Representing Numeric Values in Concept Lattices

John L. Pfaltz

Dept. of Computer Science, Univ. of Virginia
Charlottesville, VA 22904-4740
jlp@virginia.edu

Abstract. Formal Concept Analysis is based on the occurrence of symbolic attributes in individual objects, or observations. But, when the attribute is numeric, treatment has been awkward. In this paper, we show how one can derive logical implications in which the atoms can be not only boolean symbolic attributes, but also ordinal inequalities, such as $x \leq 9$. This extension to ordinal values is new. It employs the fact that orderings are antimatroid closure spaces.

1 Extending Formal Concept Analysis

Formal Concept Analysis (FCA), which was initially developed by Rudolf Wille and Bernhard Ganter [3], provides a superb way of describing "concepts", that is closed sets of attributes, or properties, within a context of occurrences, or objects. One can regard the concept as a closed set of objects with common attributes. Frequently clusters of these concepts, together with their structure, stand out with vivid clarity. However, two unresolved problems are often encountered. First, when concept lattices become large, it is hard to discern or describe significant clusters of related concepts. Gregor Snelting used formal concept analysis to analyze legacy code [6, 14].[1] Snelting's goal was to reconstruct the overall system structure by determining which variables (attributes or columns) were accessed by which modules (objects or rows). It was hoped that the concept structure would become visually apparent. Unfortunately, the resulting concept lattice shown on page 356 of [6] is little more than a black blob. Visual interpretation of closure concepts does not seem to scale well.

Second, when the attribute values are numeric, as is often the case with technical data, formal concept analysis becomes difficult. It is easy to comprehend the set of all objects whose color is "red"; But what precisely constitutes the set of objects whose weight is "9.5", or "near 9.5", or "less than 9.5" or otherwise similar to "9.5". Fuzzy set theory [5, 18] might be appropriate here, but we prefer a deterministic approach.

To cope with the first issue, our approach has been to work with the logical implications that can be deduced from adjacent closed concepts in the lattice, rather than to seek clusters of such concepts. So long as all the attributes are boolean this seems to work rather well. We have applied formal concept analysis to relations (contexts) of sizes $8,124 \times 85$ and $1,272 \times 144$. The resulting closed set lattices were large, 104,104 and 1,804 nodes respectively, but nevertheless we were able to extract useful logical implications. In Section 3 we briefly review some of this older work.

In Section 4 we will develop our new approach to representing numeric predicates. Very briefly, it will consist of viewing a concept, not as being a closed set of attributes associated with a set of objects, but rather regarding it as a closed set of "closed sets" associated with the objects.

[1] Siff and Reps [13] published shortly after.

A crucial contribution of FCA has been to establish the role and importance of closure operators and closed sets in mathematical thinking. In Section 2 we review those closure concepts that we will need later.

2 Closure Operators and Closure Spaces

The notion of "closure" plays a major role in our representation of the real world. In particular we will be concerned with closed sets of objects, closed sets of predicates and closed sets of numbers.

2.1 Closure Concepts

By a "closure system" over a "universe" $\mathbf{U}$, we mean a collection $\mathcal{C}$ of sets $X, Y, \ldots Z \subseteq \mathbf{U}$, including $\mathbf{U}$, satisfying the property that if $X, Y \in \mathcal{C}$ then $X \cap Y \in \mathcal{C}$. The sets of $\mathcal{C}$ are said to be the closed sets of $\mathbf{U}$. As an alternative to this "intersection" characterization, one can define a closure operator $\varphi : 2^{\mathbf{U}} \rightarrow 2^{\mathbf{U}}$ satisfying the following 3 axioms for all X, Y, Z:

$X \subseteq X.\varphi$,

$X \subseteq Y$ implies $X.\varphi \subseteq Y.\varphi$

$X.\varphi.\varphi = X.\varphi$.

(For technical reasons we prefer to use suffix operator notation, so read $X.\varphi$ as "X closure".) Readily, a set X is closed in $\mathcal{C}$, if $X.\varphi = X$. The equivalence of these two alternative definitions is well known [8], and we will use both in the following sections.

Closure systems can satisfy many other axioms, and those that do give rise to different varieties of mathematical systems. If $(X \cup Y).\varphi = X.\varphi \cup Y.\varphi$ we say φ is a topological closure. If the system satisfies the "exchange axiom", that is if $p, q \notin X.\varphi$ but $q \in (X \cup \{p\}).\varphi$ then necessarily $p \in (X \cup \{q\}).\varphi$, the system can be viewed as a kind of linear algebra, or more generally a "matroid". The Galois closure we will be using in this section satisfies neither of these additional axioms. But later in Section 4, we will be using "antimatroid" closure operators, that is those which satisfy the "anti-exchange axiom" if $p, q \notin X.\varphi$ and $q \in (X \cup \{p\}).\varphi$ then $p \notin (X \cup \{q\}).\varphi$.

2.2 Galois Closure and Concept Lattices

The approach we will follow is similar to that was first developed by Rudolf Wille and is best presented in [3]. Formal concept analysis begins with a relation R between two sets, say a set $\mathcal{O}$ of objects and a set $\mathcal{P}$ of object predicates, or attributes. Using standard relational terminology, each object $o_j \in \mathcal{O}$ can be regarded as a row in R and each predicate $p_k \in \mathcal{P}$ is a column. Each attribute p_k is a binary, logical property, *i.e.* $p_k(o)$ is either true or false, because the object exhibits property p_k or it doesn't. A concept C_n is a pair of closed subsets $C_n = (O_n, P_n)$ where $O_n \subseteq \mathcal{O}$, $P_n \subseteq \mathcal{P}$ with the property T that for every $o_i \in O_n$, every $p_k \in P_n$, $p_k(o_i)$ is true. Each concept is assumed to be maximal, that is for the set O_n there is no larger subset $P'_n \supset P_n$ satisfying property T, and for P_n there is no larger subset $O'_n \supset O_n$ satisfying T.

The collection $\mathcal{C}$ of all concepts C_n, so defined, forms a closure system; that is, the intersection of any two concepts in $\mathcal{C}$ is a concept. Consequently, the collection $\mathcal{C}$ of concepts forms a lattice when partially ordered by containment with respect to the predicate sets P_n. [2] If we start with the relation

[2] Ganter and Wille prefer to order with respect to object set containment yielding the dual lattice, *c.f.* page 20 [3].

R of Figure 1(a) we obtain the concept lattice $\mathcal{L}$ shown as Figure 1(b).[3] Each node is labeled with a

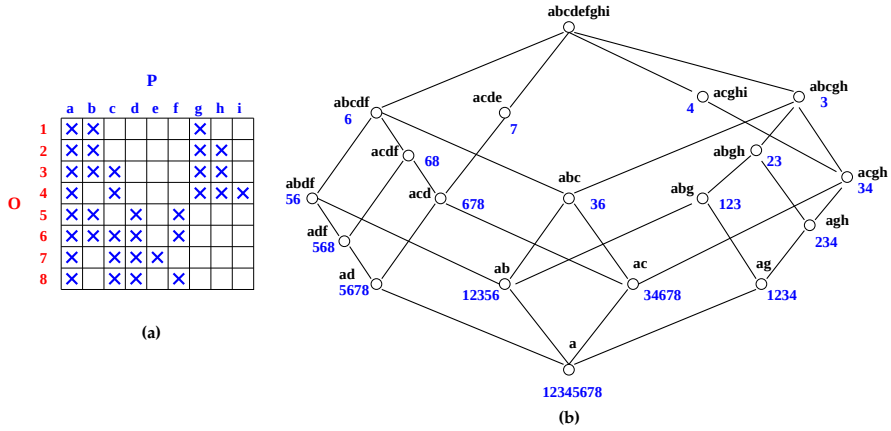

Fig. 1. A relation R and its corresponding concept lattice $\mathcal{L}$

closed set of predicates and a closed set of objects satisfying those predicates, or properties. These two closed sets constitute the concept pair. For example, the combination of properties *adf* is found in rows, or objects, 5, 6 and 8.

In the case of FCA, the closure operator is a Galois closure between $\mathcal{O}$ and $\mathcal{P}$, and has been well studied [1, 2, 7] In this paper we will emphasize the closure aspect as denoted by φ, rather than the concept aspect developed in [3].

2.3 Closed Sets, Generators and Logical Implication

Let C denote a closed set. Then there is some set $A \subseteq C$ such that $A.\varphi = C$. If A is a minimal such set, w.r.t. set inclusion, we call it a generator of C denoted by $C.\gamma$, or by $A \rightarrow C$ [8]. The latter symbolism is not accidental. If φ is a Galois closure, then closed set generation and logical implication are identical [9].

Whenever the closure operator is antimatroid, there is a unique generating set for any closed set [8]. This is the property we will need in Section 4. This need not be true in the case of a Galois closure. For example, the generating sets of *abcgh* in Figure 1(b) are *bcg* and *bch*, which can be expressed logically as

$$bcg \vee bch \rightarrow abcgh.$$

We use concatenation as a short hand to express set enumeration, and when the elements are logical predicates to express conjunction. Similarly, we often suppress the dependent variable. One could expand the expression above to

$$(\forall o \in \mathcal{O})[(b(o) \wedge c(o) \wedge g(o)) \vee (b(o) \wedge c(o) \wedge h(o)) \rightarrow a(o) \wedge b(o) \wedge c(o) \wedge h(o)]$$

which seems to be unnecessarily unwieldy. Note, that the universal quantification is restricted to $\mathcal{O}$, the set of observed objects.

[3] This figure is taken directly from [3].

The issue now becomes: "given a lattice of closed concepts, such as Figure 1(b), how does one derive the generating sets?" For example, what is the generator of the closed set *acde* in Figure 1(b)? In [4] it is shown that

Theorem 1. *Let C be a closed set with respect to φ. Let $\mathcal{F}$ denote the family of difference sets $D_k = C - C_k$ where C_k are the closed sets covered* [4] *by C in $\mathcal{L}_\varphi$. A set $C.\gamma \subseteq C$ is a generator of C if and only if $C.\gamma \cap D_k \neq \emptyset$ for all $D_k \in \mathcal{F}$.*

With this theorem, we can construct the generator of any closed set C as a combination of elements $e_i \in D_i = C - C_i$ where C covers C_i in $\mathcal{L}$. The node *acde* covers only one node *acd* in the lattice $\mathcal{L}$, so $C - C_1 = \{e\}$. Thus, using the result above we can show that the generator $acde.\gamma = \{e\}$, or $e \rightarrow acde$. That this is true is evident from Figure 1(a). Property e is only found in observation 7, where a, c, and d are also seen. So the property e, in this case, trivially implies properties a, c and d. In the case of *abcgh*, $D_1 = \{gh\}$, $D_2 = \{c\}$ and $D_3 = \{b\}$, so *bcg* and *bch* must both be generating sets.

The intuitive understanding of Theorem 1 is simple. Let C be a closed set covering C_k. Readily we cannot have $C.\gamma \subseteq C_k$ else $C.\gamma.\varphi = C_k$. So $C.\gamma$ must have at least one element e_k that is "not in" C_k to insure $C_k \subset C$. The union of these e_k is sufficient to generate a closed set strictly larger than all the C_k. And, since C covers all of the C_k, this closed set must be C. The negative motifs "not in" and "not contained" will be repeated in Section 4. With this "covering" result, the derivation of generating sets is an inexpensive, local construction whose details can be found in [11].

2.4 Path Closures in Partially Ordered Sets

Recall that a set S is partially ordered by the relation $\leq$ if it is reflexive, weakly anti-symmetric and transitive.[5] Partially ordered sets (posets) are often represented as acyclic directed graphs, such as Figure 2, which we illustrate left to right like a number line rather than up and down in a Hasse diagram.[6]

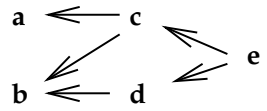

Fig. 2. A poset on 5 points.

Given any poset there are three naturally associated path closures defined
$Y.\varphi_L = \{x | x \leq y, y \in Y\}$
$Y.\varphi_R = \{z | y \leq z, y \in Y\}$
$Y.\varphi_C = \{y | y_1 \leq y \leq y_2, y_1, y_2 \in Y\}$.
These path closures have been variously called "downset" or $\downarrow$, "upset" or $\uparrow$, or "convex" closures respectively. In the case of Figure 2, these three path closures respectively yield the closed sets

[4] C covers C_k if we do not have a closed C' such that $C_k \subset C' \subset C$.
[5] It is customary to denote partial order relations with $\leq$ rather than R.
[6] Since $x \leq x, \forall x$, one could add loops at all vertices (points), but these are customarily omitted.

$\mathcal{C}_L = \mathcal{C}_{\downarrow} = \{\emptyset, a, b, ab, bd, abc, abcd, abcde\}$
$\mathcal{C}_R = \mathcal{C}_{\uparrow} = \{e, ce, de, cde, ace, acde, bcde, abcde\}$
$\mathcal{C}_C = 2^{\{abcde\}} - \{ae, be, abe, ade, bce, bde, abce, abde\}$
In the case of convex closure it is easier to enumerate the eight subsets that are not closed, *e.g.* not convex.

It is shown in [8] that these path closures on posets are antimatroid. Consequently all closed sets have unique generators. For example, consider downset closure $\downarrow$ on Figure 2. The closed sets have the indicated generating sets.

$\mathcal{C}$	$\emptyset$	a	b	ab	bd	abc	abcd	abcde
$\mathcal{C}.\gamma$	$\emptyset$	a	b	ab	d	c	cd	e

3 Logical Implication in Large Concept Lattices

In Section 1 we said that by interpreting a concept lattice, $\mathcal{L}$, as a set of logical implications rather than trying to analyze it visually, we could begin to cope with large lattices. We briefly review here the logical analysis of two such lattices, of 104,104 and 1,804 nodes each. Details can be found in [11, 12] and [10]. Our purpose is to reinforce the practicality of extracting generating sets (the premise(s) of the implications) using a local covering algorithm based on Theorem 1 of Section 2.3.

According to the "Audubon Society Field Guide of North American Mushrooms" mushrooms can be characterized with respect to 22 physical attributes, including "cap shape" and "cap surface". Cap shape can take the form of "bell", "conical", "flat", "knobbed", "sunken" or "convex" for 6 distinct logical attributes which we encode as $b1, c1, f1, k1, s1$ and $x1$. Similarly a cap surface can be "fibrous", "grooved", "smooth" or "scaley", which were encoded as $f2, g2, s2, y2$. By encoding the possible logical predicates for each of the 22 physical attributes, we get 85 distinct logical predicates characterizing the 8,125 different species of North American mushrooms.

The complete lattice of mushroom data has 104,104 nodes, or closed concepts. And, since each closed concept has at least one, but often many, generators we have many more logical implications — almost all of which are totally useless. But, it is not too hard to extract "useful" implications from this mass. For example, a simple AWK script that searches for the predicate $p0$ (poisonous) in the closed set (consequent) and a singleton predicate for the generator (premise) yields implications such as:

$$g2 \rightarrow p0, w3, t4, n5, f6, w7, n8$$

or, "don't eat mushrooms with grooved cap surfaces". Expanding the AWK script to retrieve similar implications, but with exactly two predicate generators yields, for example:

$$c1, y2 \rightarrow p0, n5, f6, w7, n8$$

or, "avoid those with conical, scaley caps".

An important concept in science is that of deterministic causality in which the occurrence of some event, or conjunction of events, must necessarily "cause" a consequent event. Indeed, this was the holy grail of Newtonian physics and much of 19^{th} century science. "Causality" implies necessity, or logical implication. But, it also involves a temporal aspect. The consequent event must temporally follow all assumed antecedent events. One arena where we expect deterministic causality is software execution.

For this application we secured trace data detailing all method invocations in the transaction management module of the open source *Jboss* 1.4.2 statistical package. My colleagues, JinLin Yang

and David Evans [17] had instrumented this module and generated 1,227 trace sequences consisting of 498,489 module invocations, of which 144 were distinct. Many invocations are repeatedly executed in a single trace. These yielded a $1,227 \times 144$ relation, or context, with a resultant lattice of 1,804 nodes and again many more logical implications.

These were run agains a rather simple temporal filter which ensured that all invocations in the premise (generator) preceded every invocation in the consequent (closed set) *in every trace*. This process yielded 43 likely causal dependencies of which

$17 \Rightarrow 22, 23$

$46 \Rightarrow 47, 48, 49, 60, 62$ [7]

are two representative examples. All appear to be true dependencies.

Our goal in recounting these two examples of past results is simply to establish (1) that software exists to process rather large binary relations, (2) that using Theorem 1 is a practical way of determining generator sets and (3) that considering the lattice to be a source of logical implications can have practical value. The software, which employs an incremental approach to creating the lattice, that we believe was first proposed by [15, 16], is available from the author.

4 Numeric Values

Formal Concept Analysis, as well as our own development to date, has been focused on binary predicates. Either a predicate p is true for a given object/observation, or it is not. This made it easy to reason about sets of true predicates. However, much of our understanding of the natural world is numeric. We count and we measure.

Every boolean algebra, or lattice $\mathcal{L}$, is a closure system because $x, y \in \mathcal{L}$ implies $x \wedge y \in \mathcal{L}$. In particular, any predicate $p = 0$ or 1 is a trivial boolean algebra, or closure system $\mathcal{C}_p$. Each element, 0 or 1, is its own generator. We achieve far greater expressive power if we let each predicate p_k of an observational tuple $(p_1, \ldots, p_n)$ be the generator of a closed set in a closure system $\mathcal{C}_k$, where $\mathcal{C}_k$ can be more complex than just $\{0, 1\}$. If each closure system $\mathcal{C}_k$ is antimatroid, as defined in Section 2.1, then every closed set has a unique generator [8]. Consequently, if the tuple of predicates $(p_1 \ldots, p_n)$ is a tuple of generators, then $(p_1 \ldots, p_n)$ denotes a unique closed set in the n-fold direct product $\mathcal{C}_1 \ldots, \mathcal{C}_n$.

It will be easier to understand the theory we are about to develop if we first consider a concrete example. Figure 3(a) has been shamelessly copied from ([3], p.44). It summarizes the ratings of 14 monuments on the *Forum Romanum* by different travel guides. Here, B = Baedecker, G = Les Guides Bleus, M = Michelin and P = Polyglott.

The 14 numeric 4-tuples can be partially ordered in the usual way, that is $(x_1, x_2, x_3, x_4) \leq (y_1, y_2, y_3, y_4)$ if and only if $x_k \leq y_k, k = 1, \ldots 4$. Figure 3(b), in which individual tuples have been prefixed with a letter denoting the monument giving rise to that tuple, illustrates this ordering. In order to make it a closure lattice, whenever two tuples are covered by a common tuple, their "intersection tuple" has been entered into the order, as required in any well-formed closure system. These intersection tuples have been underlined for emphasis.

Consider the tuple $n : (0, 2, 2, 1)$ which covers the tuples $(0, 1, 2, 1)$ and $(0, 2, 2, 0)$ in $\mathcal{L}$. Recall from Section 2.3 that the generators of a set are determined by the sets it covers. We claim that the generators of $n : (0, 2, 2, 1)$ are $(G > 1) \wedge (P > 0)$. Moreover, we claim that the implication embodied

[7] In a preprocessing step, each method invocation was uniquely identified by an integer, 1 through 144

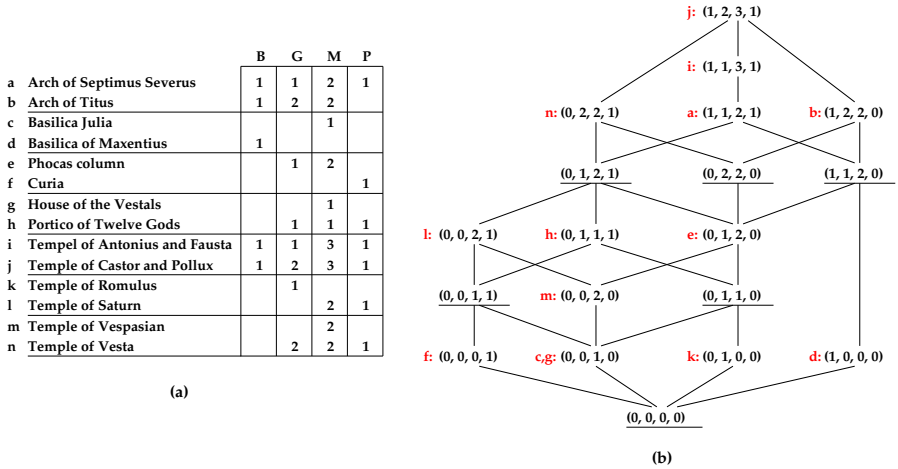

Fig. 3. (a) Ratings of Roman monuments by 4 guide books, (b) Lattice of closed sets implicit in (a).

by this closed set is $[(G > 1) \wedge (P > 0)] \rightarrow [(M \geq 2) \wedge (B \geq 0)]$. First, verify that in Figure 3(a) this implication is in fact true, with j and n being confirming instances.

The principle behind this derivation, which was sketched in Section 2.3 and rigorously proven for sets in [4], is to show "what constitutes the difference between this closed set and the closed sets it covers". In our case, the only difference between $(0, 2, 2, 1)$ and $(0, 1, 2, 1)$ is that "G is not ≤ 1", or equivalently "G is > 1". Similarly, since P cannot be ≤ 0, we must have $P > 0$. The conclusion, *i.e.* those predicates with no differences, could be $B = 0$ and $M = 2$. But, what we really know is that $\neg(B < 0)$ and $\neg(M < 2)$; so $B \geq 0$ and $M \geq 2$ represent the best inferences. The adjacent tuple in Figure 3(b) is $a : (1, 1, 2, 1)$ which covers the tuples $(0, 1, 2, 1)$ and $(1, 1, 2, 0)$. Thus the same reasoning yields $[\neg(B \leq 0) \wedge \neg(P \leq 0)] \rightarrow [\neg(G < 1) \wedge \neg(M < 2)]$ or equivalently $[(B > 1) \wedge (P > 1)] \rightarrow [(G \geq 1) \wedge (M \geq 2)]$ which is verified by observations a, i and j.

Figure 4 provides a rather typical relationship between numeric data that is a bit more complex

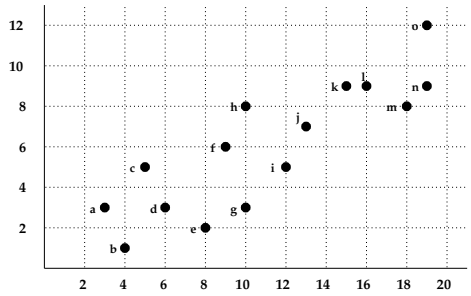

Fig. 4. A semi-random distribution of points

than Figure 3(a). We will again use downset closure, $\leq$, to order both the x and y values.

The resultant closure lattice is shown in Figure 5(b). Using the same logic that we have described

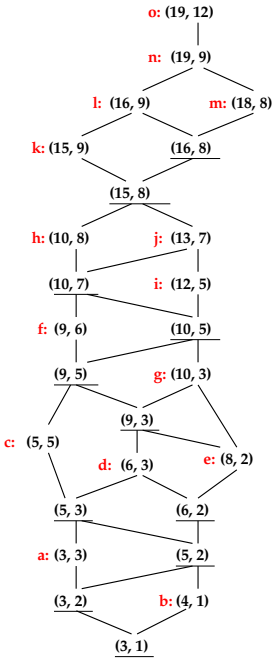

Fig. 5. The closure lattice corresponding to the points of Figure 4.

above, we can assert that $y > 3 \rightarrow x \geq 5$, based on node c; that $x > 10 \rightarrow y \geq 5$, based on node i (12,5) covering (10,5), among many others. These can be verified in Figure 5(a). We have begun to reason about the relationship between x and y.

In both the two preceding examples we used downset closure to define the numeric closure spaces. We can be more creative. For our next example we consider the same semi-random distribution of points of Figure 4, but consider 3 numeric closure spaces, specifically x with $\varphi_{\downarrow}$, y with $\varphi_{\downarrow}$ *and* y *with* $\varphi_{\uparrow}$. Thus the point i is encoded as $(12, 5, 5)$. A node $p = (x_p, y1_p, y2_p)$ is less than $q = (x_q, y1_q, y2_q)$ if $x_p \leq x_q$, $y1_p \leq y2_q$ and $y2_p \geq y2_q$. Thus $i \wedge m$ is not i as before, but rather $i \wedge m = (12, 5, 5) \wedge (18, 8, 8) = (12, 5, 8)$.

Figure 6 illustrates the resultant closed set lattice. The original points from Figure 4 are emboldened. Many of the intersection nodes are labeled with respect to these original data points. The zero element is $(3, 1, 12)$ or (min, min, MAX), while the *supremum*, or one element, is (MAX, MAX, min). We have indicated it in just this way because our software requires such a virtual supremum as its initial condition.

We observe that this lattice has a very different shape than that of Figure 5. One can see various substructures emerging within it; but nevertheless it is reaching the limit of visual comprehension.

However, we can still reason about these numeric values. For instance the node $(12, 5, 8) = i \wedge m$ covers nodes $(10, 5, 8)$ and $(12, 5, 9)$. Consequently $\neg(x \leq 10)$ and $\neg(y \geq 9)$ else $(12, 5, 8)$ could not

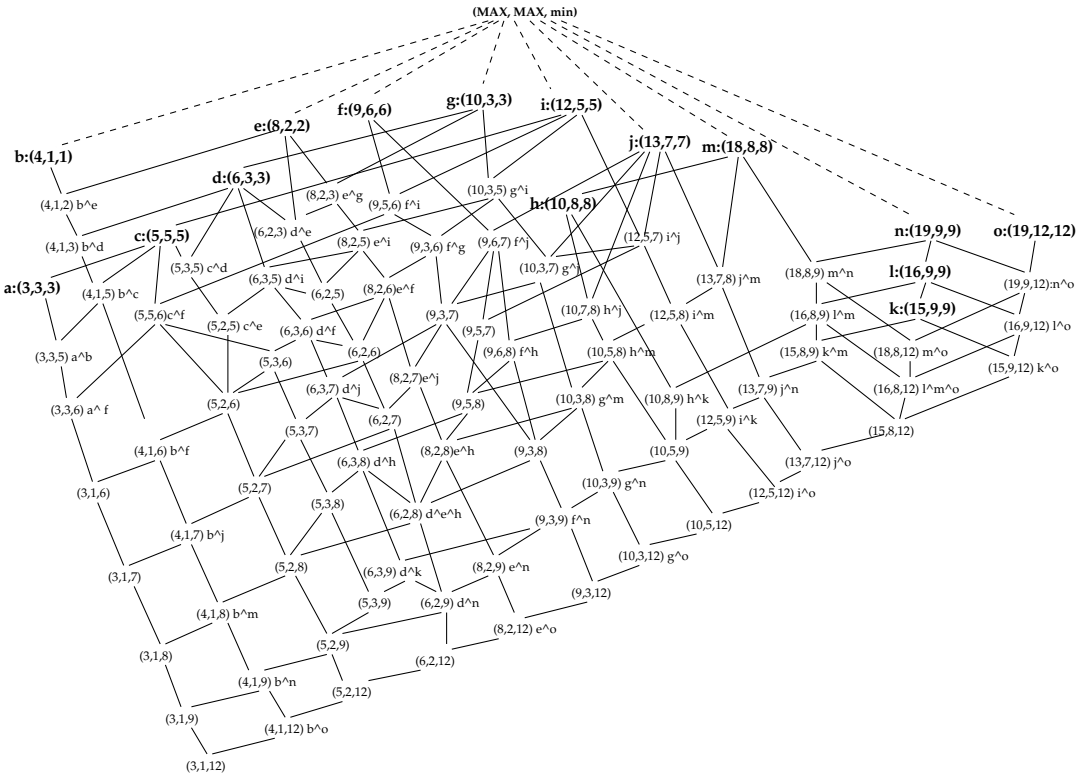

Fig. 6. The lattice of closed sets in Figure 4.

cover both nodes. Thus we know that $x > 10$ and $y < 9$ constitute the generator of $i \wedge m$, which together imply that $y \geq 5$. Inspection of Figure 5(a) shows that this describes the elements i, j, and m. The node $(16, 8, 12) = l \wedge m \wedge o$ covers only the single node $(15, 8, 12)$. If $\neg(x \leq 15)$, then because $(16, 8, 12)$ is closed we must have $y \geq 8$ and $y \leq 12$), or more compactly, $(x > 15) \rightarrow (8 \leq y \leq 12)$. This encompasses the elements l, m, n and o.

5 Conclusions

The substitution of closed sets from an antimatroid closure space for some, or all, binary predicates in FCA is very new. We are not confident that we fully understand this process. For example, we have not yet written software that will automatically produce the lattice of closed concepts nor their generators. Figures 5 and 6 were tediously produced by hand, and even now may still contain errors. However, a method that allows us to treat numerical inequalities as boolean predicates offers considerable promise. We can imagine uncovering from legacy code rules such as:

$$\text{if not } \mathit{get\_lock} \text{ and } \mathit{ntries} > 10 \text{ then } \mathit{abort} \quad .$$

Undirected data mining that can yield assertions such as

$$\text{if } 5 < x < 13 \text{ then } -4 \leq y \leq 0$$

from raw numeric input could bound the error of imprecise observation.

We have applied the antimatroid path closures to totally ordered integer (or real) data sets, but they are equally defined for all partial orders. This opens the potential of using this technique to analyze the behavior of concurrent computer systems, or biological systems in which the expression of phenomena can occur along possibly multiple paths. None of this is yet reality; but the potential for greatly expanding the application of Formal Concept Analysis is here.

References

1. Gabriele Castellini. *Categorical Closure Operators*. Birkhauser, Boston, 2003.
2. Klaus Denecke, Marcel Erné, and Shelly L. Wismath. *Galois Connections and Applications*. Kluwer Academic, Dordrecht, 2004.
3. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis - Mathematical Foundations*. Springer Verlag, Heidelberg, 1999.
4. Robert E. Jamison and John L. Pfaltz. Closure Spaces that are not Uniquely Generated. *Discrete Appl Math.*, 147:69–79, Feb. 2005. also in Ordinal and Symbolic Data Analysis, OSDA 2000, Brussels, Belgium July 2000.
5. Richard Lee. Fuzzy logic and the resolution principle. *J. of the ACM*, 19(1):109–119, Jan. 1972.
6. Christian Lindig and Gregor Snelting. Assessing Modular Structure of Legacy Code Based on Mathematical Concept Analysis. In *Proc. of 1997 International Conf. on Software Engineering*, pages 349–359, Boston, MA, May 1997.
7. Oystein Ore. Galois Connexions. *Trans. of AMS*, 55:493–513, 1944.
8. John L. Pfaltz. Closure Lattices. *Discrete Mathematics*, 154:217–236, 1996.
9. John L. Pfaltz. Incremental Transformations of Lattices: A key to Effective Knowledge Discovery. In A. Corradini, H. Ehrig, H-J Kreowsik, and G. Rozenberg, editors, *First Intern. Conf. on Graph Transformation*, volume Springer Verlag LNCS # 2505, pages 351–362, Barcelona, Spain, Oct. 2002.
10. John L. Pfaltz. Using Concept Lattices to Uncover Causal Dependencies in Software. In B. Ganter and L. Kwuida, editors, *Proc. Int. Conf. on Formal Concept Analysis, Springer LNAI #3874*, pages 233–247, Dresden, Feb. 2006.
11. John L. Pfaltz and Christopher M. Taylor. Closed Set Mining of Biological Data. In *BIOKDD 2002, 2nd Workshop on Data Mining in Bioinformatics*, pages 43–48, Edmonton, Alberta, July 2002.
12. John L. Pfaltz and Christopher M. Taylor. Concept Lattices as a Scientific Knowledge Discovery Technique. In *Workshop on Discrete Mathematics and Data Mining, 2nd SIAM International Conference on Data Mining*, pages 65–74, Arlington, VA, Apr. 2002.
13. Michael Siff and Thomas Reps. Identifying Modules via Concept Analysis. In *Intn'l Conf. on Software Maintenance*, pages 170–179, Bari, Italy, Oct. 1997.
14. Gregor Snelting and Frank Tip. Reengineering Class Hierarchies Using Concept Analysis. In *Proc. ACM SIGSOFT 6th International Symposium on Foundations of Software Engineering, FSE-6*, pages 99–110, Lake Buena Vista, FL, 1998.
15. Petko Valtchev, Rokia Missaoui, and Robert Godin. A Framework for Incremental Generation of Frequent Closed Itemsets. In Peter Hammer, editor, *Workshop on Discrete Mathematics & Data Mining, 2nd SIAM Conf. on Data Mining*, pages 75–86, Arlington, VA, April 2002.
16. Petko Valtchev, Rokia Missaoui, Rouane Hacene, and Robert Godin. Incremental Maintenance of Association Rule Bases. In *Proc. Workshop on Discrete Mathematic and Data Mining*, San Francisco, CA, 2003.
17. Jinlin Yang, David Evans, Deepali Bhardwaj, Thirumalesh Bhat, and Manuvir Das. Terracotta: Mining Temporal API Rules from Imperfect Traces. In *28th Internl. Conf. on Software Engineering (ICSE 2006)*, Shanghai, China, May 2006.
18. L. A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.