattice defined by its cut on an EA-lattice:

- A concept that is smaller than some concept in the atomic cut.
- A concept that is larger than some concepts in the coatomic cut.
- A concept that is smaller than some concept in the sublattice's coatomic cut but not comparable to any concept in the atomic cut.
- A concept that is larger than some concept in the sublattice's atomic cut but not comparable to any concept in the coatomic cut.
- A concept that is not comparable to any concept in either the sublattice's atomic cut or the sublattice's coatomic cut.

Note that the zero- and unit concepts are always part of the sublattice since only atoms and coatoms may be removed. Due to the definition of an EA-lattice, the definitions of the zero- and unit concepts however remain constant and are not dependent on the elements of the sublattice. (They are however not shown in the compressed pseudo-lattice figures below).

All concepts not in the resulting sublattice (identified by the above conditions) can be progressively removed from the original EA-lattice. All these concepts and their relations with the remaining concepts are effectively 'compressed' into either the unit concept or zero concept. In figure 6.4, the relationship of the original EA-lattice to the sublattice's cut (formed by the atoms and coatoms of the sublattice) is schematically depicted.

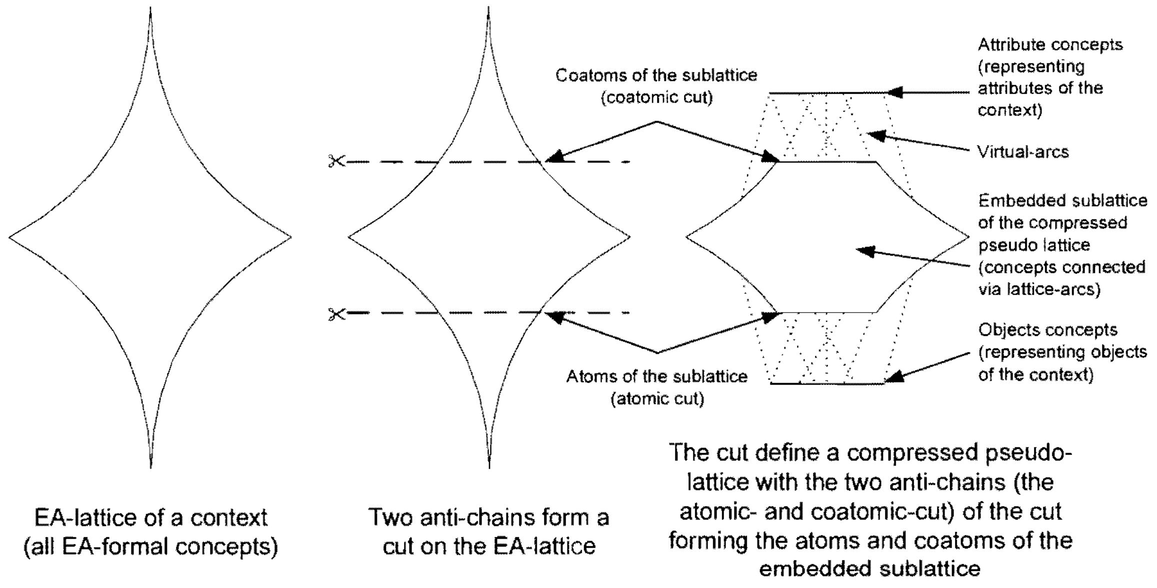

Coatoms of the sublattice (coatomic cut)
Attribute concepts (representing attributes of the context)
Virtual-arcs
Embedded sublattice of the compressed pseudo lattice (concepts connected via lattice-arcs)
Objects concepts (representing objects of the context)
Atoms of the sublattice (atomic cut)
EA-lattice of a context (all EA-formal concepts)
Two anti-chains form a cut on the EA-lattice
The cut define a compressed pseudo-lattice with the two anti-chains (the atomic- and coatomic-cut) of the cut forming the atoms and coatoms of the embedded sublattice

Figure 6.4: A cut on an EA-lattice defines a compressed pseudo-lattice (with an embedded sublattice) that can be created by the removal of atoms and coatoms of the EA-lattice

In the resulting compressed pseudo-lattice data structure we choose to keep the attribute and object concepts, but indicate their relationship to concepts that were previously in their upward or downward closure by using *virtual arcs*. The exact nature of these virtual arcs will be defined later.

It is important to note that, as a result of the five different conditions under which concepts can be removed from the original EA-lattice with respect to the cut, the nature of the removed concepts may be much more complex than depicted in figure 6.4. For example, instead of compressing the EA-lattice to a certain level (i.e. removing concepts with the same cardinality of the extent or intent), cuts may be defined to effectively remove entire sets of attributes, objects or areas from the EA-lattice. It is also worth noting that, in general, an arbitrary sublattice cannot be generated via compress lattice operations since removing non-atom concepts or non-coatom concepts can in certain circumstances also yield sublattices – there are thus limitations to this approach of generating sublattices.

6.5 THE USE OF THE INTENT- AND EXTENT REPRESENTATIVE OPERATIONS

Removing concepts from an EA-lattice has the effect of removing what was previously the infima and suprema (meets and joins) of certain subsets of concepts of the EA-lattice, these effectively moving to the zero- and unit concepts respectively.

Repeating the query operation O_{Meet} for $Q = \{nw, nc, 1lg, 2lg\}$ in figure 6.3, we find that there is no meet in the database (i.e. the meet is the zero concept in the sublattice – a 'trivial' meet). The intent- and extent representative operations defined in chapter 2 provides a logical solution for the problem and a revised query operation using these are therefore considered. This operation will define a 'second-order meet' in the case of a trivial meet.

Suppose a query operation, $O_{AIR}(D, Q)$, returns the approximate intent representatives (AIR) of Q in the sublattice embedded in D in figure 6.3, i.e. $O_{AIR}(D, Q) = AIR(D_{Embed}, Q)$ where D_{Embed} is the sublattice embedded in D (all concepts joined with normal arcs except those with dashed arcs such as DG, BR, FR, MZ, RD and SW). If $Q = \{nw, nc, 1lg, 2lg\}$ then S = {nw, 2lg, nc, 1lg, n_4, n_6, BN} and {n_6, BN} is the set of minimal elements of S. Thus

$O_{AIR}(D, Q)$ returns $AIR(D_{Embed}, Q) = \{n_6, BN\}$, assuming D_{Embed} is the sublattice in figure 6.3. $O_{AIR}(D, Q)$ thus references the objects {BN, MZ, RD, SW} and therefore solves the problem of a too specific query.

Inspecting the intents of the concepts in AIR(D, Q) we see that BN, for example, has the attribute ll in its intent that is not in Q. If we wish to restrict a query operation to find only concepts possessing attributes in Q (i.e. exactly representing Q), then we need to use the exact intent representative operation (EIR).

Let $O_{EIR}(D, Q) = EIR(D_{Embed}, Q)$. For Q = {nw, nc, 1lg, 2lg} we saw that $AIR(L, Q) = \{n_6, BN\}$ whilst $EIR(L, Q) = \{2lg, n_6\}$ since T = {BN} in the calculation of EIR. $O_{EIR}(D, Q) = EIR(D_{Embed}, Q) = \{2lg, n_6\}$. Thus, in the present example, $O_{EIR}(D, Q)$ references the same set of objects as before, namely {BN, MZ, RD, SW}.

The O_{EIR} and O_{AIR} operations can however be applied to the database in figure 6.1, in which the embedded sublattice is reduced to only the set of attributes. In that case, $O_{BP}(D, Q) = O_{EIR}(D, Q) = O_{AIR}(D, Q)$. If D is the sublattice in figure 6.2, then $O_{Meet}(D, Q) = O_{EIR}(D, Q)$ for a non-trivial Meet(D, Q).

The point is that both the O_{AIR} and O_{EIR} operations are defined in terms of a sublattice and should the sublattice be changed (keeping the same context) as in the examples for figures 6.1 to 6.3, the representative sets also change. When Q has a non-trivial meet (i.e. not the zero concept) in the lattice or sublattice then EIR(D, Q) = AIR(D, Q) = Meet(D, Q). The representative sets of Q were defined to deal with situations when Q has a trivial meet (as is often the case when working with sublattices) and yield better results. The operations may be seen as extensions of the meet or join operations.

6.6 THE COMPRESSLATTICE OPERATION

Incidence relation

	a	b	c	d	e
o1	✓	✓	✓	✓	
o2	✓	✓	✓	✓	✓
o3		✓	✓	✓	✓
o4	✓				✓

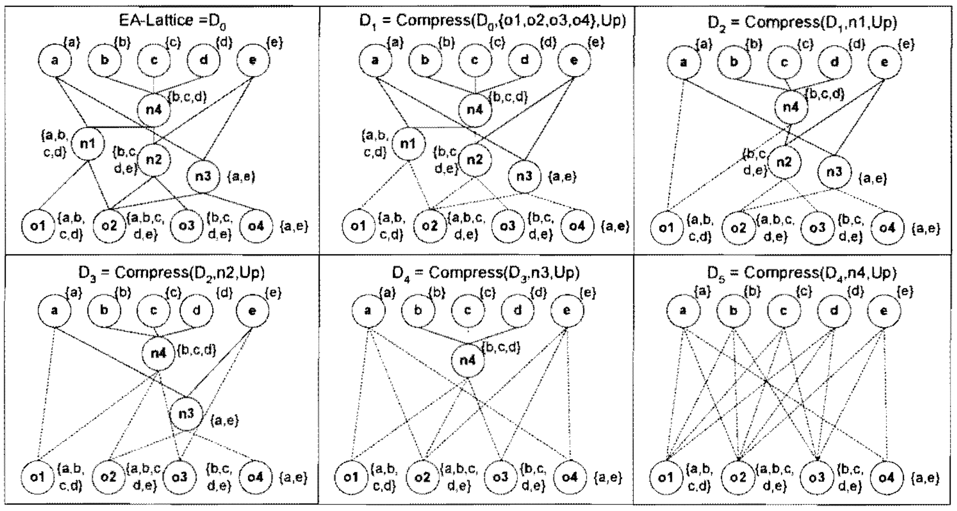

EA-Lattice =D_0
D_1 = Compress(D_0,{o1,o2,o3,o4},Up)
D_2 = Compress(D_1,n1,Up)
D_3 = Compress(D_2,n2,Up)
D_4 = Compress(D_3,n3,Up)
D_5 = Compress(D_4,n4,Up)
{a} {b} {c} {d} {e}
a b c d e
n4 {b,c,d}
n1 {a,b,c,d}
n2 {b,c,d,e}
n3 {a,e}
o1 {a,b,c,d}
o2 {a,b,c,d,e}
o3 {b,c,d,e}
o4 {a,e}

Figure 6.5: A CompressLattice example compressing EA-lattice D_0 to a bipartite graph, D_5

The CompressLattice operation removes an atom or coatom concept y from the sublattice embedded in the database and replaces the concept with virtual arcs (indicated as dashed arcs). The virtual arcs interconnect all the parent- with all the child concepts of y. Figure 6.5 shows an example of a compressed pseudo-lattice structure where all the intermediate concepts have been removed by successively using CompressLattice operations. Similarly, figure 6.3 can be verified to be the result of successive

CompressLattice operations on the EA-lattice in figure 6.2, removing the concepts DG, BR, FR, MZ, RD, SW, n_{10}, n_{11}, n_{12} and finally n_{13}.

It is important to note that the CompressLattice operation works from a *particular direction*. In the examples, the lattice was compressed in the *upward* direction, but the operation is equally valid when compressing the lattice from the top downward (or any combination of the two). Essentially, compression in the upward direction involves the removal of atoms, while compression in the downward direction involves the removal of coatoms from a sublattice.

The CompressLattice operation creates a data structure that is not an EA-lattice but one that does contain an embedded sublattice. This data structure is called a *compressed pseudo-lattice*. A compressed pseudo-lattice consists of EA-formal concepts interconnected by virtual- and lattice arcs. The concepts connected by lattice arcs (with 0_L and 1_L implied) define a sublattice called the *embedded sublattice* (of the compressed pseudo-lattice). The virtual arcs represent the relationship between the sublattice and the context. Using parameter names to imply types, the CompressLattice operation is defined as follows in terms of its pre- and post-conditions:

```
//=======================================================
CompressLattice(aCompressedLattice, aConcept, aDirection)
                  Return outCompressedLattice
//=======================================================
//Pre-condition: aConcept is an atom or coatom in the embedded
//sublattice in aCompressedLattice, it has at least one lattice arc in
//aDirection and no lattice arcs in the opposite direction
//(except to the unit- or zero concept).
//Post-condition: outCompressedlattice retains all the concepts
//(except possibly aConcept) and arcs of aCompressedLattice, except in
//the following respects. If aConcept is an attribute or object
//concept, then lattice arcs connecting it to other concepts in
//aCompressedLattice are replaced by virtual arcs in
//outCompressedLattice. Otherwise aConcept and its arcs are not in
//outCompressedLattice. Instead, virtual arcs link each of aConcept's
//parents to each of aConcept's children.
//=======================================================
```

Note that the definition above is in functional terms and the definition changes slightly when defined in an object-oriented fashion as discussed in chapter 7 where the references to aCompressedLattice and outCompressedLattice fall away.

6.7 DEFINITION AND PROPERTIES OF COMPRESSED PSEUDO-LATTICES

A compressed pseudo-lattice essentially represents a sublattice of an EA-lattice from which a number of atoms and/or coatoms have been removed. Additionally the relation of the sublattice to the context from which it was derived is preserved. As a data structure it represents a particular context $C = \langle O, A, I \rangle$. The data structure consists of a number of EA-formal concepts that are connected by one of two types of directed arcs: lattice arcs and virtual arcs. Lattice arcs preserve the existence suprema and infima across the concepts they interconnect; virtual arcs do not necessarily. The concepts are partitioned into three sets: the attribute concepts (of the context), the object concepts (of the context) and a number of intermediate concepts. A compressed pseudo-lattice contains an embedded sublattice. The embedded sublattice is the set of all concepts complying with one of the following:

- The concept is an attribute concept with no incoming virtual- or lattice arcs.
- The concept has at least one lattice arc connecting into or out of it.
- The unit and zero concepts.

This corresponds to the set of concepts remaining after reducing an EA-lattice to a sublattice by successive removal of atom and coatom concepts, as discussed previously. The properties of the compressed pseudo-lattice are thus implied by the CompressLattice operation. It is important to note that, for a given set of concepts, there may be more than one compressed pseudo-lattices that can be defined upon that set of concepts using different cuts on that lattice. This is due to the fact that concepts can be interconnected by either virtual- or lattice arcs. Both the concepts and arcs thus uniquely define the embedded sublattice. The context and the atoms and coatoms (i.e. the cut) of an embedded sublattice (i.e. the atomic and coatomic cut) also uniquely define a compressed pseudo-lattice.

The following are the *compressed pseudo-lattice properties*. They define sufficient conditions for a data structure to be a valid compressed pseudo-lattice. Note that the conditions listed are not disjoint – they may be related to or imply one another.

- ***EA-formal concepts:*** All concepts are EA-formal concepts as defined in chapter 2.
- ***Poset:*** Concepts in a compressed pseudo-lattice form a partially ordered set with respect to the partial ordering relation ($\leq_{EA}$) specified by the directed arcs (lattice or virtual).
- ***Object and attribute concepts:*** All objects, o_i, in the context have a corresponding unique associated object concept in the form $\langle E, E' \rangle$, $E = \{o_i\}$. Similarly, all attributes, a_i, in the context have a corresponding unique associated attribute concept in the form $\langle F', F \rangle$, $F = \{a_i\}$. All object- and attribute concepts are not necessarily in the embedded sublattice. If some are, they form the atoms and coatoms of the embedded-lattice. If they are not, they have only virtual arcs from or to other concepts.
- ***Context preservation:*** An object contains in its upward closure (following lattice or virtual arcs) all the corresponding attribute concepts specified in the incidence relation of the context, and no other attribute concepts. Similarly an attribute contains in its downward closure all its corresponding object concepts specified in the incidence realtion of the context, and no other object concepts.
- ***Unconnected object- and attribute concepts:*** An attribute concept cannot have any outgoing arcs to concepts other than the unit concept and similarly, an object concept cannot have any incoming arcs from concepts other than the zero concept. Object- and attribute concepts are therefore not represented as generalisations or specialisations of each other.
- ***Unique intermediate concepts:*** No two intermediate concepts may have the same extent or the same intent. This property (as well as the above property) implies that any intermediate concept has at least two upward and two downward arcs (virtual or lattice). This does not preclude attribute- and object concepts from having the same extent or intent respectively or sharing the same extent or intent of an intermediate concept (in such cases one of the concepts will have only one parent or child concept). Such attribute or object concepts are represented as distinct concepts in a compressed pseudo-lattice.

- ***Non-empty intent:*** No concept (other than 1_L and 0_L) may have an empty intent or extent (i.e. all objects must possess at least one attribute but some attributes may not have any object possessing the attribute). This limits the contexts for which a valid compressed pseudo-lattice may be constructed. Although the property is not strictly required, the practical benefits of contexts that do not conform to this requirement are not immediately clear. Attribute concepts may have an empty extent.

- ***Embedded sublattice:*** The set of all concepts in the embedded sublattice together with the partial ordering implied by the lattice arcs used in the embedded sublattice, constitute a sublattice when appropriately connected to the implied unit- and the zero concepts (i.e. the supremum and infimum of any pair of concets exits and are unique).

- ***Supremum and infimum:*** Any set, S, of concepts in the embedded sublattice has a supremum in the embedded sublattice itself. Similarly S has an infimum in the embedded sublattice.

- ***Intermediate virtual arcs:*** Intermediate concepts may not be connected to one another via virtual arcs. Their virtual arcs must end in an attribute concept or start at an object concept. This property is implied by the fact that only atoms and coatoms of a sublattice are removed.

- ***Exact representative connection:*** Virtual arcs in a compressed pseudo-lattice are not to arbitrary intermediate concepts and respect the EIR and EER operations. An object concept, o, is only connected via virtual or lattice arcs to EIR(L, Intent(o), o) (where L is the embedded sublattice) and no other concepts. A similar property holds for any attribute a and EER(L, {a}, a). These dual properties are critical in ensuring that the closure operations function as expected (e.g. that the downward closure of a concept contains its extent either via lattice- or virtual arcs).

- ***Arc duplication:*** A concept may only have one arc (either lattice or virtual) to any concept that covers it.

- ***Cover:*** A concept may not have an arc to any other concept to which it is indirectly linked[11].

The compresses pseudo-lattice definition and properties show that a compressed pseudo-lattice is essentially a bipartite graph (virtual arcs) that contains an embedded sublattice (lattice arcs). Furthermore, the compressed pseudo-lattice properties ensure a well-defined and unique structure for a given context and a given sequence of CompressedLattice operations. Various operations can be defined on a compressed pseudo-lattice, but the most important are:

- CompressLattice and ExpandLattice (described in the next section).

- Closure and LatticeClosure, where LatticeClosure follows only lattice arcs when discovering concepts whilst Closure follows any type of arc.

- AddAtom, i.e. insert a new object into the context and embedded sublattice by using a modified incremental lattice construction algorithm that operate under compressed pseudo-lattices.

- InsertVirtualObject, an alternative to AddAtom that does not use a computationally expensive lattice construction algorithm to update the embedded sublattice (refer

[11] Concept x is indirectly linked to concept y iff x has a path to y via one or more intermediate concepts.

to section 6.11). The object is inserted into the compressed pseudo-lattice by simply creating virtual arcs to its exact intent representatives.

6.8 THE EXPANDLATTICE OPERATION

A complementary operation to CompressLattice, namely ExpandLattice, can be defined to enlarge the embedded sublattice of a compressed pseudo-lattice by the insertion of new atoms or coatoms into the embedded sublattice. ExpandLattice essentially recreates concepts removed by CompressLattice. The operation works in a particular direction, starting with a concept that is incident to at least one virtual arc. In general, a concept may be incident to zero, one or more than one virtual arcs. Some virtual arcs may connect the concept to objects while others may connect it to attributes. When invoking the ExpandLattice operation, a 'direction' has to be specified. If the concept is an atom of the embedded sublattice and it has virtual arcs connecting to it, then the direction is designated 'downwards' and the operation will create new atoms in the lattice below the concept. Alternatively, if the concept is a coatom of the embedded sublattice and it has virtual arcs connecting to it, then the direction is designated 'upwards' and will create coatoms above the concept. If the concept is both an atom and coatom, then the direction may be specified as either downwards or upwards.

In the upward direction, starting with concept c, the ExpandLattice operation determines the minimal number of coatoms that must be inserted into the embedded sublattice to replace the virtual arcs from c with lattice arcs to these inserted concepts. Concept c is directly connected to these concepts by lattice arcs replacing c's virtual arcs. To comply with compressed pseudo-lattice properties further generation of concepts and/or creation or removal of arcs may be necessary. Similar remarks apply *pari passu* when expanding a given concept in the downward direction.

Note that the CompressLattice and ExpandLattice operations *are not symmetric* in that the one does not reverse the other. In most instances ExpandLattice does not recreate the concepts removed via a single CompressLattice operation. It is however always possible to completely compress an EA-lattice into a bipartite graph or to use ExpandLattice operations to completely rebuild the EA-lattice from a bipartite graph. Our implementation of this latter series of operations indicates that it is computationally more expensive than using a 'traditional' incremental lattice construction algorithm to construct a lattice but it still indicates the versatility of a compressed pseudo-lattice. The context preservation and exact representative connection properties of a compressed pseudo-lattice play important roles in the ability to rebuild the EA-lattice from a compressed pseudo-lattice.

ExpandLattice is defined below in terms of its pre- and post conditions. Again, parameter names imply their corresponding types.

```
//=====================================================================
Function ExpandLattice(aCompressedLattice, aConcept, aDirection)
            Return outCompressedLattice
//=====================================================================
//Pre-condition: aConcept is a concept in aCompressedLattice that has
//virtual arcs in aDirection.
//Post-condition: outCompressedlattice retains all the concepts and
//arcs of aCompressedLattice, except in the following respects. If
//aDirection is down (up), then the minimal number of new atom
//(coatom) concepts are inserted into outCompressedLattice's embedded
//sublattice to cover aConcept and replace the its virtual arcs with
//lattice arcs. Additional concepts are created and arcs are created,
//removed or relabelled if and only if necessary to maintain
//compressed lattice properties. If appropriate, object (attribute)
//concepts are reconnected to the embedded sublattice via lattice arcs.
//=====================================================================
```

As an ExpandLattice example, consider figure 6.5 with the compressed pseudo-lattices D_0 to D_5. When starting with D_5, i.e. the bipartite graph, the following order of ExpandLattice operations will reconstruct D_0: D_4 = ExpandLattice(D_5, c, Downward); D_2 = ExpandLattice(D_4, e, Downward); D_1 = ExpandLattice(D_2, a, Downward) and finally D_0 is the result of successive ExpandLattice calls that expand the concepts n_1, n_2 and n_3 in a downward direction. Note that ExpandLattice(D_4, e, Downward) does not produce D_3 because D_3 does not contain *all* the atoms created by ExpandLattice needed to replace the virtual arcs to e with lattice arcs (this is a example of CompressLattice and ExpandLattice not being symmetrical).

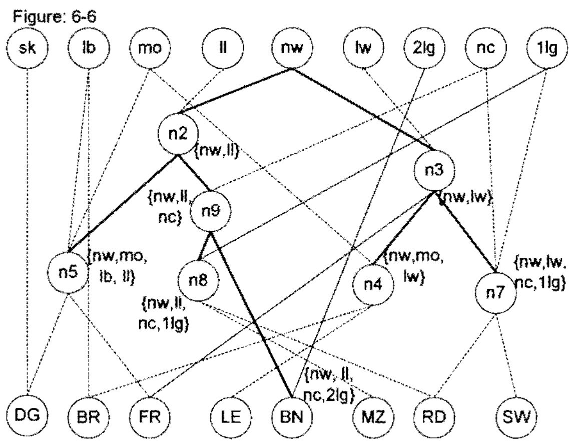
6.9 INTERPRETATION OF COMPRESSED PSEUDO-LATTICES

Since the embedded sublattice of a compressed pseudo-lattice is indeed a sublattice, the interpretation of the concepts in a compressed pseudo-lattice is analogous to that of concepts in a concept lattice. In a concept lattice, concepts are partially ordered in terms of generalisation and specialisation of their intents and extents. A parent concept, p, of a concept, c, is (in a concept lattice) the smallest concept that is more general than c and therefore moving upwards in a lattice involves the smallest increments of generalisation supported by the 'evidence' in the context. In a compressed pseudo-lattice this continues to be the case, except for the fact that concepts that were removed are not 'discovered' or visited due to being 'uninteresting', not useful or insignificant in the application context in which the compressed pseudo-lattice is being used. Algorithms based on compressed pseudo-lattices are therefore not able to discern the relationships between objects that are part of clusters referenced by removed or compressed concepts.

One may argue that this can be detrimental to such algorithms but it should be remembered that classification algorithms based on other structures such as hierarchies (for example Quinlan's (1986) ID3 decision trees and Fisher's (1987) COBWEB) do precisely this: they minimise the clusters or concepts used to describe a context often with greater classification accuracy than using more concepts. This is because at a certain point the additional resolution obtained by using more concepts is used to approximate and describe the noise inherent in the data rather than depicting the abstractions that hold in the larger population from where the data was taken. These approaches have proved successful in many areas of research, particularly in KDD and machine learning. The compressed pseudo-lattice gives the researcher the ability to apply the ideas used in other areas of research on concept lattices whilst still maintaining the benefits of the lattice properties. In essence the removal of concepts restricts the vocabulary of concepts available to KDD or machine learning algorithms in a controlled way without the loss of many desirable features of concept lattices.

One way of viewing the removal of atoms or coatoms from a concept lattice is to see the removal thereof as the change of the attributes and objects of a context. The context is redefined in terms of the new attributes and objects defined by the atoms and coatoms of the embedded sublattice. An embedded sublattice in which attributes are removed can therefore be seen as an abstraction of the context where attributes are more specialised by conjoining some of the original attributes. A removed object is 'replaced' by a more general object, this being the next more general concept in the EA-lattice. The virtual arcs in the compressed pseudo-lattice in these cases indicate and preserve the relationship between the original context and the new implied context with its specialised attributes and generalised objects.

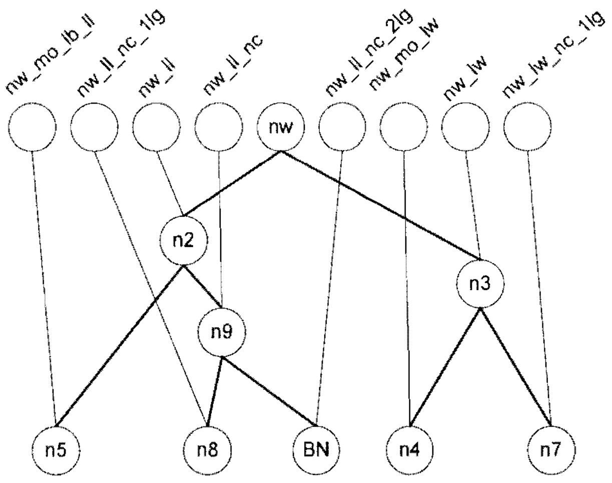
Figure 6.6 is an example in which the EA-lattice of the Living Context has been compressed (or reduced) to a hierarchy (or in ID3 terms a decision tree) that describes the context as a set of objects all of which need water. These objects are then divided into those that live on land (ll) and those that live in water (lw) and so forth. This description or classification of the context is not incorrect - it is just not the *only* description or classification of the Living Context. The use of a compressed pseudo-lattice also has the benefit that even though the embedded sublattice is essentially representing a hierarchy, the fact that there are objects that belong to more than one branch of the hierarchy is easily represented (e.g. concepts FR, RD).



Incidence relation									
	nw	lw	ll	nc	1lg	2lg	mo	lb	sk
LE	✓	✓					✓		
BR	✓	✓					✓	✓	
FR	✓	✓	✓				✓	✓	
DG	✓		✓				✓	✓	✓
SW	✓	✓		✓	✓				
RD	✓	✓	✓	✓	✓				
BN	✓		✓	✓		✓			
MZ	✓		✓	✓	✓				

Figure 6.6: A compressed pseudo-lattice structured to contain a hierarchy that implies the context and EA-lattice in figure 6.7

The embedded sublattice of a compressed pseudo-lattice implies a new context with more specialised attributes and generalised objects. Figure 6.7 shows the incidence relation of the new (implied) context as well as the EA-lattice. Note that the attributes of this new context are now conjunctions of the attributes of the previous context.



Incidence relation									
	nw	nw_ll	nw_mo_lb_ll	nw_ll_nc	nw_ll_nc_1lg	nw_ll_nc_2lg	nw_lw	nw_mo_lw	nw_lw_nc_1lg
n5	✓	✓	✓						
n8	✓	✓		✓	✓				
BN	✓	✓		✓		✓			
n4	✓						✓	✓	
n7	✓						✓		✓

Figure 6.7: The EA-lattice and context implied by the compressed pseudo-lattice in figure 6.6

6.10 WHY COMPRESSED PSEUDO-LATTICES?

'A little knowledge that acts is worth infinitely more than much knowledge that is idle'

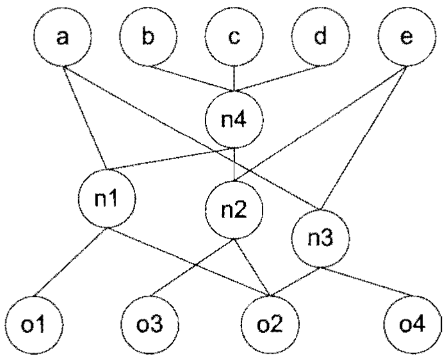
Kahlil Gibran

Key questions surrounding the use of compressed pseudo-lattices are: Why would we remove formal concepts from a formal concept lattice in the first place? Is it not better to work with all formal concepts? The answer to the question lies in the nature of the formal concept lattice: a formal concept lattice contains a concept for all possible clusters of objects supported by the incidence relation (i.e. every possible grouping of objects that have some attributes in common are represented by a formal concept). This results in a data structure that is very large and, in the worst case, exponential in size. In addition, the interpretation and use of this data may be obscured by the large amount of detail (often caused by noise in the data). Authors such as Duquenne et al. (2001) have expressed the difficulty in working with large concept lattices and have called for useful approximations of lattices. Hereth and Stumme (2001) generate Iceberg Concept Lattices in which they have purposefully removed concepts to reduce the lattice size. Iceberg Lattices are a specialisation of compressed pseudo-lattices in the sense that only atoms are removed. Mephu Nguifo (2001) also do not use the whole concept lattice in the context of machine learning. Compressed pseudo-lattices allow one to retain the benefits of a lattice, but allow for the selective and discretionary removal of concepts, thereby reducing the size of the lattice. Should it be required, the EA-lattice can be re-created using the ExpandLattice operation.

Another observation regarding the nature of a concept lattice is that some of the concepts may not, in some sense, represent 'meaningful' or 'useful' clusters of objects. An example is when attributes in a context do not imply each other[12]. For example in the Living Context, the occurrence of the attribute ll always implies the attribute nw.

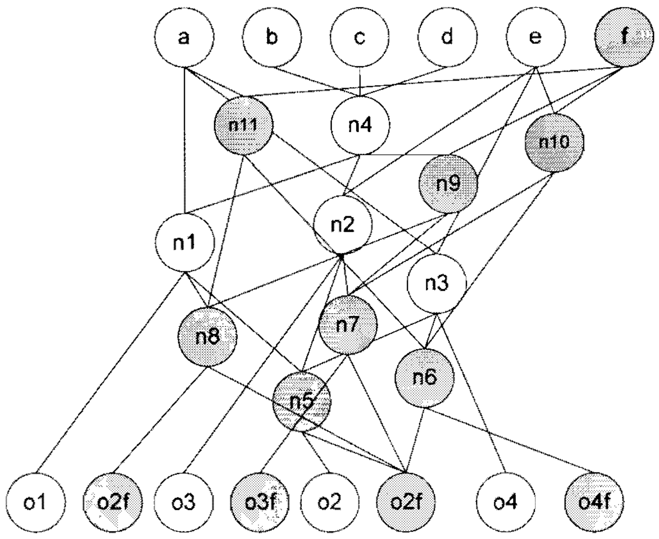
[12] An implication rule is a rule in the form $B \rightarrow C$ where B and C are sets of attributes. The support of a rule is the number of objects in a context for which this rule holds whilst the confidence of the rule is the number of times the rule holds in all objects that have B in their intent. A rule with a confidence of 100% indicates an implication rule.

As another example, consider the concept lattice for the context in figure 6.8 showing a simple context and its EA-lattice. The incidence relation of figure 6.9 is an extension of that of figure 6.8 in that the objects of figure 6.8 have been duplicated and an attribute f, which is not implied by any other attribute(s), introduced in the intent of the duplicated objects. The additional attribute, f, was added to the intent of each of the new duplicated objects so that the new context has pairs of related objects that differ only in respect of one attribute (e.g. o1f has the same intent as o1 except for the additional attribute f). Figure 6.9 shows the EA-lattice of this context. In this context, the attribute f is not implied by any other attribute since any combination of the attributes, a to e, that occurs with f in some set of objects in the context, also occurs without f in some other set of objects in the context.



Incidence relation					
	a	b	c	d	e
o1	✓	✓	✓	✓	
o2	✓	✓	✓	✓	✓
o3		✓	✓	✓	✓
o4	✓				✓

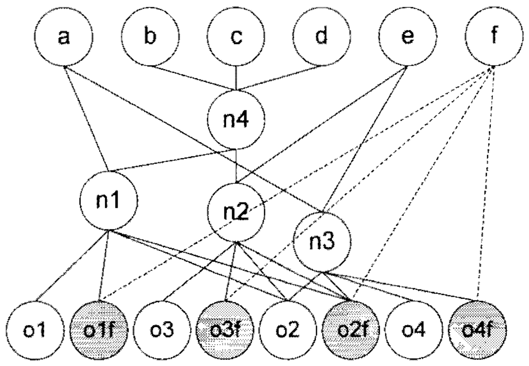
Figure 6.8: The EA-lattice of a simple context



Incidence relation						
	a	b	c	d	e	f
o1	✓	✓	✓	✓		
o2	✓	✓	✓	✓	✓	
o3		✓	✓	✓	✓	
o4	✓				✓	
o1f	✓	✓	✓	✓		✓
o2f	✓	✓	✓	✓	✓	✓
o3f		✓	✓	✓	✓	✓
04f	✓				✓	✓

Figure 6.9: An attribute, f, not implied by any other attribute(s) introduced into the context of figure 6.8 creates a large number of additional concepts

The attribute f is thus not implied by any combination of the other attributes. With regard to deriving implication rules from the lattice, no implication rule based on f is possible and yet the lattice explicitly represent all possible combinations of f and the other attributes in the newly created concepts (newly created concepts are shaded). The effect of adding such an attribute to a context can clearly be seen to significantly increase the number of concepts in the lattice. This is in fact the worst-case example, where the addition of each new attribute (or object) doubles the number of concepts in the lattice. We argue that such structures and attributes are not very useful in machine learning and KDD and are best not represented in the lattices used in these applications.



Incidence relation						
	a	b	c	d	e	f
o1	✓	✓	✓	✓		
o2	✓	✓	✓	✓	✓	
o3		✓	✓	✓	✓	
o4	✓				✓	
o1f	✓	✓	✓	✓		✓
o2f	✓	✓	✓	✓	✓	✓
o3f		✓	✓	✓	✓	✓
04f	✓				✓	✓

Figure 6.10: A compressed pseudo-lattice of the context in figure 6.8 in which the embedded sublattice corresponds to that of figure 6.8

Concept n_6 in the EA-lattice (figure 6.9) has no implication rule associated with it and has a very little support. It is thus doubtful whether it is useful in any KDD or machine learning effort and would therefore worth removing by first using CompressLattice to remove the objects underneath it and then the concept itself. The compressed pseudo-lattice in figure 6.10 shows how the creation of additional concepts can be avoided by using the CompressLattice operation to remove the concepts involving f (i.e. all shaded concepts in figure 6.9) – these relations are not lost and are still indicated by the virtual arcs. The embedded sublattice of this compressed pseudo-lattice corresponds to the EA-lattice in figure 6.8. The objects in the compressed pseudo-lattice therefore still have f in their upward closure. The advantage of a compressed pseudo-lattice is that even tough most of the information is not lost in figure 6.10, should it be required, the ExpandLattice operation can be used to regenerate the lattice in figure 6.9.

This example may seem artificial but in KDD and machine learning, a large number of objects are usually used to construct a lattice (e.g. a training set). If the sample is large enough, statistically most combinations of attributes that are not implied by other attribute(s) will occur in the set of objects. As a result, large number of concepts will be created and parts of the lattice will resemble a Boolean lattice. Even if there are very clear implication rules, if *only one* object does not conform to the rule (i.e. the confidence is not exactly 100% due to noise or errors in the data) its insertion into the lattice will cause the creation of all these additional concepts despite the low support for them. Put differently, only one exception to the rule will cause an otherwise implied set of attributes to loose this property, even though, statistically speaking, there is a high dependence correlation.

A typical approach in KDD and machine learning is to use the number of objects in the extent of each concept as a measure of support for the implication rules on its attributes. It is thus important to keep the objects in the data structure to calculate this measure. Since this is exactly what happens in a compressed pseudo-lattice it is ideal for this purpose.

In the field of KDD and machine learning, the ability to remove concepts from large lattices may prove beneficial in a number of respects. The reduced size of the lattice will improve the efficiency of algorithms whilst the removal of erroneous or noise-induced concepts with a small support may improve the results of such algorithms. A compressed pseudo-lattice based on a suitable compression strategy, offers researchers the ability to reduce the concepts in a lattice to those clusters that most accurately represent the context. This is done by removing clusters closer to the top of the lattice that are too general to allow meaningful classifications whilst also removing concepts that are too specific closer to the bottom of the lattice.

Previously the meet and join operation on a lattice was used to find the concept that represents objects that are relevant to a set of attributes. The introduction of the intent representative sets now allows these to be used as approximate or 'rough' meets in algorithms. This enhances the ability of algorithms to cope with noise in both the data as well as in the query itself (as indicated in the IR examples earlier in this chapter). Clearly, however, the results are dependent on the structure of the database. The purpose of KDD and machine learning in such situations is then to search for or construct a database (sublattice) that best typifies the inherent clusters, rules and implications of a context instead of generating, by brute force, all possible rules and implications that can be derived from a context. To paraphrase Einstein, the lattice should be as simple as possible but not simpler. The benefit of using intent and extent representative sets are that they are defined in terms of discrete operations and behave in a predictable fashion, whereas operations using 'fuzzy' and other approximations often result in a number of anomalies due to their inherently non-discrete nature.

It should be noted that the approaches described above are in general not appropriate for all areas where concept lattices have been used. Some authors (e.g. Wille (2001)) have described numerous case studies where the line diagrams of concept lattices aid human understanding of a context. Clearly compressed pseudo-lattices are less appropriate in these areas of application.

Compressed pseudo-lattices are however an alternate method of supporting *conceptual views* (Wille 2001, 2002). Conceptual views are formed when focussing on a particular part of the context. In this case a certain number of columns (i.e. attributes) in the cross table of the context are selected and the conceptual view is then defined as the lattice formed by only those columns. Each concept in the conceptual view is annotated with the number of objects that are in the extent of the concept, in this way a human is able to focus on a specific domain or view within the context which may assist in finding relevant information. It is easy to see that the lattice for a particular conceptual view can be easily extended using virtual arcs to form the compressed pseudo-lattice for which the conceptual view is the embedded lattice. The advantage of this approach is that it is then easier (from a data rather than a human perspective) relate the conceptual view back to the original concept. In addition, compressed pseudo-lattices are not restricted to sublattices defined by the columns of the cross table. Compressed pseudo-lattices therefore provide a more flexible but still formal way of defining conceptual views.

6.11 COMPRESSION STRATEGIES AND CRITERIA

The start of this chapter defined a very specific domain of discourse of which there are three components: the context, the database and the query operation. A compressed pseudo-lattice may serve as such a database. The separation of the database and query operation creates an interesting deviation from some traditional information retrieval approaches: the organisation of the database co-determines the outcome of the of the query operation. Given a context and a query, the result of the query depends on the compressed pseudo-lattice used to represent the context. The question that arises is thus: 'Are there databases derived from compressed pseudo-lattice databases that, on average, result in better retrieval for the same context and query operations?'

Although a full exploration of this question is beyond the scope of this text, limited experimental results to date suggest that there are indeed better methods of organisation. Specifically, it appears that a database consisting of the EA-lattice or formal concept lattice of a given context need not, in general, be the best database. In many instances significantly compressed EA-lattices performed equally or better, hinting at an amount of redundancy embedded in concept lattices. Further experimentation is required in order to explore compression strategies and concept pruning criteria that are likely to lead to optimal performance in various contexts.

In general, there are a number of possible compression strategies that seem to deserve such exploration. The most obvious is the one stated above where a lattice is compressed up to a specific level of above a support threshold.

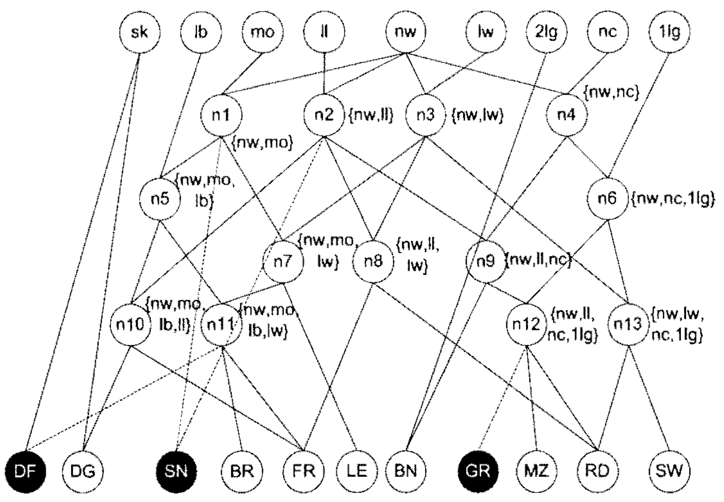
It is useful to have a compression strategy combined with a threshold on the embedded sublattice size. The embedded sublattice is then repeatedly compressed until the embedded sublattice size is below the threshold. This can be combined with an adapted incremental lattice construction algorithm where the pruning mechanism is invoked after each individual object or batch of objects has been inserted into the compressed pseudo-lattice. This has the added advantage of limiting the size of the lattice and therefore the time taken to build a compressed pseudo-lattice.

Compression strategies that have been preliminarily tested use a combination of the following:

- Compress concepts with an extent of size smaller than t and larger than u. This is a more general approach than just excluding concepts at the bottom of the lattice. This approach is taken by Hereth and Stumme (2001). They call the resulting structures iceberg lattices.
- Compression based on the number of arcs to child or parent concepts in the lattice.
- Compression based on $EP(c)$, an estimate of prior probability of the concept c. $EP(c)$ is the number of objects in the extent of c divided by the total number of objects in the context. Refer to Oosthuizen (1994b) for a discussion and examples.
- Compress, based on the difference between an estimate of the expected probability $ExP(c)$[13] and $EP(c)$. This compression strategy performed the best in most preliminary test results.

A useful variation on the insertion of an object into a lattice, that does not require the search for and insertion of the necessary new concepts into a lattice, can be defined as follows. Instead of the creation of concepts, the new object is simply connected to $EIR(L, Intent(o))$ by means of virtual arcs to create a compressed pseudo-lattice. The example in figure 6.11 shows such a compressed pseudo-lattice after the living has been extended with the objects DF, SN and GR. Even though the context and example is relatively simple, it does show that a number of operations were already avoided (e.g. the creation of an intermediate EA-formal concept $\langle\{SN\}, \{mo, nw, lw\}\rangle$). When large lattices are involved, this method of insertion saves much processing. This function is called InsertVirtualObject.

[13] $ExP(c) = EP(a_1) \times EP(a_2) \times \ldots \times EP(a_n)$, where a_i is an attribute in the intent of c. This estimate assumes that the attributes are independent.



Incidence relation									
	nw	lw	ll	nc	1lg	2lg	mo	lb	sk
LE	✓	✓					✓		
BR	✓	✓					✓	✓	
FR	✓	✓	✓				✓	✓	
DG	✓		✓				✓	✓	✓
SW	✓	✓		✓	✓				
RD	✓	✓	✓	✓	✓				
BN	✓		✓	✓		✓			
MZ	✓		✓	✓	✓				
DF	✓	✓					✓	✓	✓
SN	✓		✓				✓		
GR	✓		✓	✓	✓				

Entities/objects	
LE	Leech
BR	Bream
FR	Frog
DG	Dog
SW	Spike-weed
RD	Reed
BN	Bean
MZ	Maize
DF	Dolphin
SN	Snake
GR	Grass

Attributes	
nw	Needs water
lw	Lives in water
ll	Lives on land
nc	Needs chlorophyl
1lg	1 leaf germination
2lg	2 leaf germination
mo	Is motile
lb	Has limbs
sk	Suckles young

Figure 6.11: A compressed pseudo-lattice after the living has been extended with the objects DF, SN and GR by connecting the new objects via virtual arcs to EIR(L, Intent(o)) using the InsertVirtualObject function

In this manner large numbers of objects can be inserted into a concept sublattice for the purposes of KDD or machine learning without incurring the potentially exponential time complexity of the creation of the EA-lattice. Since the support of each concept in both the EA-lattice and the compressed pseudo-lattice is exactly the same, algorithms using these statistical metrics will operate correctly on these lattices.

A related strategy is to insert an object into the lattice by the compressed pseudo-lattice adapted AddAtom algorithm. Upon reaching a threshold on the lattice size, the lattice is compressed to a size below this threshold using a suitable compression strategy and the CompressLattice function. This procedure is repeated for all inserted objects. Since the size of the lattice can be controlled, a compressed pseudo-lattice for a large number of objects can be efficiently constructed. For an appropriate compression strategy this could potentially prevent the degradation of the performance of KDD or machine learning algorithms if it is assumed that 'uninteresting' or 'unnecessary' concepts are not represented in the resulting lattice by being compressed. Note that in the process new concepts would continuously be created by AddAtom. CompressLattice would remove concepts but not necessarily the recently created ones if the compression strategy rates the latter more important or 'meaningful'. In this manner the lattice size remains under control. Experiments have shown that given sufficient data and an appropriate compression strategy, the structure of the embedded lattice stabilise. Different data sets from the same universe with objects presented in different sequences also resulted in substantially similar embedded lattices hinting to the fact that this approach avoids the problems created by hill-climbing searches that are very sensitive to the order in which training sets are presented to these algorithms. This is a topic for further research.

6.12 IMPLEMENTATION AND DISCUSSION OF PRELIMINARY RESULTS

The potential gain in computational efficiency of having a compressed pseudo-lattice should be weighed against the advantages of having a larger and more complete set of concepts available in a particular domain. Preliminary results however suggest that a compressed pseudo-lattice may be a useful generic data structure for various IR (information retrieval) and machine learning problem domains.