Incremental Transformation of Lattices: A Key to Effective Knowledge Discovery*

John L. Pfaltz

Dept. of Computer Science, Univ. of Virginia
{jlp,cmt5n}@virginia.edu

Abstract. We show that a formal concept lattice $\mathcal{L}$, with explicit generators, constitutes a viable medium for discrete, deterministic, data mining. All implications of interest can be found and extracted from $\mathcal{L}$ independent of the frequency of their occurance.
More importantly, we show that these lattices can be grown from a binary relation R one row, or observation, at a time using a transformation that is based on the mathematical properties of the generators and faces of closed sets. Incremental growth is orders of magnitude faster than existing methods which rely on global closure operators.

1 Introduction

In [17], we first presented the ideas: (a) that concept lattices represent a viable way of representing deterministic logical implications based on a set of observations, and (b) that the concept lattices should evolve through a sequence of "local" lattice transformations. In that paper we could illustrate that such transformations were indeed local in nature, but we had no effective algorithm for performing it. Since then, we have developed the necessary mechanisms to make graph transformation (or in this case lattice transformation) an important new approach to knowledge discovery, which we call discrete, deterministic data mining, or DDDM. We believe that DDDM is closely analogous to classical scientific empiricism.

We begin by contrasting our concept of DDDM with more traditional data mining approaches. The basic goal of "data mining" has been the discovery of frequent, or common, attribute associations in very large data sets such as point-of-sale data [2,22,24]. These associations are statistical in nature; we have for example the old favorite "people who buy hot dogs often buy catsup". They may also be evanescent. A frequent association this month may not be frequent next month. The association may be indicative of a causal relationship, but most often it is not. Surely the purchase of hot dogs does not "cause" a purchase of catsup. Frequent associations, whether temporary or not, can be very important to our understanding of the world around us. But, we would argue it is not for the most part "scientific empiricism". Classical science seeks the discovery

* Research supported in part by DOE grant DE-FG05-95ER25254.

A. Corradini et al. (Eds.): ICGT 2002, LNCS 2505, pp. 351–362, 2002.

and understanding of causal relationships among world phenomena that can be expressed with logical precision.

Like all data mining algorithms, the raw data of DDDM is a binary relation $R : O \rightarrow A$. We denote the domain (rows) of this relation as O, which we regard as "observations", or observations of "objects"[1]. By A, the codomain (columns) of R, we mean those attributes, or properties, that may be observed, *i.e.* that an object may exhibit.

Now, let O denote any universe of interest. Our paradigm of scientific knowledge is an assertion of the form $(\forall o \in O)[P(o) \rightarrow Q(o)]$ where $P(o)$ and $Q(o)$ denote predicate expressions over the bound variable o. Readily, there are other forms of scientific knowledge; but a focus on logical implication subsumes most causal assertions.

This concentration on deterministic causal implications appears to be new, even though other authors have suggested many of its elements. The lattice transformation paradigm [19] which treats its input as a possibly unbounded stream of observations appears to be completely new.

2 Closure, Concepts and Implication

DDDM is based on formal concept analysis [7] and closure theory [16]. These tools have been used before in data mining. For example, Zaki uses closure concepts in [24] to minimize the occurance of redundant associations; and both Godin and Missaoui [8] and Pasquier *et al.* [15] use concept lattices to optimize an *a priori* like search.

As developed in [7], formal concept analysis relies on visual display to reveal structure. But this becomes problematic with more than 30 concepts. One can no longer "see" the structure. Lindig and Snelting [10] discovered this when they applied formal concept analysis to code re-engineering. We make no use of visual display techniques, except for illustrative examples.

Where our approach differs from these earlier works is our emphasis on the "generators" of closed sets. Let φ denote a closure operator, so φ satisfies the 3 standard closure axioms for all X, Y,

$$X \subseteq X.\varphi,$$
$$X \subseteq Y \text{ implies } X.\varphi \subseteq Y.\varphi, \text{ and}$$
$$X.\varphi.\varphi = X.\varphi.$$

(We use a suffix notation, so read $X.\varphi$ as "X closure".) A set Z is closed if $Z.\varphi = Z$. Let Z be a closed set; a set $X \subseteq Z$ is said to be a generator if $X.\varphi = Z = Z.\varphi$. We are normally only interested in the minimal ($w.r.t$ inclusion) generators, which we denote by $Z.\gamma$. A closed set Z can have several minimal generators, so we subscript them if necessary, as in $Z.\gamma_1 \ldots Z.\gamma_n$, and let $Z.\Gamma$ denote the set of all minimal generators.

Formal concept analysis yields a lattice of pairs of closed sets, $k : (O_k, A_k)$ where $O_k \subseteq O, A_k \subseteq A$. Every closed set A_k has a set, $A_k.\Gamma = \{A_k.\gamma_1, \ldots, A_k.\gamma_i,$

[1] These rows are often regarded as "transactions" in the data mining literature.

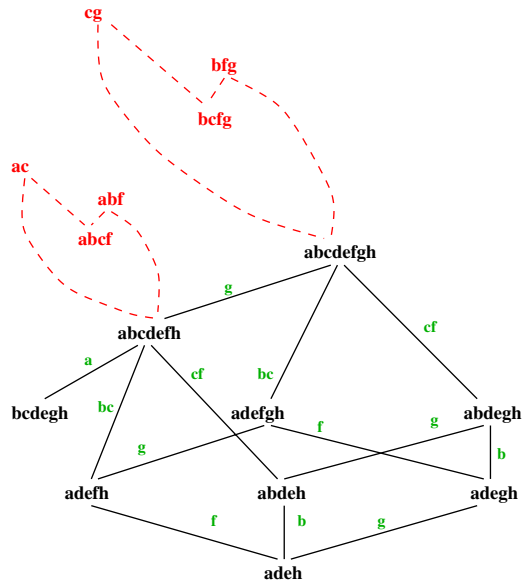

Fig. 1. A small portion of a concept lattice $\mathcal{L}$.

$\ldots, A_k.\gamma_n\}$ of minimal generators. That is, the closure φ of $A_k.\gamma_i$ (the i-th minimal generator of A_k) is $A_k.\gamma_i.\varphi = A_k$ for all i. Figure 1 shows a small portion of a concept lattice. (For this paper, we illustrate only the closed sets of attributes and their generators, but in practice we also retain O_k which constitutes the support of the concept.) The solid lines connect closed concepts. The dashed lines suggest generators. The set cg and bfg are minimal generators of the closed set $abcdefgh$. (Later, using Theorem 1 we will prove that these are its minimal generators. For now, we only observe that $cg.\varphi \subseteq abcdefgh$, but looking at the rest of the concept lattice consisting of closed subsets of $abcdefgh$, we see none containing cg.) All subsets Y such that $cg \subseteq Y \subseteq abcdefgh$, such as $bcfg$ have the same closure; $Y.\varphi = abcdefgh$. The minimal generators, $abcdefh.\Gamma = \{ac, abf\}$. We have illustrated the generating sets for only these two closed sets.

Unlike other research, we keep an explicit list of the minimal generators with each closed concept. Creation of a concept lattice *with generators* is quite a bit slower than, say the algorithm of Godin and Missaoui [8]. But in compensation, one can directly read off all possible implications. For each concept-k, we can assert that $(\forall o \in O)[A_k.\Gamma(o) \rightarrow A_k.\varphi]$. That is, because of the way the concept lattice closure operator is defined, the generators of a closed set logically imply all of the attributes in the closed set[2]. For example, from Figure 1 we immediately see that $(\forall o \in O)[[c(o) \wedge g(o)] \vee [b(o) \wedge f(o) \wedge g(o)] \rightarrow a(o) \wedge b(o) \wedge c(o) \wedge d(o) \wedge$

[2] This φ closure is a Galois closure. See [7] for its precise definition. Also observe that there is a strong similarity to the concept of attribute closure, X^+, with respect to functional dependencies in relational database theory [11].

$e(o) \wedge f(o) \wedge g(o) \wedge h(o)]$. Or more compactly we say, $(cg \vee bfg) \rightarrow abcdefgh$ and $(ac \vee abf) \rightarrow abcdefh$. And, as Zaki has observed [24], because the left side of these implications is minimal while the right side is maximal, we can expect several orders of magnitude fewer implications than obtained with frequent set data mining. In Section 4, we describe in some detail the results of applying our method to a $8,124 \times 54$ relational database describing mushroom characteristics. The closed set DDDM method yields 2,640 concepts (or implications) versus 15,552,210 associations derived by frequent set mining[3]. In a smaller 76×141 relational database listing the properties of marine sponges, frequent set mining yields 22,029,172 associations to 77,762 concepts with DDDM.

3 Incremental Growth

Since a concept lattice $\mathcal{L}$ captures all the logical attribute implications one can make about a collection of objects; it is natural to ask "suppose we observe one more object and its attributes. The set of attributes characterizing some new observation o' must either (a) be identical to an existing concept, in which case only the support of that concept need be incremented, or (b) the attribute set constitutes a new closed concept whose only support is itself. How will this latter case transform the lattice $\mathcal{L}$?" This is the essence of discrete, empirical induction. Our understanding of world phenomena normally occurs incrementally. We assimilate new observations, or rows of R if you choose, into an existing concept structure. Some new observations reinforce existing understanding; others lead to new understanding.

Godin and Missaoui [8] have shown one can easily insert a new concept into $\mathcal{L}$ using the intersection property of closed sets. Since the intersection of closed sets must be closed, if a new closed set, call it Y is introduced into the lattice so that it is covered by an existing set Z. Then at most we need only concern ourselves with the substructure below Z. This nature of closure systems ensures that the resulting changes will be, in some sense, "local". Figure 2, in which the closed set $Y = acdegh$ has been inserted, illustrates this principle. In addition to $acdegh$, a second closed set $acdeh$ also had to be added because the intersection of closed sets $abcdefh \cap acdegh$ must be closed. See the dot-dashed lines in Figure 2.

But, this ignores the real problem. How does such an addition change the generators of existing closed sets? Or equivalently, given a system of universally quantified implications, $(\forall o \in O)[P(o) \rightarrow Q(o)]$, how does a single existential assertion of the form $(\exists o' \in O)[P'(o')]$ necessarily change this logical system?

In Figure 1 (and Figure 2) we have labeled the edges with those elements which represent the difference between the closed sets. Each difference we call a face F of the larger closed set. For example, the faces of $abcdefgh$ are $\{g, bc, cf\}$. A close relationship between the faces and the generators of a closed set was proven, in [18], that is:

[3] In all comparisons, we set $\sigma = 3\%$ and $\gamma = 95\%$ in the *apriori* procedure.

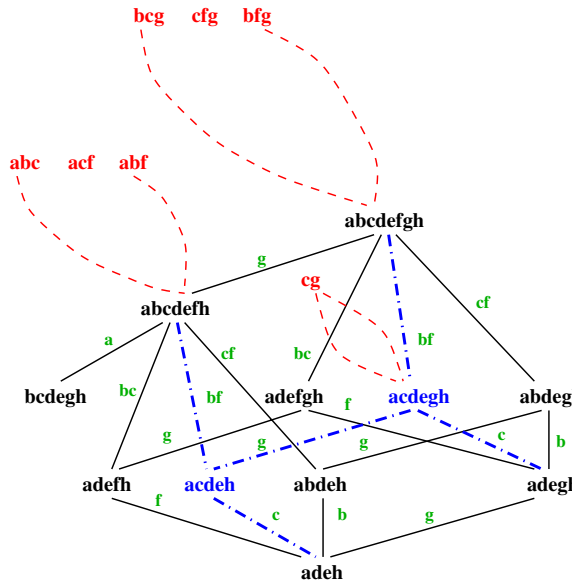

Fig. 2. The concept lattice of Figure 1 after *acdegh* has been incrementally added.

Theorem 1. *If Z is closed and $Z.\Gamma = \{Z.\gamma_i\}$ is its family of minimal generators then Z covers X in $\mathcal{L}$ iff $Z - X$ is a minimal blocker of $Z.\Gamma$.*

A set $F = Z - X$ blocks the family of generators $Z.\Gamma$ if for each $Z.\gamma_i \in Z.\Gamma$, $F \cap Z.\gamma_i \neq \emptyset$. Similarly each generator $Z.\gamma_i$ is a minimal blocker of the set of faces.

A new closed concept Y (*e.g. acdegh*) will be covered by a single closed concept Z [4]. Consequently, Y creates a new face F (*e.g.* bf) of Z. If an existing generator $Z.\gamma_i \subseteq Y$, then it can no longer be a generator of Z. If $Z.\gamma_i \cap Y = \emptyset$, it must be augmented by elements of F so that each becomes a blocker of F. In this case, $bfg \cap bf \neq \emptyset$, so bfg remains a generator with no changes. However, $cg \cap bf = \emptyset$, so $cg \cup b$ becomes a new generator of Z, as does $cg \cup f$. In the following pseudocode, a COLLECTION denotes a set of SETS.

```
void update_generators( CONCEPT cov_c, CONCEPT new_c )
     // "cov_c" will cover the new concept "new_c". update cov_c.gens to
     // reflect the face cov_c.atts - new_c.atts
     {
     ELEMENT     elem;
     SET         face, gsup;
     COLLECTION  new_GENS, keep_GENS, diff_GENS;
     BOOLEAN     minimal;
```

[4] Z covers Y if $Y \subset Z$, but there is no closed set Y' such that $Y \subset Y' \subset Z$. That there can be only one such Z is easily shown using the intersection property of closed sets.

```
face = cov_c.atts - new_c.atts;
keep_GENS = EMPTY_COL;
// these generators will be unchanged by the insertion
for each gen_set in cov_c.GAMMA
     if( not_EMPTY (gen_set INTERSECT face) )
          add_to_collection (gen_set, keep_GENS);
new_GENS = keep_GENS;
diff_GENS = cov_c.GAMMA - keep_GENS;
for each gen_set in diff_GENS
     {                    // gen_set does not intersect the new face
     for each elem in face
          {
          minimal = true;
          gsup = gen_set UNION {elem};
          for each keep_set in keep_GENS
               if( keep_set IS_SUBSET_OF gsup )
                    {             // gsup is not a minimal generator
                    minimal = false; break;
                    break;
                    }
          if( minimal )
               add_to_collection (gsup, new_GENS);
          }
     }          // replace cov_c.GAMMA with the new generating sets
cov_c.GAMMA = new_GENS;
}
```

Next the intersections of Y with each of its "siblings" (other concepts covered by Z) must be calculated to determine the faces, and generators, of Y itself. If one of these intersections, such as *acdeh*, yields a new closed concept not already in $\mathcal{L}$, it is recursively added in the same manner as Y.

```
insert_closed_set (SET Z, SET Y, LATTICE L)
       // Insert the closed set Y into L so that it is covered
       // by Z
       {
       SET    Y[];

       update_generators (Z, Y);
       for_each Y[i] covered by Z do
              {             // Y[i] will be a sibling of Y
              if ( not_EMPTY (Y INTERSECT Y[i]) )
                     {
                     update_generators (Y, Y INTERSECT Y[i]);
                     if (Y INTERSECT Y[i] in L)
                            continue;
```

```
                    X = new SET (Y INTERSECT Y[i]);
                    insert_closed_set (Y[i], X, L);
                    }
                }
            }
```

Entry of a new closed set Y into an existing concept lattice $\mathcal{L}$ can require a variable number of additional forced insertions, through recursive calls to INSERT_CLOSED_SET in the code above. As expected, the local nature of each insertion limits the extent of this recursion[5]. In Figure 3, we plot the *average* relative increase per insertion (dashed line) represented by a sequence of up-

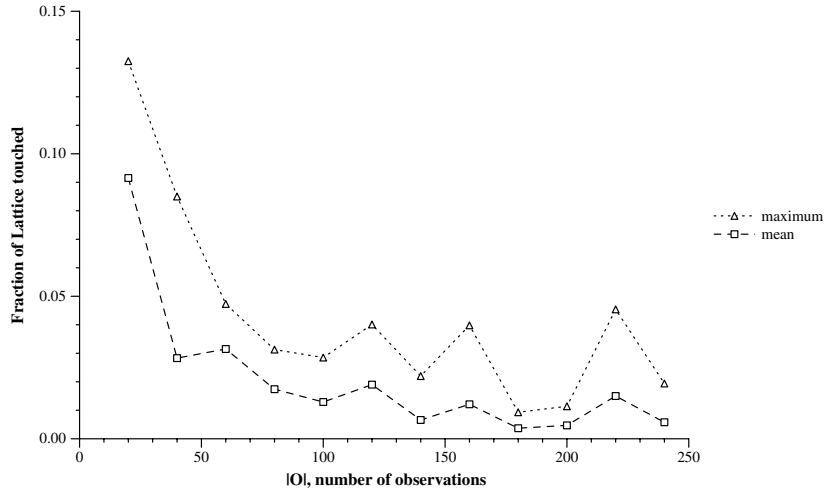

Fig. 3. Relative increase in $\mathcal{L}$ per insertion.

dates to a random 20 attribute relation together with the *maximum* relative increase (dotted line). This graph, although generated synthetically, appears to be quite representative. Initial entries into the lattice affect nearly the entire lattice, $\mathcal{L}$. But gradually the structure stabilizes, and successive entries tend to be much more local. A process which normally affects approximately 2% of the total lattice structure can reasonably be called "local".

We have discovered that, compared to a batch process which analyzes the entire relation R, as required by *a priori* [1], incremental update decreases processing time upto 3 orders of magnitude. Incremental lattice transformation makes concept lattices with generators a practical knowledge discovery method.

[5] It was this locality that Godin and Missaoui [8] exploited. But, although their paper gives performance results in terms of times, it does not detail an actual counts of their recursive step.

4 Representative Output

Interesting data sets are large. One cannot visualize them; we can only select representative concepts from the lattice and display them[6]. To get a feeling for DDDM we consider the well-known MUSHROOM data set, which we obtained from the UCI Machine Learning Repository at

`http://www1.ics.uci.edu/ mlearn/MLRepository.html`.

Many data mining experiments, using this data set, have been reported in the literature. Most have been concerned with the edibility of various mushrooms. The data set R consists of 8124 observations of 22 nominal attributes. Attribute-0 has values "edible" and "poisonous", denoted `e0` and `p0` respectively. For illustrative brevity, we ran the process using just the first 9 attributes listed below:

Attr-0 **edibility**: e=edible, p=poisonous;
attr-1 **cap shape**: b=bell, c=conical, f=flat, k=knobbed, s=sunken, x=convex;
attr-2 **cap surface**: f=fibrous, g=grooved, s=smooth, y=scaly;
attr-3 **cap color**: b=buff, c=cinnamon, e=red, g=gray, n=brown, p=pink, r=green, u=purple, w=white, y=yellow;
attr-4 **bruises?**: t=bruises, f=doesn't bruise;
attr-5 **odor**: a=almond, c=creosote, f=foul, l=anise, m=musty, n=none, p=pungent, s=spicy, y=fishy;
attr-6 **gill attachment**: a=attached, d=descending, f=free, n=notched;
attr-7 **gill spacing**: c=close, d=distant, w=crowded;
attr-8 **gill size**: b=broad, n=narrow.

Because of multiple attribute values, these correspond to a binary array of 54 boolean attributes. The concept lattice generated by this $8,124 \times 54$ binary relation, R, consists of 2,640 concepts.

Implications with a single antecedent are often the most important. They are the easiest to apply in practice. Scanning the concept lattice for singleton generators yields the 22 implications of Figure 4. We see that 12 of the implications have to do with edibility, `e0` or `p0`.

We list the support for each rule to the right. The reader can decide whether the σ-frequency they use would have discovered the implication. For example, to discover that mushrooms with "sunken" caps are edible (#313) would require $\sigma < 0.004$. Such a low σ-value would suggest that the number of frequent sets would approach $2^{54-\epsilon}$, or possibly as many as $2^{50} = 1.125 \times 10^{15}$, a number that can exhaust main memory.

Virtually any data mining process would discover that "odor" is a crucial determinant. In particular, a "creosote"(#668), "foul"(#924), "musty"(#2022), "spicy"(#1597) or "fishy"(#1687) odor betokens "poisonous". Since "almond" (#117) and "anise"(#144) indicate "edible", only "no odor" is ambiguous. Such a mushroom can be "edible"(#313, #1081, #1553) or "poisonous"(#1401,

[6] It is interesting to "surf" the lattice following links to higher and lower conclepts; but demonstration of this process in a paper is impractical.

CONCEPT	IMPLICATION			SUPPORT
60	w3	->	f6	1040
105	t4	->	f6	3376
109	n8	->	f6	2512
117	a5	->	e0, t4, f6	400
144	l5	->	e0, t4, f6	400
313	s1	->	e0, f2, f4, n5, f6, c7, n8	32
556	f2	->	f6	2320
604	w7	->	f6	1312
668	c5	->	p0, x1, f4, f6, n8	192
720	y3	->	f6	1072
898	b3	->	t4, f6, c7, b8	168
924	f5	->	p0, f6, c7	2160
1007	p3	->	f6	144
1081	u3	->	e0, y2, f4, n5, f6, c7, n8	16
1401	g2	->	p0, w3, t4, n5, f6, w7, n8	4
1553	r3	->	e0, y2, f4, n5, f6, c7, n8	16
1597	s5	->	p0, f4, f6, c7, n8	576
1687	y5	->	p0, f4, f6, c7, n8	576
2019	a6	->	f4, c7, b8	210
2022	m5	->	p0, y2, f4, c7, b8	36
2162	e3	->	c7	1500
2562	c1	->	p0, n5, f6, w7, n8	4

Fig. 4. Implications in the MUSHROOM data set that have singleton generators (or antecedents).

#2562). There are only 4 conically capped instances and only 4 with grooved cap surfaces; but, although not frequent, eating any might be unpleasant.

Because "poisonous" is thought to be an important characteristic of mushrooms, we scanned the concept lattice for those concepts which (a) had `p0` in the closed (consequent) set, and (b) had a two element generator *not* containing `p0`. There are 64 such implications. We then passed this list through a filter, eliminating those whose generators included a poisonous odor, *viz.* `c5, f5, m5, s5` or `y5`. The resulting 15 implications are shown in Figure 5.

We observe that 7 of these instances could also be determined by odor, either `c5` or `m5`. But, 7 have "no odor" (`n5`) and would thereby be ambiguous in any case. In none of these extractions has the support played a role. DDDM implications are found independently of their frequency — which might be desirable if one is considering tasting one of the 876 instances of "red" mushrooms that "don't bruise" easily. Figure 6 illustrates the same kinds of logical criteria for edibility. Figures 5 and 6 both illustrate implications used to "classify" data, into either "edible" or "poisonous". It is worth comparing the DDDM approach with the nice approach found in [13].

Because DDDM yields implications that are universally quantified over the data set, one can perform logical transformations. One can reason. But, we must consider to data errors. Since it is not statistical, DDDM is not forgiving of erroneous input. If a new observation o' would change the generators of a concept above a specified threshold, our system can flag the observation and defer the insertion. The observation is then carefully examined for validity, and either discarded or reentered.

This approach to error control has an unexpected benefit. Suppose for example in a study of animal species that $a_1 \equiv$ "nurses its young" and $a_2 \equiv$ "gives live

CONCEPT	IMPLICATION			SUPPORT
666	p3, w7	->	p0, x1, f4, c5, f6	32
667	p3, f4	->	p0, x1, c5, f6, n8	64
667	p3, n8	->	p0, x1, f4, c5, f6	64
696	f2, p3	->	p0, x1, f4, c5, f6, n8	32
1184	b1, n8	->	p0, n5, f6	12
1495	b1, b3	->	p0, t4, n5, f6, c7, b8	12
1567	b1, p3	->	p0, t4, n5, f6, c7, b8	12
2081	y3, n5	->	p0, f4, f6, n8	24
2177	e3, f4	->	p0, c7	876
2181	y2, a6	->	p0, f4, m5, c7, b8	18
2372	c3, a6	->	p0, y2, f4, m5, c7, b8	6
2470	e3, a6	->	p0, y2, f4, m5, c7, b8	6
2561	c1, y3	->	p0, y2, f4, n5, f6, w7, n8	2
2561	c1, f4	->	p0, y2, y3, n5, f6, w7, n8	2
2563	c1, y2	->	p0, n5, f6, w7, n8	3

Fig. 5. Implications of "poisonous" in the MUSHROOM data set with two generating attributes (or antecedents).

CONCEPT	IMPLICATION			SUPPORT
61	w7, b8	->	e0, f4, n5, f6	1056
119	y3, t4	->	e0, f6	400
131	s2, y3	->	e0, t4, f6	152
452	f2, t4	->	e0, f6	912
586	f2, e3	->	e0, t4, n5, f6, c7	288
1015	x1, r3	->	e0, y2, f4, n5, f6, c7, n8	8
1261	k1, b3	->	e0, t4, n5, f6, c7, b8	16
1347	f2, c3	->	e0, f4, n5, f6, w7, n8	12
1944	b1, g3	->	e0, f4, n5, f6, w7, b8	48
1966	k1, g3	->	e0, f4, m5, f6, w7, b8	48
2342	s2, c3	->	e0, t4, m5, f6, c7, b8	4
2343	c3, t4	->	e0, n5, f6, c7, b8	8

Fig. 6. Implications of "edible" in the MUSHROOM data set with two generating attributes (or antecedents).

birth". The implication $a_1 \rightarrow a_2$ is supported by thousands of observations — until we encounter a "duck-billed platypus". If convinced of its validity, we still flag the occurance as being "unusual", and hence of possible importance. Surely the duck-billed platypus is a relative unimportant mammal, *except* for this deviant behavior. Because DDDM works with deterministic, logical assertions, we can uncover this kind of "outlying" occurance.

A great deal of the phenomena around us is non-deterministic and statistical in nature. As a tool, DDDM is inappropriate to analyze this kind of data. But, when we are looking for deterministic, causal relationships, incremental DDDM becomes an effective knowledge discovery process. Moreover, it mimics to a large extent the procedures that classical science has always used to understand its world.

5 Conclusions

In the past graphs have been primarily used as a static display of the relationships between objects. Only fairly recently has the transformation of graphs really been regarded as an area of interest. There have been many formal studies of relatively abstract graph transformation, *e.g.* [3,9,14]. The transformation of program structure was an early target of the field [20,6] which is naturally extended to type structures [5]. Having working systems such as [4,21] has been a boon. Since UML diagrams are already in graphical form, their transformation becomes interesting [23].

Data mining has not been a traditional target of graph transformation (in this case lattice transformation); but it ought to be. This paper and [19] have demonstrated that an iterative, transformational approach to data mining can be far more effective that frequent set *apriori* mining given the right circumstances. One can alternatively regard this as empirical learning where a knowledge structure is transformed by new observations [17]. The empirical accquisition of knowledge [12] and concept formation are ripe for further exploration.

References

1. Jean-Mark Adamo. *Data Mining for Association Rules and Sequential Patterns.* Springer Verlag, New York, 2000.
2. Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining Association Rules between Sets of Items in Large Databases . In *Proc. 1993 ACM SIGMOD Conf.*, pages 207–216, Washington, DC, May 1993.
3. Stefan Arnborg, Bruno Courcelle, Andrzej Proskurowski, and Detlef Seese. An Algebraic Theory of Graph Reduction. In Hartmut Ehrig, Hans-Jorg Kreowsik, and Gregor Rozenberg, editors, *Graph Grammars and their Application to Computer Science*, Lecture Notes in Computer Science, #532, pages 70–83, Bremen, Germany, 1990. Springer-Verlag.
4. R. Bardohl, T. Schultzke, and G. Taentzer. Visual Language Parsing in GenGEd. In *2nd International Workshop on Graph Transformation and Visual Modeling Techniques*, pages 291–296, Crete, Greece, July 2001. Satellite Workshop of ICALP 2001.
5. H. Ehrig and F. Orejas. Integration Paradigm for Data Type and Process Specification Techniques. In G. Paun, G. Rozengerg, and A. Salomaa, editors, *Current Trends in Theoretical Computer Science: Entering the 21st Century*, pages 192–201. World Scientific, Singapore, 2001.
6. Gregor Engels, Claus Lewerentz, and Wilhelm Schafer. Graph Grammar Engineering: A Software Specification Method. In Hartmut Ehrig, Manfred Nagl, Gregor Rozenberg, and Azriel Rosenfeld, editors, *Graph Grammars and their Application to Computer Science*, Lecture Notes in Computer Science, #291, pages 186–201, Warrenton, VA, 1986. Springer-Verlag.
7. Bernard Ganter and Rudolf Wille. *Formal Concept Analysis - Mathematical Foundations.* Springer Verlag, Heidelberg, 1999.
8. Robert Godin and Rokia Missaoui. An incremental concept formation approach for learning from databases. In *Theoretical Comp. Sci.*, volume 133, pages 387–419, 1994.

9. Reiko Heckel, Hartmut Ehrig, Gregor Engels, and Gabriele Taentzer. Classification and Comparison of Modularity Concepts for Graph Transformation Systems, 1999.
10. 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.
11. David Maier. *The Theory of Relational Databases*. Computer Science Press, Rockville, MD, 1983.
12. Brian Mayoh. Graph Grammars for Knowledge Representation. In Hartmut Ehrig, Hans-Jorg Kreowsik, and Gregor Rozenberg, editors, *Graph Grammars and their Application to Computer Science*, Lecture Notes in Computer Science, #532, page 53, Bremen, Germany, 1990. Springer-Verlag.
13. Carlos Ordonez, Edward Omiecinski, Levien de Braal, Cesar A. Santana, Norberto Ezquerra, Jose A. Taboada, David Cooke, Elizabeth Krawczynska, and Ernst V. Garcia. Mining Constrained Association Rules to Predict Heart Disease. In *IEEE International Conf. on Data Mining, ICDM*, pages 433–440, 2001.
14. J. Padberg, H. Ehrig, and L. Riberio. Algebraic High-Level Net Transformation Systems. *Math. Structures in Computer Science*, 5:217–256, 1995.
15. Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lofti Lakhal. Discovering Frequent Closed Itemsets for Association Rules. In *Proc. 7th International Conf. on Database Theory (ICDT)*, pages 398–416, Jan. 1999.
16. John L. Pfaltz. Closure Lattices. *Discrete Mathematics*, 154:217–236, 1996.
17. John L. Pfaltz. Transformations of Concept Graphs: An Approach to Empirical Induction . In *2nd International Workshop on Graph Transformation and Visual Modeling Techniques*, pages 320–326, Crete, Greece, July 2001. Satellite Workshop of ICALP 2001.
18. John L. Pfaltz and Robert E. Jamison. Closure Systems and their Structure. *Information Sciences*, 139:275–286, 2001.
19. John L. Pfaltz and Christopher M. Taylor. Concept Lattices as a Scientific Knowledge Discovery Technique. In *2nd SIAM International Conference on Data Mining*, pages 65–74, Arlington, VA, Apr. 2002.
20. Terrence W. Pratt. Pair grammars, graph languages, string to graph translations. *JCSS*, 5(6):760–795, Dec. 1971.
21. Andy Schurr. PROGRESS, A VHL-Language Based on Graph Grammars. In Hartmut Ehrig, Hans-Jorg Kreowsik, and Gregor Rozenberg, editors, *Graph Grammars and their Application to Computer Science*, Lecture Notes in Computer Science, #532, pages 641–659, Bremen, Germany, 1990. Springer-Verlag.
22. Ramakrishnan Srikant and Rakesh Agrawal. Mining Quantitative Association Rules in Large Relational Tables . In *Proc. of 1996 SIGMOD Conference on Management of Data*, pages 1–12, Montreal, Quebec, June 1996.
23. Aliki Tsiolakis. Integrating Model Information in UML Sequence Diagrams. In *2nd International Workshop on Graph Transformation and Visual Modeling Techniques*, pages 268–276, Crete, Greece, July 2001. Satellite Workshop of ICALP 2001.
24. Mohammed J. Zaki. Generating Non-Redundant Association Rules. In *6th ACM SIGKDD Intern'l Conf. on Knowledge Discovery and Data Mining*, pages 34–43, Boston, MA, Aug. 2000.