# Inferring dependencies from relations: a conceptual clustering approach

CLAUDIO CARPINETO, GIOVANNI ROMANO AND PAOLO D'ADAMO

Fondazione Ugo Bordoni,

Via B. Castiglione 59, 00142 Rome, Italy

Tel: +39-6-54803426

Fax: +39-6-54804405

E-mail: carpinet@fub.it

## Abstract

In this paper we consider two related types of data dependencies that can hold in a relation: conjunctive implication rules between attribute-value pairs, and functional dependencies. We present a conceptual clustering approach that can be used, with some small modifications, for inferring a cover for both types of dependencies. The approach consists of two steps. First, a particular clustered representation of the relation, called concept (or Galois) lattice is built; then, a cover is extracted from the lattice built in the earlier step. The main emphasis of this paper is on the second step. We study the computational complexity of the proposed approach and present an experimental comparison with other methods that confirms its validity. The results of the experiments show that our algorithm for extracting implication rules from concept lattices clearly outperforms an earlier algorithm, and suggest that the overall lattice-based approach to inferring functional dependencies from relations can be seen as an alternative to traditional methods.

**Keywords:** knowledge discovery, implication rules, functional dependencies, conceptual clustering, Galois lattices, computational complexity, empirical evaluation.

# 1. INTRODUCTION

Recent research across a number of fields including databases, machine learning, and knowledge discovery has addressed the problem of automatic extraction of data dependencies from relational databases. A large variety of data dependencies have been investigated, ranging from classical functional dependencies (Mannila and Räihä 1986, 1994; Schlimmer 1993; Savnik and Flach 1993) to several types of attribute-oriented rules (Piatetsky-Shapiro 1991; Agrawal and Srikant 1994; Godin and Missaoui 1994a, Ziarko and Shan 1996), and a number of systems for their extraction have been presented. While differing in many respects, most of this work has two main common features.

One major characteristic of rule discovery systems is the goal of finding *all* possible rules of a given type that can be extracted from data, as opposed to other methods for inducing rules from data, such as those developed in machine learning, which are primarily biased towards producing minimum subsets of classification rules (Michalski 1983, Quinlan 1986, Cai *et al.* 1991). As a general consequence, specific methods must be devised for exhaustively searching a large hypothesis space and producing a virtually unbiased set of rules; by contrast, the use of classical machine learning algorithms, such as decision trees, would typically result in the omission of many possible, and equally plausible, rules (Carpineto and Romano 1993), while a straightforward adaptation of such methods, although possible, would suffer from serious inefficiency as well as redundancy problems (Schlimmer 1993; Oosthuizen 1994).

Another common feature of rule discovery systems is the ability to deal with compact representations of the set of rules generated. As such sets may grow very large, thus making the presentation, comprehension and utilization of the rules themselves very difficult, it is necessary to focus only on the interesting rules, discarding the irrelevant ones. Of the many optimality and pruning criteria that have been presented (Piatetsky-Shapiro 1991; Shan *et al.* 1995, Toivonen *et al.* 1995), the notion of *cover*, as developed in the database theory (Maier 1983; Ullman 1988), is probably one of the best well-founded and understood, for which there exists a large body of concepts and methods. While the use of this notion has been traditionally confined to functional dependencies, it has been recently applied also to attribute-oriented dependencies (Godin and Missaoui 1994a).

In this paper we deal with the problem of finding a reduced cover for two distinct, yet related, classes of data dependencies that can be extracted from a relation: conjunctive implication rules between the attribute-value pairs describing the objects in the relation, useful for discovering hidden patterns in data, and functional dependencies, useful in database design. Our approach is based on a particular clustering structure built from the relation, called concept (or Galois) lattice (Wille 1984; Davey and Priestley 1990), that contains all complete classes of objects having a conjunctive description, ordered by generality/specificity. Godin and Missaoui (1994a) have recently shown that concept lattices can be used as support structures for extracting covers of the set of (conjunctive) implication rules that hold in a relation. We use Godin and Missaoui‘s intuitions and findings as a starting point for this research. First of all, we introduce a slightly generalised definition of implication rules that allow for statistical significance of the rules generated. Then we provide an algorithm for extracting a cover for such rules from the lattice that turned out to be much more efficient than Godin and Missaoui’s earlier algorithm. The next step is to extend the lattice-mediated approach to other inference tasks. We advocate that the same basic methodology used for inferring implication rules can be adapted to solve the problem of inferring functional dependencies. A major part of this paper is a study of the complexity of the latter approach along with an experimental comparison with Mannila and Räihä’s well-known algorithm for finding functional dependencies (Mannila and Räihä 1986, 1994). The results of the experiment showed that the the lattice-based approach performed favourably.

The rest of the paper has the following structure. In the next section, we introduce implication rules and generalised implication rules, characterizing their relationships with association rules and with functional dependencies. In section 3, we provide the basics of the theory of concept lattices and justify their utilization for supporting dependency inference. Section 4 is devoted to the inference of rules; we describe an algorithm for rule extraction based on concept lattices, discuss the complexity of the rule inference problem and that of the algorithm, and present an experimental comparison between the performance of our algorithm and that of Godin and Missaoui's one. Section 5 describes the extended method for inferring functional dependencies, studies its complexity, and provides a comparison with Mannila and Räihä's algorithm. Section 6 concludes the paper with a summary and some directions for future work.

## 2. IMPLICATION RULES AND FUNCTIONAL DEPENDENCIES

Given a set of objects (O), a set of attributes (A), and a set of attribute values (V), a relation (or *context*, as it is usually referred to in the concept lattice theory literature) is a quadruple (O, A, V, I) where I is a ternary relation between O, A, and V (i.e., $I \subseteq O \times A \times V$) such that $(o,a,v_1) \in I$ and $(o,a,v_2) \in I$ imply $v_1 = v_2$. Note that $(o,a,v) \in I$ reads: the object *o* has the value *v* for the attribute *a*; instead of writing $(o,a,v) \in I$ we can write $a(o) = v$. This is the usual relation employed in relational databases; our notation, borrowed from the theory of concept lattices, has the advantage of expliciting taking into account both the presence of attributes and attributes values, therefore helping clarify the relationships between the two types of dependencies that we consider in this paper.

*Definition 1.* An implication rule (IR) between two sets of attribute-value pairs is an expression $[(r_1, s_1), (r_2, s_2),...(r_h, s_h)] \rightarrow [(t_1, u_1), (t_2, u_2),... (t_k, u_k)]$ where $(r_x, s_x), (t_x, u_x) \subseteq (A \times V)$.

A relation (O, A, V, I) satisfies the IR $[(r_1, s_1), (r_2, s_2),...(r_h, s_h)] \rightarrow [(t_1, u_1), (t_2, u_2),... (t_k, u_k)]$ if $\forall\ o \in O$, $[r_1(o)=s_1 \wedge r_2(o)=s_2 \wedge ...\ r_h(o)=s_h] \Rightarrow [t_1(o) = u_1 \wedge t_2(o)=u_2 \wedge ...\ t_k(o)=u_k]$.

In other terms, an implication rule between two subsets of attribute-value pairs Q and R means that if a set of objects satisfies the attribute-value pairs contained in Q then it necessarily satisfies the attribute-value pairs contained in R. For the sake of conciseness, in the following we often refer to an IR between two subsets of attribute-value pairs as Q$\rightarrow$R.

Although it is always theoretically possible to find all the IRs that a context satisfies, such an approach would be impractical. Fortunately, knowing some members of a set of IRs $\Sigma$, it is often possible to infer other members of $\Sigma$. In fact, owing to the nature of IRs, the inference system developed in database theory for functional dependencies (Maier, 1983) holds also for IRs. The only caution is that it seems useful to restrict the axiom "augmentation" (Q$\rightarrow$R implies QZ$\rightarrow$R) by requiring that the new left side (QZ) should be contained in at least one object of the context. The following definitions are useful to find succint representations of the complete set of IRs that hold in a context.

Given a set $\Sigma$ of IRs, the closure $\Sigma^+$ is the set of rules implied by $\Sigma$ by application of Armstrong's inference axioms. Two sets $\Sigma$ and $\Sigma$' are equivalent if they have the same closure. If $\Sigma$ and $\Sigma$' are equivalent, then $\Sigma$' is a *cover* for $\Sigma$. A cover $\Sigma$' for $\Sigma$ is nonredundant if removing any IR from $\Sigma$' would give a set of IRs that was not equivalent to $\Sigma$.

IRs, sometimes called implications (Guiges and Duquenne 1986; Wille 1992) or simply rules (Ziarko and Shan 1996), are a simple form of regularities and have the advantage of supporting reasoning. However, they suffer from one main practical limitation. Following Definition 1, an IR holds even if there is only one object that satisfies the rule; consequently, there may be many spurious and irrelevant rules in the set of IRs that hold in a context. To alleviate this problem, it is convenient to generalise the above definition to allow for statistical significance of the holding rules.

*Definition 2.* Given a real number $\alpha$ (support threshold), $0 \leq \alpha < 1$, a generalised IR is an IR that is satisfied by more than $n\alpha$ objects, where $n$ is the number of objects in the context.

The introduction of the support threshold $\alpha$ (note that for $\alpha < 1/n$ we get just the definition of ungeneralised IRs) does not adversely affect reasoning in that inference axioms hold also for generalised IRs. The support threshold $\alpha$ is reminiscent of an analogous threshold used to define association rules, a related type of data regularity that has attracted some attention recently (Agrawal and Srikant 1994; Toivonen *et al.* 1995). Before we proceed, it is useful to clarify that IRs and association rules should not be confused at all. Roughly, association rules are statements of the form "for a certain percentage of the objects, if an object has Q then it has also R". Thus, there may well be an association rule $Q \rightarrow R$ with some fraction of the objects having Q and not having R, while for a generalised IR to hold, it is necessary that whenever an object has Q it has also R. Association rules are more flexible, but they do not support inference axioms, unless we make particular assumptions about the description of the data (Toivonen et al 1995). In a sense, generalised IRs seem to retain most advantages of IRs and association rules while avoiding their main disadvantages.

The other kind of data dependency that we consider in this paper is functional dependency, which has been deeply investigated in database theory (Maier, 1983; Ullman 1988) . Recall that for $H, K \subseteq A$, K is called functionally dependent on H if for all $o_1, o_2 \in O$, $h(o_1) = h(o_2)$ for all $h \in H$ implies $k(o_1) = k(o_2)$ for all $k \in K$.

Implication rules ($\alpha < 1/n$) and functional dependencies have strong relationships. In a sense, FDs can be seen as an abstraction of IRs. Given the set of IRs that hold for a context, in order to see whether the FD $H \rightarrow K$ holds, it is sufficient to check that there is an implication rule between each combination of the values taken on by the attributes in H and some combination of the values of the attributes in K. Deriving the set of IRs from the set of FDs is more difficult, because there may be IRs that hold despite the fact the corresponding attributes are not functionally dependent. As an illustration, we refer (see Table 1) to a simple relation consisting of nine objects (the planets of our solar system) described by four many-valued attributes; i.e., size (small/medium/large), distance from sun (near/far), moon (yes/no), and period (short/long). Many IRs hold in this context that do not have a functional counterpart. For instance, while the attribute *distance from sun* does not functionally determines the attribute *planet size,* the attribute-value pair *distance-near* implies the attribute-value pair *size-small.*

Thus, implication rules allow discovery of data dependencies which would be not disclosed by functional dependencies. It is also worth noting that IRs, unlike FDs, cannot hold vacuosly; therefore, given a FD between two sets of attributes, it is not possible to automatically derive the corresponding set of IRs without checking which values are taken on by those attributes. In the planet context, for example, the FD *(size distance)* $\rightarrow$ *(period)* is valid, but there is no IR whose left-hand side is (*size-medium distance-near*), because this combination of attribute-value pairs never appears in the context.

Covers of the set of IRs that hold in the planet context, as well as of the set of FDs, are shown in section 4.1 and 5.1, respectively.

## 3. USING CONCEPT LATTICES TO SUPPORT DEPENDENCY INFERENCE

In this section we first give a brief characterization of concept lattices, and then we introduce some basic notions and principles concerning the utilization of concept lattices as support structures for dependency inference.

### 3.1. Concept lattices: basic notions

For $X \subseteq O$ and $Y \subseteq AxV$, define

X' = {(a∈ A, v∈ V) | ∀ o∈ X, a(o) = v}, Y' = {o∈ O | ∀ (a,v)∈ Y, a(o) = v};

so X' is the set of attribute-value pairs common to all objects in X and Y' is the set of objects possessing all the attribute-value pairs in Y. Then a concept of the context (O,A[illegible]efined to be a couple (X, Y) where $X \subseteq O$, $Y \subseteq AxV$, and

X' = Y, Y' = X;

X and Y are called the *extent* and the *intent* of the concept, respectively. Ther[illegible] only some couples (X,Y) - i.e., the couples that are complete with respect to I accordi[illegible] the given definition - represent admissible concepts. Note that a subset Y of AxV is the[illegible]nt of some concept if [illegible] the unique concept of which Y is an [illegible]s (Y',Y); a dual st[illegible]ent hold[illegible] of a [illegible].

The set of al[illegible]pts of the [illegible]O,A,V,I) is denoted by (O,A,V,I). An or[illegible] relation (≤) is[illegible]ly defined on this set of concepts by

$(X_1,$ [illegible] $(X_2, Y_2) \leftrightarrow X_1 \subseteq$ [illegible]

or, equivalen[illegible]y

$(X_1, Y_1)$ [illegible] $Y_2) \leftrightarrow Y_1 \supseteq Y_2$[1]

(O,A,V,I) along w[illegible]red set; it turns out that this ordered set is a complete lattice (see Wille (1984) and Davey and Priestley (1990) for the theorem and further mathematical characterizations).

As an illustration, in Figure 1 we show the concept lattice associated with the context introduced in Table 1. The lattice is represented by a Hasse diagram, in which there is an edge between two nodes if and only if they are comparable and there is no other intermediate concept in the lattice. The ascending paths represent the subconcept/superconcept relation; the bottom concept is defined by the set of all attribute-value pairs and contains no objects, the top concept contains all objects and is defined by their common attribute-value pairs (possibly none, as in

[1] $Y_1$ and $Y_2$ are sets of attribute-value pairs. Two attribute-value pairs are equal if they have the same attribute and the same attribute value.

this case). The labels in Figure 1 show the intent of each node, as well as the extent of those nodes whose intent coincides with object descriptions. It should be noted that, owing to the completeness property, the lattice usually contains a small subset of the couples that can be theoretically generated; in this case, for instance, the intent *dn* does not belong to any concept in the lattice in that all objects having *dn* have also *ss* and *ps*.

This approach to conceptual clustering has gained some attention recently. Several algorithms for computing the lattice have been devised, both nonincremental (Bordat 1986) and incremental (Godin *et al.* 1991, 1995; Carpineto and Romano 1993, 1996), and many applications have been proposed, including data mining (Godin *et al.*, 1991; Godin and Missaoui, 1994a), predictive tasks (Carpineto and Romano 1993), and browsing retrieval (Carpineto and Romano 1996). Unlike most conceptual clustering structures, concept lattice possesses a clear semantics by which to characterize the whole hierarchy in terms of the object description. Its most distinguishing features are that the cluster structure contains *all* complete classes of objects having a conjunctive description, and that the Hasse diagram shows for each concept its minimal conjunctive enlargements and refinements, with respect to that context. In the next section we will see how, based on these properties, it is possible to use a concept lattice for supporting IR inference from relations.

## 3.2. Lattice-based inference of dependencies: preliminaries

One basic property of concept lattices is expressed by the following proposition (Godin and Missaoui 1994a).

*Proposition 1*. The IR $Q \rightarrow R$ holds if and only if the smallest concept (w.r.t. the intent) containing Q as a part of its intent is also described by R.

For example, referring to the context shown in Table 1, the rule *(dn)* $\rightarrow$ *(ps ss)* is valid since, when starting from the node at the top of the lattice shown in Figure 1, the first encountered node containing *dn* contains also *ps* and *ss*. A consequence of Proposition 1 is that the IRs that can be generated from a node N=(X,Y) are of the form Q → R, where Q ∈ (Y), R = Y - Q, and Q is *consistent* with the parents of N; i.e., there is no parent P=(W, Z) such that Q ⊆ Z. A further consequence of Proposition 1 is that in order to find the complete set of IRs that hold in a context it is sufficient to collect the rules generated from each node of the lattice associated with the context. However, the resulting set of IRs will, in general, be highly redundant, so we are interested in finding a reduced cover. One m[illegible] represented by the fact that in the set of rules associated with ea[illegible]de there m[illegible]e rules that can be obtained from some other rule *r* in the same set by shifting attrib[illegible]irs from the right-hand side (*rhs*) of *r* to its left-hand side (*lhs*); i.e., by *projectivit*[illegible] by [illegible]*n*, i[illegible] terms of the Armstrong's axioms as described in Maier (1983). Fo[illegible]ample[illegible] nod[illegible] intent (*sl df my ps*) in the concept lattice shown in Figure 1 generates[illegible]IR (*sl*[illegible]*df my ps*), but it also generates the IRs (*sl df*) → (*my ps*), (*sl my*) → (*df ps*), (*sl ps*) → (*df*[illegible] and [illegible] *ps*) → (*my*), all of which can be derived from the former by attribute-value pairs s[illegible]pe with this kind of redundancy, it is convenient to think of the set (Y), that co[illegible] all the possible conjunctions of attr[illegible]the intent of the node in question, equipped with a partially-[illegible]ed relatio[illegible]ned by standar[illegible] inclusion over the elements of (Y); i.e., $c_1$ is more genera[illegible]than $c_2$ if $c_1$[illegible]$c_1 \supset c_2$).

From Propositio[illegible]nd th[illegible]ce[illegible]s, it can be easily proved the following proposition.

*Proposition 2*. The s[illegible]f IRs[illegible]ociated with node N=(X,Y) that are nonredundant with respect to attribute-va[illegible]ng are of the form *lhs* →*rhs*, where *lhs* is a most general element of (Y) that is co[illegible]nt with the parents of N, and *rhs* is given by Y - *lhs*.

# 4. INFERRING IRS FROM CONCEPT LATTICES

In this section we first describe an algorithm for inferring a cover for the set of generalized IRs that hold in a context from the concept lattice associated with the context. Then we study the computational complexity of the algorithm, and finally present an experimental comparison with Godin and Missaoui's IR-finding algorithm.

## 4.1. Description of the algorithm

The outer loop of the algorithm iterates on the concepts in the lattice. For [illegible]cept, [illegible] algorithm first checks whether the concept's extent gives sufficient supp[illegible] respe[illegible] which case it invokes the procedure *Find-IRs-for-node-N*, for determining [illegible] IRs associated with the concept. The procedure must solve the problem stated in P[illegible]sition [illegible] it must find all the most general subsets of Y that are consistent with the in[illegible] of th[illegible]ent nodes of N. This is a computationally difficult problem, for which we have [illegible]ted a[illegible]hod based on incremental pruning of the hypothesis space, reminiscent of an ap[illegible]h to [illegible]ept induction proposed by Mitchell (1982) and Mellish (1991). We can take tak[illegible] the partial ordering relation defined by generality over the elements of (Y), because [illegible] ordered set is well formed with respect to the task at hand. In particular, if an elem[illegible]of (Y) is consistent (with the parents of N) then all its more specific elements are also consi[illegible] conversely, if an element is inconsistent then all its more general elements are also inco[illegible]ent. We examine one parent at a time, maintaining and updating a G-set containing all t[illegible]ost general elements that are consistent with the parents that have been seen so far. The G-[illegible]cts as an upper boundary for the hypothesis space: more specific hypotheses are consiste[illegible]hile more general hypotheses are inconsistent and do not need be considered. The G-set is [illegible] in the following way. Initially, it contains the most general elements of (Y); i.e., th[illegible] formed by si [illegible] attrib [illegible] alue pairs. Then, for each parent of N, the procedure deletes all the elements that [illegible] ncons [illegible] t with the parent from the G-set, and replaces each of them by its most general [illegible] ializations (a specialization of an element G in G-set is any element of (Y) that is m [illegible] ecific than G); finally, it removes the consistent specializations that are more specific than or equal to some other elements in G-set.

From a computational point of view, the crucial step is the determination of the most general consistent specializations. While finding all the most general specializations of a given element is straightforward (i.e., it is sufficient to add a single attribute-value pair in all possible ways to the element), a potentially-demanding aspect is the number of minimal specialization steps that must be performed before each specialization becomes consistent. Fortunately, it turns out that all most general consistent specializations of an element G of G-set belong to the set of children of G. This result follows from the observation that (i) in the set of children of the elements of G-set there is at least one consistent specialization (unless the node N generates no rule at all), and (ii) more specific consistent conjunctions are supersets of some consistent element contained in the set of children of G-set. Therefore, for each element of G-set that needs be specialized, it is sufficient to generate and test only the children of G. As more and more parents of N are examined the boundary G-set moves down and the hypothesis space shrinks, until it may become empty, in which case the set of rules generated from N is empty. Table 2 gives the complete description of the cover-finding algorithm.

To see the algorithm at work, consider the determination of the set of IRs associated with the node with intent (*ss df my pl*) in the concept lattice shown in Figure 1, assuming $\alpha < 1/9$. The procedure *Find-IRs-for-node-N* initially sets G-set to {*ss, df, my, pl*}, and then iterates on the two parents of the node. Examination of parent (*ss my*) causes the following actions to be performed on the elements of G-set. *Ss* is specialized into three candidate conjunctions (*ss df*), (*ss my*), (*ss pl*), none of which is then confirmed by *Confirm-Specialization*; *df* is added to G'-set, *my* is specialized into three unconfirmed conjunctions (*my ss*), (*my df*), and (*my pl*), and *pl* is added to G'-set. The G-set consistent with the first parent is therefore {*df, pl*}.

Examination of parent (*df my pl*) updates the elements of G-set in the following way. *Df* is specialized into (*df ss*), (*df my*), and (*df pl*), the first of which is confirmed and added to G'-set; *pl* is specialized into (*pl ss*), (*pl df*), and (*pl my*), the first of which is confirmed and added to G'-set. The final G-set returned for the node at hand is therefore {(*df ss*), (*pl ss*)}; the corresponding rules are $(ss\ df) \rightarrow (pl\ my)$ and $(ss\ pl) \rightarrow (my\ df)$.

The complete set of IRs returned by the algorithm is the following.

$(pl) \rightarrow (my\ df)$

$(df) \rightarrow (my)$

$(ss\ ps) \rightarrow (dn)$

$(dn) \rightarrow (ps\ ss)$

$(mn) \rightarrow (ps\ dn\ ss)$

$(my\ dn) \rightarrow (ps\ ss)$

$(my\ ss\ ps) \rightarrow (dn)$

$(df\ ps) \rightarrow (my\ sl)$

$(sl) \rightarrow (ps\ my\ df)$

$(sm) \rightarrow (pl\ my\ df)$

$(ss\ pl) \rightarrow (my\ df)$

$(ss\ df) \rightarrow (pl\ my)$

The cover returned by the algorithm will, in general, contain many redundant rules, as in this case. Although Godin and Missaoui (1994a) have shown that it is possible to remove some redundant rules from it by further exploiting the information contained in the lattice, in order to get a nonredundant cover we must resort to more systematic approaches. One simple way is to apply the procedure for removing redundant FDs described by Maier (1983) on page 73 to IRs. Basically, it consists of checking whether each rule is implied by the remaining ones, where this inner test can be implemented efficiently, for instance through a marker passing algorithm, to ensure that its time complexity is at most proportional to the number of rules to be examined.

By applying the described method to the set of IRs returned by our *Find-IRs* algorithm, a nonredundant cover for the planet context was obtained:

$(pl) \rightarrow (df)$

$(df) \rightarrow (my)$

$(ss\ ps) \rightarrow (dn)$

$(dn) \rightarrow (ps)$

$(dn) \rightarrow (ss)$

$(mn) \rightarrow (dn)$

$(df\ ps) \rightarrow (sl)$

$(sl) \rightarrow (ps)$

$(sl) \rightarrow (df)$

$(sm) \rightarrow (pl)$

$(ss\ df) \rightarrow (pl)$

## 4.2. Complexity of IR inference

In this section we consider the computational complexity of IR inference. We consider the three main parameters of the problem: the number of objects $n$, the number of attributes $m$, and the support threshold $\alpha$. We start by discussing how the growth of each parameter affects the size of the set of IRs that hold in a relation.

As the number of objects grows, the number of IRs may decrease or it may increase. To explain this, consider that the introduction of a new object may result in new IRs between combinations of attribute-value pairs that had not been seen before; but it may also disconfirm IRs that held previously. Furthermore, things are made slightly more complicated by the fact that when some formerly-valid rule in the cover becomes invalid as a consequence of the introduction of a new object, it may be replaced by more rules (e.g., P→Q may be replaced by PR→Q and PS→Q); thus, the size of the cover may increase even while the set of IRs that hold in a context decreases. In fact, we have observed that an increase in the number of objects $n$ usually results in a noticeable increase in the cover size, at least for small values of $n$.

The number of IRs that hold in a relation grows monotonically with respect to the number of attributes in the relation. Mannila and Räihä (1994) have shown that for some relations over $m$ attributes all the covers of *FD* sets are of exponential size in $m$. Thus, the covers of the *IR* sets of those relations are - *a fortiori* - of exponential size in $m$. On the other hand, experience with real data sets confirms that covers of IR sets may indeed be exponential in the number of attributes (see Table 4). Admittedly, this is a serious problem in many situations; however, the growth of the number of IRs may be dramatically reduced by introducing the support threshold $\alpha$, as illustrated below.

We did some experiments to see how the size of the cover determined by the *Find-IRs* algorithm varies with respect to $\alpha$. We used an artificially-generated data set ("Random-300"), containing 300 objects and 10 attributes, with 10 randomly-assigned values per attribute, and eleven machine learning benchmarks drawn from the UCI repository (Murphy and Aha, 1995), whose main features are described in Table 4. Numeric attributes were treated as if they were nominal;[2] for the missing values, we used the overall modal (most frequent) value for nominal attributes and Boolean features and used the overall mean for numeric attributes. The results are shown in Table 3. The most important finding is that it is usually possible to have very small and manageable sets of rules by choosing small values of $\alpha$. As shown in Table 3, when passing from ungeneralised IRs ($\alpha=0$) to generalised IRs with $\alpha=0.01$, in seven out of the twelve benchmarks the number of IRs reduces by more than 99.5%. The drop in the number of rules is amazingly sharp for the Random-300 data set, but this is not surprising. Under the hypothesis of uniform distribution, and provided that the object description space is much larger than the number of objects present in the database (i.e., $p^m >> n$, where $p$ is the number of values per attribute), it is actually very unlikely that the same combinations of attribute-value pairs will occur in more than one object. Solar Flares is the only benchmark in which the reduction in the number of rules is slow, but this data set is anomalous because it contains many identical objects.

## 4.3. Complexity of lattice-based inference of IRs

We first analyse the complexity of the *Find-IRs* algorithm described in Table 2. The main loop of the algorithm iterates on the nodes in the lattice. Each iteration invokes the *Find-IRs-for-node-N* procedure. For each parent of the node, we need to update the G-set containing the most general consistent conjunctions, which requires time at most proportional to $g^2m$, where $g$ is the largest size of G-set and $m$ is the number of attributes. As a consequence, the time complexity of the *Find-IRs* algorithm is at most proportional to $Ng^2mq$, where $N$ is the number of nodes and $q$ is the largest number of parents per node.

[2] All experiments reported in this paper were repeated discretizing the values of numeric attributes into ten equal-length intervals. The results were different but consistent with those reported here.

The introduction of the support threshold $\alpha$ has a positive effect on the time complexity of the algorithm. Since the information about the statistical significance of the rules generated from each node is already encoded in the extent of the node itself, we are able to generate only the rules that comply with a given support threshold (by considering only the nodes whose extent contains a sufficient number of objects). Therefore inferring generalised IRs through a lattice conceptual clustering approach is always faster than inferring ungeneralised IRs; indeed, because the nodes that violate the threshold are those whose intent contains more attribute-value pairs, their pruning usually results in a dramatic improvement of the time complexity of the algorithm for rule extraction.

So far we have analysed the complexity of extracting the rules from a concept lattice. If we want to estimate the overall complexity of inferring IRs using a concept lattice-based approach, we must consider also the time necessary to build the concept lattice itself. For this purpose, we can use GALOIS, an incremental algorithm for concept lattice construction (Carpineto and Romano, 1996). The basic working of GALOIS consists of computing the intersections of each object with every concept in the current lattice, then adding such intersections to the lattice and finally placing edges in the Hasse diagram appropriately. Each update performed by GALOIS takes time proportional to the number of concepts in the lattice being updated (Carpineto and Romano, 1996), thus the key parameter for estimating its time complexity is the size of the lattice. By definition of concept lattice, the number of concepts is theoretically upper bounded by $2^n$, where $n$ is the number of objects in the context. In fact, Oosthuizen 1994 gave an example showing that with $n$ objects (each having $n$ binary attributes) one can actually generate $2^n$ nodes; it is sufficient to consider a $n \times n$-dimensional context containing ones in all positions, but zeroes along the diagonal. Thus, the incremental construction of the lattice by GALOIS takes time at most proportional to $n2^n$, while the whole method for finding the set of IRs that hold in a relation, consisting in the application of the *Find-IRs* algorithm to the output produced by GALOIS, has the following worst case time complexity.

*Proposition 3*. The concept lattice-based method for finding a cover of the set of IRs that hold in a relation can operate in time $\Theta(2^n(n + g^2mq))$.

If the number of attributes per object is fixed and bounded by a constant $K$, as usually happens in practical applications, the number of nodes in the lattice is bounded by $2^kn$ and the worst case time complexity is $\Theta(2^k (n^2 + g^2mq))$. Although both of these upper bounds may be very large, the situation is, on the average, much better. Experience with several natural databases and theoretical analysis using a uniform distribution hypothesis suggest that, at least when the number of attributes is limited, the growth of the number of nodes is at most quadratic, and usually linear, with respect to the number of objects (Carpineto and Romano 1996).

For domains that frequently change over time, it would be more efficient to have an algorithm that can incrementally extract the set of rules that hold in a relation without recomputing it from scratch each time a new object is introduced in the relation. Godin and Missaoui (1994a) showed how it is possible to extend an incremental algorithm for concept lattice construction to generate both the concepts and the rules. We might take a similar approach with GALOIS. Although we have not fully worked out such a refinement, the clue seems to add the rules associated with any new node generated by each object, and to delete those rules associated with the children of the new node in the Hasse diagram that have become redundant after the introduction of the new object.[3]

In the next section we compare the *Find-IRs* algorithm for extracting the IRs from a concept lattice to an earlier (nonincremental) algorithm described by Godin and Missaoui (1994a).

## 4.4. Comparison with Godin and Missaoui’s IR-finding algorithm

[3] This holds for simple IRs. The incremental determination of generalized IRs seems to be more difficult, because the introduction of a new object, in addition to create new nodes, may also change the extent of old nodes without modifying their intent.

For the sake of conciseness, we refer to our algorithm for IR extraction as $\mathrm{CRAAL_{IR}}$ (Carpineto, Romano and d'Adamo's ALgorithm for Implication Rules), and to Godin and Missaoui's algorithm as GMAL. GMAL accepts the same input (i.e., a concept lattice) and produces the same output as $\mathrm{CRAAL_{IR}}$ (with $\alpha$=0). The difference between the two algorithms lies in the procedure for finding the rules associated with each node N=(X,Y). GMAL systematically generates the power set of Y, in ascending cardinality order, and, for each set H in $2^Y$, it updates the set of IRs $\Delta$, provided that (i) H is consistent with the parents of N and (ii) there is no rule in the current set $\Delta$ whose *lhs* is more general than H. Assuming that the number of attributes is bounded by a constant *m*, Godin and Missaoui have shown that the time complexity of their algorithm is linear in the number of concepts with a large constant. A similar situation holds for $\mathrm{CRAAL_{IR}}$, because both $g$ and $q$ are bounded by the constant $2^m$; in particular, $g$ is bounded by the largest number of unordered conjunctions that can be found in the conjunction space; i.e., $\max_k \binom{m}{k} = \binom{m}{\lfloor m/2 \rfloor} < 2^m$.

However, in practice the behavior of the two algorithms may be significantly different, so it is useful to compare their performance in some real domains.[4] We implemented MGAL ourselves, taking great care to ensure that the implementation was efficient. In particular, for the generation of the elements in $2^Y$ we used a trie-like enumeration order. All attributes are placed in a fixed order prior to the generation, and then, to expand the current element H, only attributes appearing after the highest ranked attributes in H may be added.[5] Furthermore, as soon as the above condition (ii) is not satisfied, the algorithm avoids generating and testing all the elements of $2^Y$ that are more specific than H.

For the experiment, we used the same twelve data sets used in the experiment described in section 4.2. We used GALOIS to build the lattice associated with each data set, measuring the

[4] All the programs tested in this paper are written in Common Lisp. For the experiments we used a SPARCstation 20 equipped with 64 Mbytes of RAM.

number of nodes in the resulting lattice and the time necessary to compute it. Then we ran CRAAL$_{IR}$ and GMAL (with $\alpha$=0) on each lattice, computing the following variables: (a) size of the cover determined by the algorithm (equal for both algorithms), (b) size of a nonredundant cover (equal for both algorithms), (c) CPU time necessary to find the (redundant) cover.

The results are shown in Table 4. “Size of nonredundant cover” columns labeled “NA” indicates that the procedure for computing the nonredundant cover ran into computational barriers, due to memory fragmentation, and therefore the datum was not available. For some data set, the size of the cover returned by CRAAL$_{IR}$ is apparently smaller than the size of the nonredundant cover; this is due to the fact that the rules in the nonredundant cover have singleton right sides (i.e., every right side is a single attribute-value pair) while the redundant cover may contain compound rules, i.e., such that their right side contains more than one attribute-value pair.

These experiments indicate that the performance of CRAAL$_{IR}$ is markedly superior to that of GMAL. CRAAL$_{IR}$ did dramatically better than GMAL on every benchmark. In particular, in seven out of the twelve data sets the CPU time reduces by more than 95%, and in the remaining five it reduces by at least 80%. Not only do these results confirm the clear superiority of CRAAL$_{IR}$ over GMAL, but they also give some insights into the effort necessary for rule extraction compared to the time needed for lattice construction. As shown in Table 4, the CPU time of CRAAL$_{IR}$ never exceeds one fourth of the time necessary to build the lattice, and in nine out of twelve benchmarks it is less than 12%. With CRAAL$_{IR}$ therefore, the time for rule extraction becomes a small fraction of the time for lattice construction. Although this does not solve the problem of lattice construction itself, it is a very encouraging result for the feasibility of the whole lattice-based approach to dependency inference.

Before ending this section we must emphasize that an alternative IR-finding algorithm has been recently presented by Ziarko and Shan (1996). Similar to our approach, their method is based on a particular representation of the input relation, called decision matrix, from which the

[5] This technique has also been used by Schlimmer (1993), in a similar context.

rules are then extracted. The cover output by Ziarko and Shan's method is less redundant than ours, in that it contains left-reduced (*minimal*, according to their terminology) rules; i.e., such that no attribute-value pair on any left side is redundant.[6] However, the complexity of the method, as also noted by the authors, seems to be very high; in fact, no evidence is reported in the paper that in some real domains such an approach would be feasible.

## 5. INFERRING FDS THROUGH CONCEPT LATTICES

In this section we show that the method described in section 4 to find IRs can also be applied, with some modifications, to the problem of inferring FDs. We first illustrate the adapted method, then present a study of its computational complexity, next review related work on FD inference, and finally offer an experimental comparison between our algorithm and another, well-known FD-finding algorithm proposed by Mannila and Räihä.

### 5.1. FDs as IRs in transformed contexts

One might think that one simple method for finding a cover of the set of FDs that hold in a context would be to transform the cover of IRs determined, for the same context, by the algorithm described in section 4.1. Such an approach, however, would be highly inefficient, in that it is not possible, in general, to derive a cover of the FD set from a cover of the IR set without applying the inference axioms. As an example, consider again the context shown in Table 1. In order to derive the FD *(size distance)* $\rightarrow$ *(period)* we should ascertain that the following IRs hold: *(ss dn)* $\rightarrow$ *(ps)*, *(sl df)* $\rightarrow$ *(ps)*, *(sm df)* $\rightarrow$ *(pl)*, *(ss df)* $\rightarrow$ *(pl)*. However, of these four IRs, only the fourth is explicitly present in the cover found by the algorithm (see section 4.1.). Missaoui and Godin (1994b) showed that such a direct derivation is possible for complete covers, as long as the IRs are not left-reduced during their generation. Here we show an

[6] More precisely, a set $\Delta$ of IRs are left-reduced if for no $Q \rightarrow R$ in $\Delta$ and proper subset S of Q is $\Delta - \{Q \rightarrow R\} \cup$

alternative and more convenient way to cast the FD inference problem as an IR inference problem.

Following an idea expressed by Wille (1992), it is possible to deduce from a context C a transformed context $C_t$ whose IRs are exactly the FDs that hold in C. The transformed context $C_t$ is a triple (O', A, I'), where O' is the set of all two-elements subsets of O and {f,g}I'A implies that a(f)=a(g). Recalling the definition of functional dependency, one can see immediately that for H, K $\subseteq$ A, K is functionally dependent on H in C if and only if H $\rightarrow$ K is an IR in $C_t$. Thus, the cover for the IRs that hold in $C_t$, determined by the algorithm described in section 4.1, can also be taken as a cover for the FDs that hold in C. To illustrate, consider the context C shown in Table 1. The corresponding transformed context $C_t$, built from the distinct objects in C (e.g., Mercury and Venus are labelled as object 1, Earth and Mars as object 2, etc.) is shown in Table 5. We should emphasize that that while the objects in C are necessarily described by a *fixed* number of *multi-valued* attributes (i.e., attribute-value pairs), the objects in $C_t$ are described by a *variable* number of *single-valued* attributes (i.e., just attributes, without values). In fact, an untransformed context is defined as a quadruple, while its transformed context is defined as a triple. With some precautions (Carpineto and Romano 1996), the theory of concept lattices can accomodate both situations. In Figure 2 we show the concept lattice built from the transformed context $C_t$.

By running the algorithm described in section 4.1 on the concept lattice shown in Figure 2, we get the following (redundant) set of FDs for the context shown in Table 1:

(*size distance*) $\rightarrow$ (*period*)

(*size period*) $\rightarrow$ (*distance*)

(*size moon period*) $\rightarrow$ (*distance*)

(*size distance moon*) $\rightarrow$ (*period*).

{S$\rightarrow$R} equivalent to $\Delta$.

It is worth noting that the last two dependencies are generated from the bottom of the lattice in Figure 2. In effect, the determination of the FD set, unlike that of the IR set, requires that the dependencies produced by the lattice bottom, which hold vacuously, should be considered.

## 5.2. Complexity of lattice-based inference of FDs

The method described in section 5.1 has been implemented as a two-step procedure: the first step builds the transformed lattice, the second extracts the rules from it. The first step is carried out by applying GALOIS to the transformed context derived from the given context; in particular, we generate one transformed object at a time and use it as an input to GALOIS, which updates the corresponding (transformed) lattice. Once the transformed lattice has been built, we extract the rules from it using the algorithm described in section 4.1.

The complexity analysis presented in section 4.3 for the extraction of IRs can be applied also for the FDs, provided that we consider the number of concepts present in the lattice built from the *transformed* context rather than from the original context. In particular, we need a characterisation of the growth of the number of concepts $N_T$ present in the transformed lattice as a function of the number of objects $n$. First of all, it should be noted that $N_T$ grows monotonically with respect to $n$, because the addition of a new object increases the number of transformed objects, which, in turn, increases $N_T$.[7] The number of transformed concepts $N_T$ is still theoretically upper bounded by $2^n$, because the transformed objects (at most $\binom{n}{2}$) along with their possible combinations cannot exceed $\sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{2k} = 2^{n-1}$. As in the untransformed case, one can use a *nxn* dimensional context containing ones in all positions but zeroes along the diagonal to prove that a transformed lattice with $2^n$ nodes can be actually generated.

[7] The growth of $N_T$ is not inconsistent with the fact that the number of FDs that hold in a relation decreases monotonically with respect to the number of objects (Savnik and Flach 1993), because the addition of a new node to a lattice usually reduces the number of rules generated by the other nodes.

Considering that the construction of the transformed lattice by GALOIS takes at most $n^2 2^n$, for the worst case time complexity of whole FD-finding method the following result holds.

*Proposition 4.* The concept lattice-based method for finding a cover of the set of FDs that hold in a relation can operate in time $\Theta(2^n(n^2+g^2mq))$.

In practice, relations that have inherently large transformed lattices should be rare. In the following, we study the growth of $N_T$ in a simple case, i.e., under the assumption of uniform distribution of the attribute values over a discrete range. Let *m* the number of attributes, *p* the number of values per attribute. If the attribute values in the original context are uniformly distributed, then, in the transformed context, each attribute will be assigned to each of the *n(n-1)/2* transformed objects with (constant, independent) probability *1/p*. Under a similar hypothesis, Godin *et al.* (1986) pointed out that the number of attributes per (transformed) object follows a binomial distribution with a mean value of *m/p* (note that, in the transformed relation, the probability of occurrence of 0-attribute objects will be $\binom{m}{0}\left(\frac{p-1}{m}\right)^m$, the probability of occurrence of 1-attribute objects will be $\binom{m}{1}\left(\frac{p-1}{p}\right)^{m-1}\frac{1}{p}$, etc), and they derived a formula that gives the expected number of nodes in the corresponding lattice. The formula, adapted to the transformed context, is the following:

$$N_T = \sum_{i=0}^{n(n-1)/2} \sum_{j=0}^{m} \binom{n(n-1)/2}{i}\binom{m}{j}\left(\frac{1}{p}\right)^{ij}\left(1-\frac{1}{p^i}\right)^{m-j}\left(1-\frac{1}{p^j}\right)^{n(n-1)/2} \qquad (1)$$

We have plotted, in Figure 3, $N_T$ as a function of *n* for five pairs of values for (m,p): (7,7), (20,20), (20,50), (50,20), (50,50). The results suggest that the growth of the number of nodes in the transfomed lattice varies from linear (i.e., the curves obtained for (20,50) and (20,20)) to quadratic (i.e., the curve obtained for (50,20)) with respect to the number of objects. The results also show the two situations in which the space complexity moves towards the upper bound; this is when the ratio between $m$ and $p$ increases, or, less markedly, when both $m$ and $p$ grow by a same factor and the number of objects is not too small. The lower curve in Figure 3, relative to the pair (7,7), represents a limit situation. For a fixed value of $m$, given that each object in $C_T$ is described by single-valued attributes, $N_T$ has an upper bound of $2^m$ ($2^7$=128 nodes, in our case); therefore, after a certain point the lattice is saturated (i.e., it contains 128 nodes) and $N_T$ remains constant as new objects are added.

We have to emphasize that these results hold for random descriptor assignment. While we have chosen this model because it is simple to analyse, we are aware that it may not be a good model for attribute values. In fact, we observed that real databases usually produce much smaller transformed lattices, because the regularities that they exhibit typically limit the number of non-empty transformed objects and produce many identical object descriptions, whereas with random value assignments there is a large number of distinct transformed objects, which increases the likelihood and diversity of intersections, and therefore the size of the transformed lattice. In particular, Table 6 shows that for a number of natural data sets the size of the transformed lattice is comparable to the number of objects, and it is sometimes a small fraction of it.

From the point of view of incrementally building the transformed lattice using GALOIS, this suggests that for real data sets the time complexity with respect to the number of objects may be dramatically reduced. In fact, considering that the update of the lattice takes place only if the transformed objects are nonempty and distinct, the number of invocations of GALOIS is usually much smaller than the $n(n-1)/2$ transformed objects that can be theoretically generated from the $n$ initial objects; secondly, the time required by each invocation of GALOIS is, in practice, limited, because it depends on the size of the current transformed lattice. However, we should also notice that, as will be seen more fully in Section 5.4.2., there are some atypical, yet real, data sets for which the complexity of transformed lattices seems to grow quadratically with respect to the number of objects, similar to the upper curve of Figure 3.

Before ending this section, it is useful to look into the relationship between the size of a normal lattice and the size of a transformed lattice for a given context. Even though one can easily construct some special context such that its transformed lattice has more nodes than its normal lattice, in practice such a situation would be rare. Transformed lattices are usually much smaller than normal lattices, because transformed contexts tend to have fewer rows, fewer columns, and a higher sparseness than untransformed contexts. This observation has been confirmed by all the data sets that we used in our experiments; if we compare the data shown in Table 4 and Table 6 we see that transformed lattices are significantly smaller than untransformed lattices.

## 5.3. Related work on FD inference

The study of functional dependencies arising from data has been addressed in various fields, including rough sets, artificial intelligence, and database theory.

### 5.3.1. Reducts

Using the framework of the rough sets theory (Pawlak, 1991), a bunch of useful concepts for discovering and analysing data dependencies have been developed (Ziarko, 1991; Shan *et al.*, 1995; Skowron and Polkowski, 1997). In this context, of primary importance is the notion of reduct, which has been explored, among others, by Orlowska and Orlowski (1992), Skowron and Rauszer (1992), and Skowron and Polkowski (1997). The set of reducts associated with a relation can be seen as particular FDs; namely, those of size equal to the number of attributes in the relation, and such that they are nonredundant with respect to attribute-value pairs shifting, as defined above. In order to determine the set of reducts, we can use a procedure based on a particular representation of the input relation, called discernibility matrix, from which a set of dependencies is then extracted and simplified using Boolean reasoning.[8] On an abstract level, discernibility matrix and transformed context support dual forms of reasoning, one based on the differences between the pairs of objects in the relation and the other based on their similarities. However, a direct utilization of the reduct sets approach to generate the set of FDs that hold in a relation seems to be difficult, because the set of reducts do not cover the complete set of holding FDs; in addition, the computational complexity of the procedure based on Boolean reasoning seems to be very high even for databases of modest size, although some ways of mitigating the latter problem have been proposed, such as performing incremental determination of the set of reducts (Orlowska and Orlowski 1992), or using approximations of relations (Shan *et al.*, 1995) or approximations of reducts (Skowron and Polkowski, 1997).

## 5.3.2. Determinations

Functional dependencies have been also studied in artificial intelligence, where they are sometimes called determinations, even though the notion of determinations, as first proposed by Russel (1986), is more inclusive than the one considered here. A common approach is to seek the set of left-reduced FDs that hold in a relation[9] by performing, for each attribute *a*, a complete search through the semi-lattice containing all possible FDs having *a* as right-hand side (Schlimmer, 1993; Savnik and Flach, 1993; Bell and Brockhausen, 1995). The search is complete but it may be not exhaustive, due to various pruning criteria driven by the search space ordering. Usually, the complexity of such algorithms is subquadratic with respect to the number of objects, but it takes exponential time with respect to the number of attributes *m* even when the FD set that hold in a relation is small, unless we restrict the search only to FDs of size less than *m*.

[8] Ziarko and Shan (1996), mentioned in Section 4.4, have used notions derived from those presented here to find a cover for the set of IRs that hold in a relation.

[9] Analogously to IRs, a set F of FDs are left-reduced if for no $Q \rightarrow R$ in F and proper subset S of Q is $\Delta - \{Q \rightarrow R\} \cup \{S \rightarrow R\}$ equivalent to F.

### 5.3.3. Related work in database theory

The FD inference problem has been deeply studied in database theory, in particular by Mannila and his co-workers. They found theoretical lower bounds for the worst case time complexity of inferring FDs; mainly, they showed that this problem may require time proportional to $n \log n$, where $n$ is the number of objects, even for a relation containing two attributes, and that for some relations over $m$ attributes all the covers of *FD* sets are of exponential size in $m$, as already mentioned in section 4.2. They also proposed three algorithms for inferring FDs, with the main goal of achieving a polynomial-time solution with respect to the number of objects and the size of the smallest cover for the dependency set of the relation. Since none of these algorithms has, in principle, better computational performance than the others, here we concentrate on the algorithm given first by Mannila and Räihä (1986), and later reviewed by the same authors in (Mannila and Räihä, 1994) as "Algorithm 2", which is more simple and better understood. For each attribute $a$, it computes the set *lhs*(a), containing the left sides of all left-reduced FDs having $a$ as right side. The core of the algorithm is based on pairwise comparison of objects: examination of each pair may leave the current *lhs* set unchanged, or it may result in a specialization of some elements in *lhs*(a). Mannila and Räihä (1994) showed that their algorithm can operate in time $\Theta(m^3 n^2 (\max\|lhs\|)^2)$, where $\max \|lhs\|$ is the size of the largest *lhs* collection (of the subrelations of the given relation).

Although we chose to use "Algorithm 2" for the experimental comparison with our approach (see below), it is useful to briefly consider also the second method for FD generation ("Algorithm 3" and "Algorithm 4") described in (Mannila and Räihä, 1994). Building on earlier work by Demetrovics and Gyepesi (1981), the starting point is the extraction, for each pair of objects in the relation, of the set of attributes whose values agree/disagree on the pair. This is, in fact, similar to transformed context and discernibility matrix; but then Mannila and Räihä use this information in a different way. For each attribute $a$, they derive from the disagree sets a collection of subsets of "necessary" attributes; i.e., such that each subset contains at least one attribute that should appear in any *lhs*(a). Then, they find the minimal attribute subsets that intersect all the subsets in the collection; i.e., a minimal hypergraph transversal. They showed that in this way we obtain all the left-reduced FDs with *rhs*=a. The overall complexity of this method depends on the time necessary to perform its second step, which is, however, difficult to characterize.

The use of set-theoretic techniques for the dependency inference problem has been also advocated in an earlier work (Cosmadakis *et al*, 1986). The authors showed that reasoning over the sum and product of the partitions of objects that can be induced over the values of each attribute leads to the discovery of a general form of database constraint, named partition dependency (PD), which includes the strict FD. Then, they concentrated on the problem of PD implication, finding interesting relationships with problems and notions developed in the theory of lattices. On close inspection, the resemblance to our approach is rather superficial, because concept lattices and lattices, as used by Cosmadakis *et al.*, are quite distinct mathematical entities and they served for different purposes; in particular, Cosmadakis *et al.* did not consider the problem of finding the set of PDs (or FDs) that hold in a relation. Nonetheless, both approaches contribute to shed light on the algebraic nature of the dependency inference problem.

In general, the complexity of the algorithms that have been proposed for finding unrestricted FDs seems to depend in inverse ways on the two main input problem parameters. While there are algorithms that can identify the FD set that hold in a relation in subquadratic time with respect to the number of objects, like Schlimmer’s one, or in subexponential time with respect to the number of attributes (provided that the FD set is not inherently large), like Mannila and Räihä (1994)’s Algorithm 2, it is difficult to find an algorithm that exhibits both behaviours. Under this respect, our approach, more similar to the work done in the latter vein, does not represent an exception.

In the next section we experimentally compare the performance of our algorithm with that of Mannila and Räihä (1994)’s Algorithm 2.

## 5.4 Comparison with Mannila and Räihä's FD-finding algorithm

For the sake of conciseness, we refer to the FD-finding method based on concept lattices as $\mathrm{CRAAL}_{\mathrm{FD}}$ (Carpineto, Romano and d'Adamo's ALgorithm for Functional Dependencies), and to Mannila and Räihä (1994)'s Algorithm 2 as MRAL. We describe two experiments aimed at evaluating the performance of $\mathrm{CRAAL}_{\mathrm{FD}}$ versus MRAL in two different scenarios.

### 5.4.1 Static evaluation

For this experiment, we used the same data sets used for testing the IR-finding algorithms with the addition of the benchmark "Nursery". Nursery is described by 12960 objects and nine attributes, with no missing values; it has not been considered in Table 4 because the time necessary to build its (untransformed) lattice ran into computational barriers, due to memory fragmentation. We ran MRAL and $\mathrm{CRAAL}_{\mathrm{FD}}$ on each data set, measuring the following variables. For MRAL, the maximum size of the *lhs* set, the CPU time, the size of the cover, and the size of a nonredundant cover; for $\mathrm{CRAAL}_{\mathrm{FD}}$, the number of nodes in the (transformed) lattice, the time taken to build the lattice, the total CPU time, the size of the cover, and the size of a minimal cover. The mimimal cover was computed by applying Maier's procedure to the cover returned by the algorithms. To normalize the time measurements with respect to the number of dependencies produced, we included in the CPU time the time required to find a minimal cover. In practice, this last time was always a very small fraction of the total time because the covers output by the two algorithms were of limited size; even for Wdbc and Wpbc, the data sets with the largest redundant covers, the time necessary to find a minimal cover was negligible.

We show in Table 6 the results of the experiments. MRAL and $\mathrm{CRAAL}_{\mathrm{FD}}$ performed in a similar manner on only one data set, while they showed remarkably different performance on the other data sets. $\text{CRAAL}_{\text{FD}}$ beat MRAL on seven of the benchmarks, and lost on the remaining six. The results suggest that the performance of MRAL is probably poor when the number of attributes is not very limited, because the value of max||*lhs*||, although not directly related to the number of attributes**,** may grow larger (max||*lhs*|| is upper bounded by $\max_k \binom{m}{k} = \binom{m}{\lfloor m/2 \rfloor} < 2^m$) This was the case for the Wdbc and Wpbc data sets, although the excessively large performance gap on these two data sets deserves more attention. On close inspection, we observed that this result was also due to the numerical nature of their attributes. Without discretization, many objects did not match any other object in the data set, which implied a huge number of valid dependencies and made MRAL’s pruning less effective; on the other hand, this situation suited $\text{CRAAL}_{\text{FD}}$ characteristics very well, because the transformed context was very small, the transformed lattice contained very few nodes, and the many rules with a same left side were naturally “compounded” in single rules. Table 6 shows also that the number of objects seems to be less important for MRAL’s performance, because MRAL did generally better than $\text{CRAAL}_{\text{FD}}$ on the more numerous data sets, although there is some significant exception (e.g., Abalone).

Differently from MRAL, the performance of $\text{CRAAL}_{\text{FD}}$ is more difficult to characterize in terms of the parameters describing the data, such as number of objects and number of attributes. As shown in Table 6, the total CPU time of $\text{CRAAL}_{\text{FD}}$ was almost entirely due to the time necessary for building the transformed lattice. The results show that the lattice may be built efficiently even when there are relatively many attributes and relatively many objects, although the numbers of objects appears to be a more critical factor for the $\text{CRAAL}_{\text{FD}}$ complexity. On the other hand, there are some small data sets that produce large lattices. The key observation here is that while the lattice makes certain relationships between the data explicit, thus facilitating FD extraction, it also reveals other regularities and spurious similarities that may be difficult to acquire and represent. More precisely,[10] the lattice grows large when (a) there are many subsets of objects having some attributes in common, and (b) the attributes in common to a class of objects do not coincide with (or are not a superset of) the attributes shared by some other class. Depending on the presence (absence) of these regularities, the resulting lattice may be hard (easy) to compute even for small (large) contexts. Consistently with this observation, the results show that there is no direct relation between the time complexity of $\text{CRAAL}_{\text{FD}}$ (in particular, the size of the lattice) and the number of FDs that hold in a context. Small lattices may produce small covers as well as large covers. Likewise, large lattices may produce large covers, but they may also give rise to small covers. The last phenomenon may seem a bit surprising, but we must consider that the number of rules depends not only on the number of nodes but also on the number of rules associated with each node.[11]

The covers produced by the two algorithms are equivalent, but they may be of very different size. The covers output by $\text{CRAAL}_{\text{FD}}$ tend to be smaller because they may contain compound dependencies (i.e., such that their right side contains more than one attribute); incidentally, this is also the reason why, analogously to what happened with IRs, the size of the cover returned by $\text{CRAAL}_{\text{FD}}$ is sometimes smaller than the size of the nonredundant cover. However, the results of the experiments show that it is sometimes the case that the covers produced by MRAL are actually smaller than $\text{CRAAL}_{\text{FD}}$, because the rules found by MRAL, unlike those found by $\text{CRAAL}_{\text{FD}}$, are left-reduced. As the covers found by the two algorithms were typically different, also the nonredundant covers derived from them presented some difference. In particular, we observed that even for the cases when the two minimal covers contained the same number of rules (see Table 6), the sets of rules generated were not identical. This is not surprising, considering that the procedure for finding a minimal cover is sensitive to the order of presentation of rules and, thus, it would, in general, produce a different output even if the

[10] Recall that the lattice contains, by definition, *all* nonempty and complete intersections between the objects of a context.

[11] In the limit, the transformed lattice is saturated (i.e., it contains $2^{m-1}$ nodes). Each node has, as parents, all its possible maximally specific generalisations and the algorithm is unable to find any rules. The benchmark Solar Flares is a case in point. Its transformed lattice contains exactly $2^{12} = 4096$ nodes, while there are no FDs that hold in this data set.

initial redundant covers contained the same rules. The equivalence of the minimal covers produced by the two algorithms for each data set was double-checked by using the procedure described by Maier (1983) on page 72. Finally, we would like to emphasize that the different output formats of the two algorithms did not affect the subsequent generation of a minimal cover, because this was obtained with little effort in both cases. The important point is not the format of the redundant cover, but the complexity of the algorithm that computes the cover itself**.**

The results of the experiment described above give some insights into the overall performance of the two algorithms, but they do not help characterize their dynamic behavior. We now present another experiment aimed at analysing how the CPU time of the two algorithms empirically varies with the number of objects.

### 5.4.2 Dynamic evaluation

For this experiment, we used three data sets that were especially hard for at least one of the two algorithms: an artificially-generated data set ("Random"), containing 1000 objects and 10 attributes, with 10 randomly-assigned values per attribute, the Credit Appointment data set ("Credit"), containing 690 objects and 16 attributes, and the Congressional Voting Records data set ("Voting"), containing 435 objects and 17 attributes. We ran each algorithm on each data set, varying the number of objects (step = 25) and computing, in each run, the total time necessary to find a minimal cover. As in the static experiment, the time necessary to generate a minimal cover from the cover output by the algoritms was usually negligible, due to the limited size of the covers. We show, as an example, the results obtained by each algorithm on each complete data set. We report the total time (secs), the time for minimalization, the size of the cover generated by the algorithm, and the size of the minimal cover: Random $\text{CRAAL}_{\text{FD}}$ [83, 2, 144, 137], Random MRAL [39839, 12, 500, 137], Credit $\text{CRAAL}_{\text{FD}}$ [810, 62, 937, 198], Credit MRAL [55244, 23, 950, 199], Voting MRAL [804, 0, 0, 0], Voting $CRAAL_{FD}$ [59617, 0, 0, 0].

The results of the experiment, reported in Figure 4, show that the time complexity of MRAL usually grows with respect to the number of objects. There are, however, some cases in which the time required to compute the cover decreases as new objects are added. In fact, an increase in the number of objects may sometimes be accompanied by a decrease of $||lhs||$, which may result in better overall time performance. In particular, referring to the Voting data set, we observed that when passing from 175 objects to 200 objects the value of the maximum *lhs* decreases from 459 to 323, and, analogously, when the number of objects moves from 375 to 400 the value of the largest *lhs* collection decreases from 220 to 155. As for $CRAAL_{FD}$, the results suggest that its time complexity varies from linear to quadratic in the number of objects, consistently with the results of Section 5.2. The experimental analysis shows also that the growth is nearly monotonic. This is not surprising, because the construction of the lattice, which is by far the most expensive operation involved, was done by an incremental algorithm. The few exceptions visible in the curves can be explained considering that the time for rule extraction may also decrease as a result of the addition of new objects (because adding new objects implies not only increasing the number of nodes in the lattice but it also changes the links), and such a decreasing in the extraction time may sometimes compensate for the increase in the construction time.

On the whole, the results shown in Figure 4 confirm that neither algorithm is clearly superior to the other. Both algorithms have strengths but also limitations. In particular, Figure 4 points out one serious weakness of each method. On one hand, it shows that MRAL may be highly inefficient even for data sets containing a very limited number of attributes (10 in the Random data set and 16 in the Credit data set, respectively), because the value of *lhs* may grow very large and may become the key factor in the product that gives the algorithm complexity. For instance, we observed that for both the Random and the Credit data set the value of *lhs* grew quadratically with respect to number of attributes for most of the subsets of objects. This result is particularly important considering that MRAL, as stated explicitly by Mannila and

Räihä (1994), was specifically designed to perform well in those situations where the number of attributes is limited. On the other hand, Figure 4 suggests that there are some databases for which CRAAL$_{FD}$ performs very badly too. The Voting data set is a case in point. All its attributes are Boolean valued, so there are plenty of spurious similarities between the objects. Even for a few tens of objects, the lattice may contain many thousands of nodes, and it may therefore be hard to compute.

## 6. CONCLUSION

In this paper we considered the problem of finding reduced covers for two distinct, yet related, types of data dependencies: implication rules and functional dependencies. We presented a two-stage finding-cover method based on the construction of an intermediate clustered representations of the input relation, called concept lattice, from which rules are then extracted. We analysed the complexity of the proposed approach and compared it with other dependency inference algorithms. The results of the experiments were encouraging. On one hand, they showed that our algorithm for extracting rules from lattice-based representations of relations clearly outperformed an earlier algorithm. Furthermore, the results suggest the feasibility of the overall lattice approach to inferring functional dependencies, thus confirming and expanding the potentials of the lattice theory for knowledge discovery in databases, as yet not fully investigated.

Although some of the databases used in the experiments were reasonably-sized, their scale is still not large relative to operational situations. One crucial issue to investigate is whether our approach would scale up to large databases. As highlighted in the paper, the construction of the concept lattice prior to rule extraction is the most critical step. While the use of nonincremental algorithms (e.g., Bordat 1986) might reduce the time needed for lattice construction, the key limiting factor of this operation seems to be the size of the memory necessary to store the whole structure (Carpineto and Romano 1996), which is independent of the building algorithm. In order to alleviate this problem, we might apply the methods that have been developed to compose or decompose concept lattices (Ganter 1988), or, we could try to incorporate the use of the support threshold, or other pruning mechanisms aimed at reducing the complexity of rule extraction (Godin and Missaoui 1994a; Oosthuizen 1994), directly into the early phase of lattice construction. On the other hand, a more general and powerful answer to the inherent computational limitations of our approach relies on the use of sampling techniques, which are well understood and can achieve high performance (Casella and Berger, 1990; John and Langley, 1996). In this spirit, Mannila and Räihä (1994) suggested using a sample of the relation to efficiently find an initial set of dependencies, and then refine them using the whole relation.

A further anticipated difficulty for the application of our approach to real-life problems is the presence of noise in the data. Noisy data may cause two main problems. The first is the discovery of irrelevant or meaningless rules, although the use of the support threshold may greatly help reduce this inconvenience. The second problem is that the system may fail to produce interesting rules, because it does not permit exceptions (i.e., the rules must hold for all the objects in the relation). To deal with this issue it is necessary to allow for some form of nondeterministic or approximate dependencies (Kivinen and Mannila, 1995; Skowron and Polkowski, 1997), with the goal of handling noise while keeping reasoning capabilities. This is an avenue for future research.

Another important issue is the utilization of background knowledge in the process of rule formation. A simple form of background knowledge, expressed as conceptual hierarchies over the attribute values domains, can already be accomodated in the basic lattice framework (Carpineto and Romano 1996), but the resulting hypothesis space (i.e., conjunctive descriptions defined on structured attributes) is still rather limited. Using more powerful formalisms for representing concepts in the lattice would result in a richer rule description language; in particular, we plan to investigate the use of an extension of the concept lattice theory, recently presented by Wille (1995), that combines objects, attributes, and conditions under which objects may have certain attributes.

## ACKNOWLEDGMENTS

We would like to thank three anonymous reviewers for many useful comments and suggestions that greatly improved this paper. This work was carried out under the framework of the agreement between the Italian PT Administration and the Fondazione Ugo Bordoni.

## REFERENCES

AGRAWAL, R., and R. SRIKANT. 1994. Fast algorithms for mining association rules. *In* Proc. of the 20th VLDB Conference, Santiago, Chile, 487-499.

BELL, S., and P. BROCKHAUSEN. 1995. Discovery of data dependencies in relational databases. *In* Working Notes of the MLnet Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases. Crete, Greece, pp. 53-58.

BORDAT, J. (1986). Calcul pratique du treillis de Galois d'une correspondance. Mathématiques et Sciences Humaines, **96**:31-47.

CAI Y., N. CERCONE, and J., HAN. 1991. Learning in relational databases: an attribute-oriented approach. Computational Intelligence, **7**: 119-132.

CARPINETO C., and G. ROMANO. 1993. GALOIS: An order-theoretic approach to conceptual clustering. *In* Proc. of the Tenth International Conference on Machine Learning, Amherst, MA: Morgan Kauffmann, pp. 33-40.

CARPINETO C., and G. ROMANO. 1996. A lattice conceptual clustering system and its application to browsing retrieval. Machine Learning, **24**:95-122.

CASELLA, G., and R. BERGER. (1990). Statistical inference, Wadsworth & Brooks/Cole, Duxbury Press.

COSMADAKIS, S., KANELLAKIS, P., and N. SPYRATOS. 1986. Partition semantics for relations. Journal of Computer and System Sciences, **33**:203-233.

DEMETROVICS J., and G. GYEPESI. On the functional dependency and some generalizations of it. Acta Cybernetica, **8**(3):273-278.

DAVEY B., and H. PRIESTLEY. 1990. Introduction to lattices and order. Cambridge University Press, Cambridge, Great Britain.

GANTER, B. 1988. Composition and decomposition of data in formal concept analysis.*In* Classification and related methods of data analysis. *Edited by* H. Bock, North Holland, pp. 561-566.

GODIN, R., E. SAUNDERS, and J. GECSEI. 1986. Lattice model of browsable data spaces. Information Sciences, **40**:89-116.

GODIN, R., R. MISSAOUI, and H. ALAOUI. 1991. Learning algorithms using a Galois lattice structure. *In* Proc. of the 1991 IEEE International Conference on Tools for AI. San Jose, CA, IEEE Computer Society Press, pp. 22-29.

GODIN, R., and R. MISSAOUI. 1994a. An incremental concept formation approach for learning from databases, Theoretical Computer Science: Special Issue on Formal Methods in Databases and Software Engineering, **133**:387-419.

GODIN, R., R. MISSAOUI, and H. ALAOUI. 1995. Incremental concept formation algorithms based on Galois (concept) lattices. Computational Intelligence, **11**(2):246-267.

GUIGES, J. L., and V. DUQUENNE. 1986. Famille non redondante d'implications informatives résultant d'un tableau de données binaires. Mathématique and Sciences Humaines, **95**:5-18.

JOHN, G. and P. LANGLEY. 1996. Static versus dynamic sampling for data mining. *In* Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press, pp. 367-370.

KIVINEN, J., and H. MANNILA. 1995. Approximate inference of functional dependencies from relations. Theoretical Computer Science: Special Issue on Formal Methods in Databases and Software Engineering, **149**:129-149.

MAIER, D. 1983. The theory of relational databases. Computer Science Press, Rockville, MD.

MANNILA, H., and K. RÄHIÄ. 1986. Design by example: an application of Armstrong relations. Journal of Computer and System Sciences, **33**: 126-141.

MANNILA, H., and K. RÄHIÄ. 1992. On the complexity of inferring functional dependencies. Discrete Appl. Math. , **40:**237-243.

MANNILA, H., and K. RÄHIÄ. 1994. Algorithms for inferring functional dependencies from relations. Data & Knowledge Engineering, **12**(1):83-99.

MELLISH, C. 1991. The description identification problem. Artificial Intelligence, **52**(2):151-168.

MICHALSKI, R. 1983. A theory and methodology of inductive learning. Artificial Intelligence, **20**:111-161.

MISSAOUI, R., and R. GODIN. 1994b. Search for concepts and dependencies in databases. *In* Rough sets, fuzzy sets and knowledge discovery. *Edited by* W. Ziarko, Workshops in Computing series, Springer-Verlag, pp. 16-23.

MITCHELL T. 1982. Generalization as search. Artificial Intelligence, **18**:203-226.

MURPHY, P., and D. AHA. 1995. UCI repository of machine learning databases (Machine-readable data repository). Irvine, CA: University of California, Department of Information and Computer Science.

OOSTHUIZEN, D. 1994. The application of concept lattices to machine learning. Technical Report CSTR 94/01, University of Pretoria.

ORLOWSKA, M., and M. ORLOWSKI. 1992. Maintenance of knowledge in dynamic information systems. Handbook of applications and advances of the rough sets theory. *Edited by* R. Slowinski. Kluwer Academic Publishers, Dordrecht, Netherlands, pp. 315-329.

PAWLAK, Z. 1991. Rough sets: theoretical aspects of reasoning about data. Kluwer Academic Publishers, Dordrecht, Netherlands.

PIATETSKY-SHAPIRO, J. 1991. Discovery, analysis and presentation of strong rules. *In* Knowledge Discovery in Databases. *Edited by* G. Piatetsky-Shapiro and W. Frawley. AAAI Press, 1991, pp. 229-248.

QUINLAN, R. 1993. C4.5: programs for machine learning. Morgan Kaufmann, San Mateo, CA.

RUSSEL, S. 1986. Preliminary steps toward the automation of induction. *In* Proc. of the 5th national Conference on Artificial Intelligence, Philadelphia, PA, AAAI Press, pp. 477-484.

SAVNIK I., and P. FLACH. 1993. Bottom-up induction of functional dependencies from relations. *In* Proc. of the AAAI-93 Workshop on Knowledge Discovery in Databases, pp.174-185.

SCHLIMMER, J. 1993. Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. *In* Proc. of the.10th International Machine Learnig Conference, Amherst, MA, Morgan Kaufmann, pp.284-290.

SHAN, N., W. ZIARKO, H. HAMILTON, and N. CERCONE. 1995. Using rough sets as tools for knowledge discovery. *In* Proc. of the 1st International Conference on Knowledge Discovery and Data Mining, Montreal, Canada, AAAI Press, pp. 263-268.

SKOWRON, A., and C. RAUSZER. 1992. The discernibility matrices and functions in information systems. *In* Intelligent decision support. Handbook of applications and advances of the rough sets theory. *Edited by* R. Slowinski. Kluwer Academic Publishers, Dordrecht, Netherlands, pp. 331-362.

SKOWRON, A., and L.POLKOWSKI. 1997. Synthesis of decision systems from data tables. *In* Rough Sets and Data Mining. Analysis of Imprecise Data. *Edited by* T.Y. Lin and N. Cercone. Kluwer Academic Publishers, Dordrecht, Netherlands, pp. 259-299.

TOIVONEN, H., M. KLEMETTINEN, P. RONKAINEN, K. HAETOENEN, and H. MANNILA. 1995. Pruning and grouping of discovered association rules. *In* Working Notes of the MLnet Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases. Crete, Greece, pp. 47-52.

ULLMAN, J. 1988. Principles of database and knowledge-base systems, Vol. I, Computer Science Press.

WILLE, R. 1984. Line diagrams of hierarchical concept systems. International Classification, **2**:77-86.

WILLE, R. 1992. Concept lattice and conceptual knowledge systems. *Computer & Mathematics with Applications,* **23**(6-9):493-515.

WILLE, R. (1995). The basic theorem of triadic concept analysis. Order **12**:149-158.

ZIARKO, W. 1991. The discovery, analysis and representation of data dependencies in databases. *In* Knowledge discovery in databases. *Edited by* G. Piatetski-Shapiro and W. Frawley. AAAI Press, menlo Park, CA, pp. 195-209.

ZIARKO, W., and N. SHAN. 1996. A method for computing all maximally general rules in attribute-value systems Computational Intelligence, **12**(2):223-234.

TABLE 1.  Example relation for the planets of the solar system.

| | Planet size | | | Distance from sun | | Moon | | Period | |
|---|---|---|---|---|---|---|---|---|---|
| | **ss** | **sm** | **sl** | **dn** | **df** | **my** | **mn** | **ps** | **pl** |
| **Mercury** | x | | | x | | | x | x | |
| **Venus** | x | | | x | | | x | x | |
| **Earth** | x | | | x | | x | | x | |
| **Mars** | x | | | x | | x | | x | |
| **Jupiter** | | | x | | x | x | | x | |
| **Saturn** | | | x | | x | x | | x | |
| **Uranus** | | x | | | x | x | | | x |
| **Neptune** | | x | | | x | x | | | x |
| **Pluto** | x | | | | x | x | | | x |

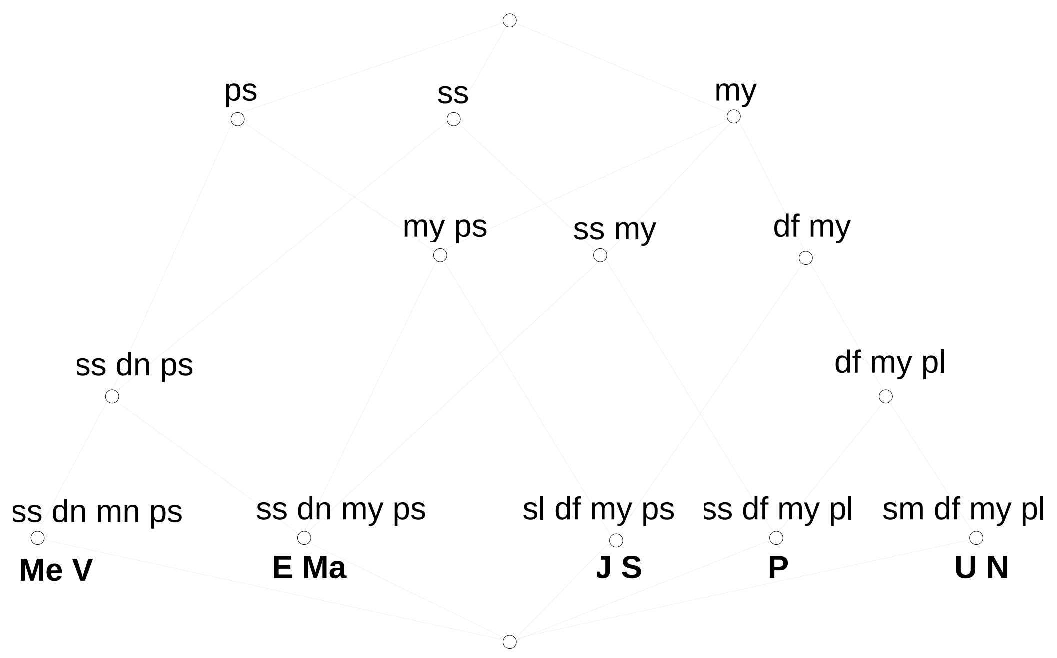

FIGURE 1. The concept lattice associated with the context of Table 1.

TABLE 2. The algorithm for inferring a rule cover from a lattice

```
Find-IRs
Input: a lattice L, a support threshold $\alpha$
Output: a set $\Sigma$ containing the cumulative set of IRs
begin
$\Sigma := \emptyset$;
for each node N = (X, Y) in L do
    if $\frac{\|X\|}{\|O\|} > \alpha$ then
    $\Sigma := \Sigma \cup$ Find-IRs-for -node-N (N= (X, Y))
    endif
endfor
return ($\Sigma$)
end


function Find-IRs-for-node-N (N= (X, Y))
/* Returns the set of IRs associated with the node N = (X, Y) */
begin
G-set := Y;        /* Contains the left-hand-sides of the IRs associated with N */
for each parent P = (W, Z) of N do
    G'-set := $\emptyset$;      /* Contains the elements of G-set updated by P* )
    for each G$\in$G-set do
        if G $\not\subseteq$ Z then
            G'-set := G'-set $\cup$ {G}
        else
            S-set := $\emptyset$;     /* Contains candidate specializations of the elements in G-set */
            for each attribute-value pair av $\in$ Y / G do
                S-set := S-set $\cup$ { G $\cup$ {av} }
            endfor;
            for each S $\in$ S-set do
                if Confirm-Specialization (S, G, Z, G-set) = true then
                    G'-set := G'-set $\cup$ {S}
                endif
            endfor
        endif
    endfor;
    G-set := G'-set
endfor;
$\Delta := \emptyset$;        /* Contains the set of IRs associated with N */
for each G $\in$ G-set do
    $\Delta := \Delta \cup$ {G $\rightarrow$ Y - G }
endfor
return ($\Delta$)
end


function Confirm-Specialization (S, G, Z, G-set)
/* Checks if a specialization is consistent with the parents of N and maximally general wrt other specs. */
begin
if S $\not\subseteq$ Z and there is no element S' $\in$ G-set - {G} such that S' $\subseteq$ S then
    return true
endif
end
```

TABLE 3. Size of the cover of generalised IRs as a function of the support threshold α.

| α / *dataset* | 0 | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.1 | 0.15 | 0.2 | 0.25 | 0.3 | 0.35 | 0.4 | 0.45 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Abalone | 29250 | 0 | | | | | | | | | | | | |
| Breast Cancer | 11710 | 5241 | 1885 | 933 | 544 | 356 | 65 | 15 | 6 | 0 | | | | |
| Breast Cancer W | 19724 | 2927 | 1924 | 1477 | 1275 | 1002 | 530 | 190 | 79 | 77 | 74 | 55 | 25 | 0 |
| Bridges 1 | 3065 | 1709 | 1223 | 948 | 774 | 624 | 239 | 75 | 36 | 10 | 6 | 2 | 1 | 0 |
| Bridges 2 | 5027 | 3603 | 2837 | 2331 | 1968 | 1634 | 566 | 161 | 61 | 15 | 6 | 2 | 1 | 0 |
| Glass | 1760 | 9 | 5 | 4 | 3 | 2 | 0 | | | | | | | |
| Liver Disorders | 3423 | 10 | 2 | 0 | | | | | | | | | | |
| Random-300 | 24959 | 5 | 0 | | | | | | | | | | | |
| Solar Flares | 13245 | 6856 | 4679 | 3832 | 3146 | 2686 | 1434 | 889 | 572 | 434 | 330 | 248 | 161 | 117 |
| Tic Tac Toe | 45504 | 2124 | 288 | 104 | 30 | 12 | 0 | | | | | | | |
| Wpbc | 17549 | 70 | 11 | 3 | 2 | 2 | 0 | | | | | | | |
| Wdbc | 6647 | 6 | 6 | 0 | | | | | | | | | | |

TABLE 4. Experimental evaluation of Carpineto, Romano and d'Adamo's IR-finding algorithm ($CRAAL_{IR}$) versus Godin and Missaoui's algorithm (GMAL).

| Dataset | | | | Lattice | | IRs ($\alpha$=0) | | GMAL | $CRAAL_{IR}$ |
|---|---|---|---|---|---|---|---|---|---|
| Name | # objects | # attributes | missing values | # nodes | CPU time (sec) | # IRs | size of nonredundant cover | CPU time (sec) | CPU time (sec) |
| Abalone | 4177 | 9 | N | 4262 | 748 | 29250 | NA | 67 | 2 |
| Breast Cancer | 286 | 10 | Y | 9936 | 153 | 11710 | 3449 | 73 | 13 |
| Breast CancerW | 699 | 11 | Y | 9860 | 261 | 19724 | NA | 78 | 6 |
| Bridges 1 | 108 | 13 | Y | 1667 | 19 | 3065 | 1681 | 188 | 2 |
| Bridges 2 | 108 | 13 | Y | 3242 | 39 | 5027 | 1635 | 233 | 4 |
| Glass | 214 | 11 | N | 258 | 4 | 1760 | 3608 | 27 | 1 |
| Liver Disorders | 345 | 7 | N | 1265 | 20 | 3423 | 4174 | 5 | 1 |
| Random-300 | 300 | 10 | N | 6248 | 92 | 24959 | NA | 166 | 23 |
| Solar Flares | 1066 | 13 | N | 10299 | 332 | 13245 | NA | 313 | 7 |
| Tic Tac Toe | 958 | 10 | N | 59503 | 1564 | 45504 | NA | 276 | 26 |
| Wdbc | 569 | 31 | N | 599 | 45 | 17549 | NA | 560 | 5 |
| Wpbc | 198 | 35 | Y | 384 | 11 | 6647 | NA | 507 | 4 |

TABLE 5. The transformed context associated with the context shown in Table 1.

| | S | D | M | P |
|---|---|---|---|---|
| (1, 2) | x | x | | x |
| (1, 3) | | | | x |
| (1, 5) | x | | | |
| (2, 3) | | | x | x |
| (2, 4) | | | x | |
| (2, 5) | x | | x | |
| (3, 4) | | x | x | |
| (3, 5) | | x | x | |
| (4, 5) | | x | x | x |

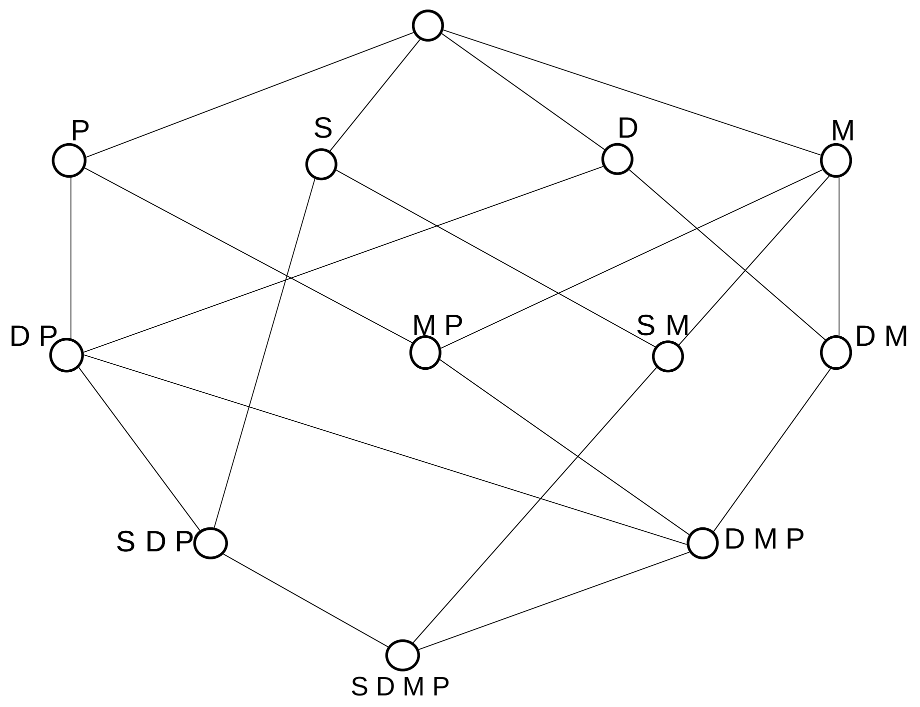

FIGURE 2.  The concept lattice associated with the transformed context shown in Table 5.

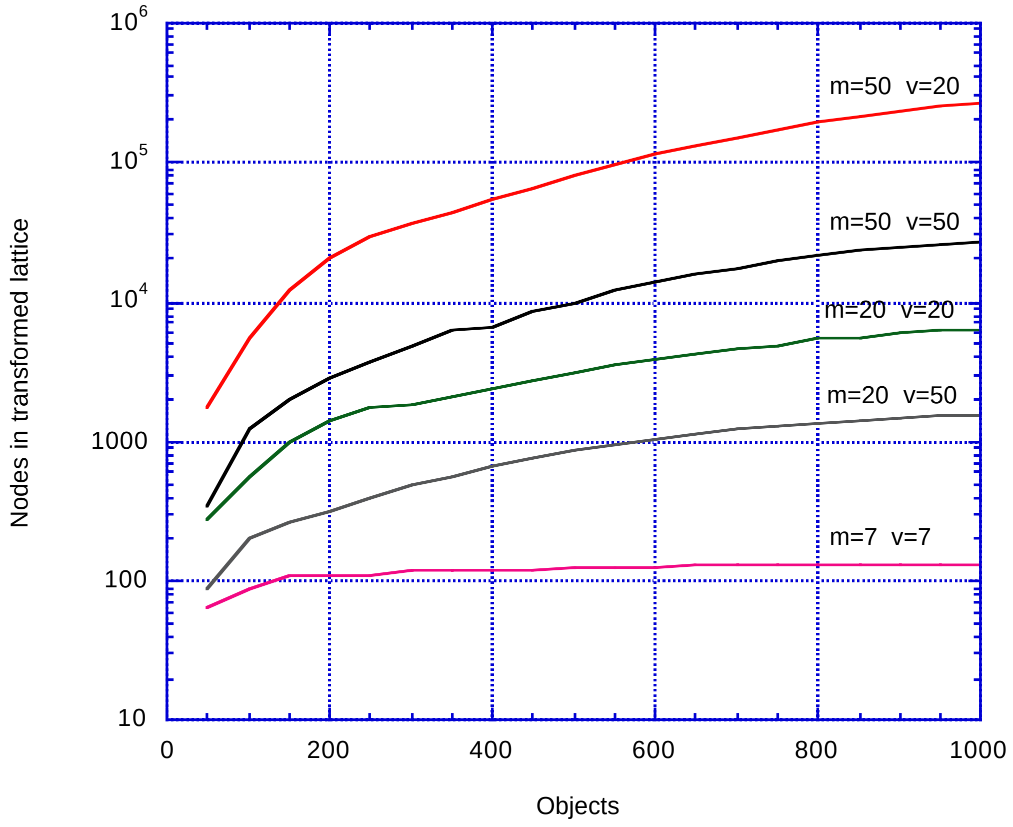

FIGURE 3. Size of transformed concept lattices as a function of data complexity, under the assumption of uniform distribution. Y scale is logarithmic.

TABLE 6. Experimental evaluation of Carpineto, Romano and d'Adamo's FD-finding algorithm ($CRAAL_{FD}$) versus Mannila and Räihä's algorithm (MRAL).

| Dataset | MRAL | | | | $CRAAL_{FD}$ | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Name | Max lhs | CPU time (secs) | # FDs | size of minimal cover | # nodes | Lattice CPU time (secs) | Total CPU time (secs) | # FDs | size of minimal cover |
| Abalone | 8 | 6610 | 55 | 9 | 4 | 364 | 364 | 8 | 9 |
| Breast Cancer | 25 | 3 | 0 | 0 | 1022 | 32 | 32 | 0 | 0 |
| Breast Cancer-W | 44 | 502 | 20 | 20 | 1113 | 79 | 86 | 364 | 41 |
| Bridges 1 | 32 | 22 | 116 | 61 | 1329 | 22 | 28 | 434 | 61 |
| Bridges 2 | 28 | 7 | 52 | 35 | 2621 | 77 | 85 | 340 | 35 |
| Glass | 10 | 27 | 61 | 11 | 27 | 2 | 2 | 10 | 15 |
| Liver Disorders | 13 | 49 | 38 | 14 | 45 | 4 | 4 | 12 | 13 |
| Nursery | 3 | 2694 | 1 | 1 | 509 | 8802 | 8802 | 1 | 1 |
| Random-300 | 96 | 4526 | 707 | 153 | 441 | 16 | 16 | 182 | 154 |
| Solar Flares | 29 | 23 | 0 | 0 | 4096 | 512 | 516 | 0 | 0 |
| Tic Tac Toe | 10 | 85 | 18 | 10 | 1003 | 140 | 140 | 9 | 10 |
| Wdbc | 59 | 24816 | 1673 | 48 | 11 | 22 | 22 | 67 | 50 |
| Wpbc | 44 | 2180 | 1496 | 52 | 23 | 3 | 3 | 50 | 78 |

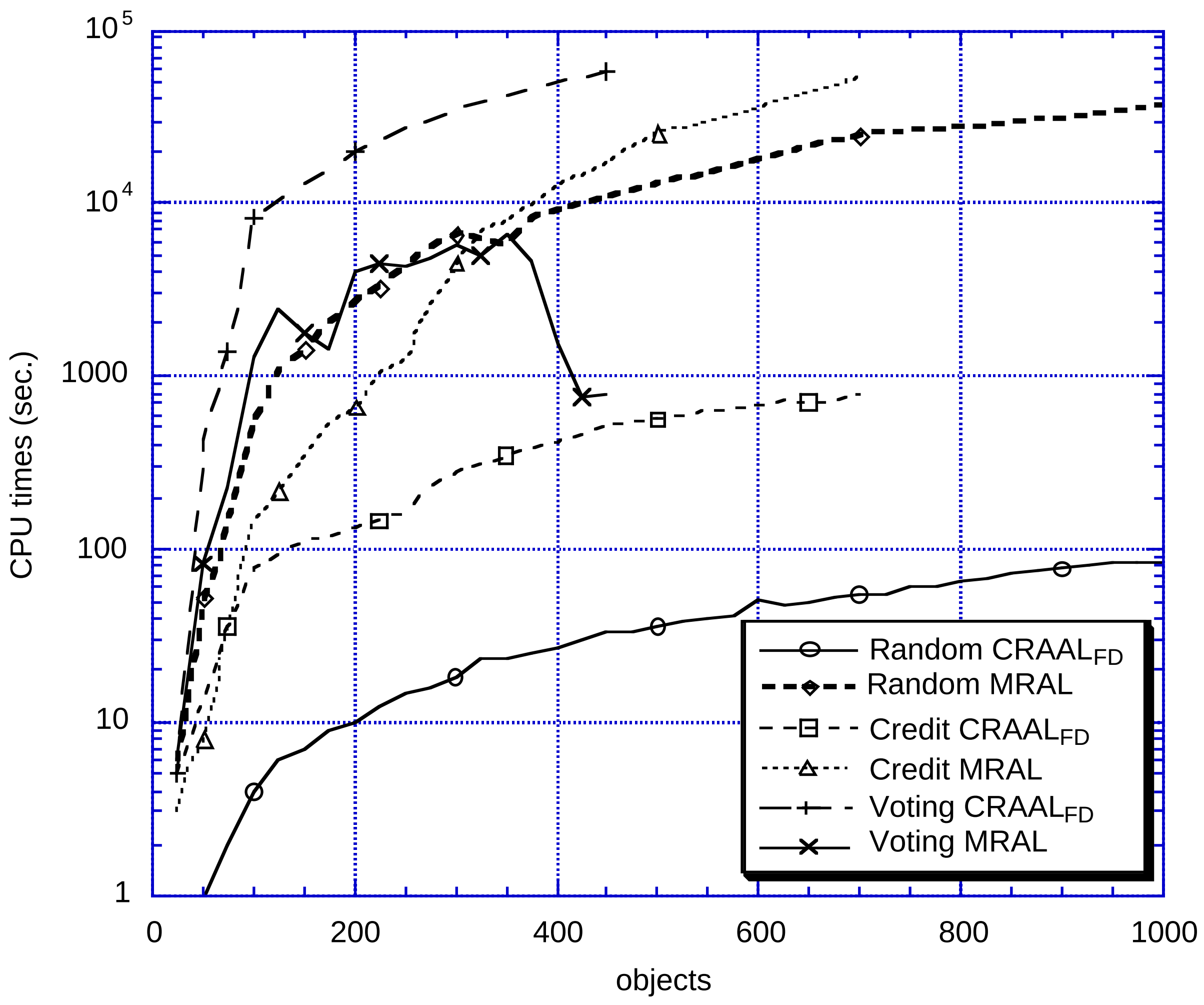

FIGURE 4. Experimental time complexity of CRAAL$_{FD}$ and MRAL as a function of the number of objects. Y scale is logarithmic.