Inferring minimal rule covers from relations

CLAUDIO CARPINETO and GIOVANNI ROMANO

Fondazione Ugo Bordoni,
Via B. Castiglione 59, 00142 Rome, Italy

Tel: +39-6-54803426
Fax: +39-6-54804405
E-mail: carpinet@fub.it

Abstract

An implication rule Q→R is roughly a statement of the form "for all objects in the database, if an object has Q then it has also R". We introduce a definition of minimal cover for the set of implication rules that hold in a relation, by analogy with earlier work on functional dependencies, and present an approach to computing it. The core of the proposed approach is an algorithm for inferring a reduced cover containing only maximally general rules, i.e., such that no attribute-value pair on any left side is redundant. We study the computational complexity of the proposed approach and present an experimental comparison with another method that confirms its validity, especially when the number of attributes in the database is limited.

Keywords: learning from databases, knowledge discovery, implication rules, cover, 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. Several types of dependencies have been investigated, ranging from classical functional dependencies (Schlimmer 1993; Mannila and Räihä 1994) to several types of attribute-oriented rules (Piatetsky-Shapiro 1991; Agrawal and Srikant 1994; Godin and Missaoui 1994, 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. The first 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). The second common objective 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 can also be applied to attribute-oriented dependencies.

In this paper we deal with the problem of finding a minimal cover for a class of data dependencies called implication rules, useful for discovering hidden patterns in data (Godin and Missaoui 1994, Ziarko and Shan 1996). An implication rule $Q \rightarrow R$ is roughly a statement of the form “for all objects in the database, if an object has Q then it has also R”. We provide a definition of minimal cover for the set of implication rules that hold in a relation, based on an analagous definition provided by Ullman (1988) for functional dependencies, and present a method for computing it. The main component of our method is an algorithm that finds a cover containing only maximally general rules; this reduced, although not minimal, cover is subsequently processed by a procedure that removes the remaining redundant rules from it. A

study of the computational complexity of the algorithm, along with an experimental comparison with an alternative clustering-based approach, suggests that the proposed approach performs favourably, especially in those situations where the number of attributes is limited.
The rest of the paper has the following structure. In the next section, we introduce implication rules and characterize their minimal covers. In section 3, we describe the algorithm for computing a reduced cover for the set of implication rules that hold in a relation; then we discuss how to remove redundancy from the reduced cover to obtain a minimal cover. Section 4 deals with the complexity of inferring implication rules and analyses the complexity of the algorithm for finding maximally general rules. Section 5 contains an experimental comparison between the approach described in this paper and that based on lattice conceptual clustering, which was first proposed by Godin and Missaoui (1994) and has been later improve Carpineto and Romano (submitted). Section 6 concludes the paper with a summary an directions for future work.

2. Minimal covers for implication rules

Following the attribute-value representation model widely used in machine learning, knowl discovery, and databases, we assume that the data are represented by a relation. More precise given a set of objects (O), a set of attributes (A), and a set of attribute values (V), a relation is a quadruple (O, A, V, I) where I is a ternary relation between O, A, and V (i.e., I __ O x A x 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. Implication rules are defined in the following way.

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 $\forall o \in O, \forall i \in [1,h], \forall j \in [1,k], r_i (o) = s_i \Rightarrow t_j (o) = u_j$.

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 Σ, it is often possible to infer other members of Σ. In fact, owing to the nature of IRs, the inference system developed in database theory for functional dependencies (Maier 1983) holds also for

IRs. This contrasts with other types of attribute oriented rules, such as association rules (Agrawal and Srikant 1994), that cannot support reasoning.[1] The following definitions are useful to find succint representations of the complete set of IRs that hold in a context.

Given a set Σ of IRs, the closure Σ^+ is the set of rules implied by Σ by application of Armstrong's inference axioms. Two sets Σ and Σ' are equivalent if they have the same closure. If Σ and Σ' are equivalent, then Σ' is a *cover* for Σ. By analogy with Ullman (1988), we propose the following definition for minimal cover.

Definition 2. A cover Σ' is minimal if:
a) Every right side of an implication rule in Σ' is a single attribute-value pair.
b) For no $Q \rightarrow R$ in Σ' is the set $\Sigma' - \{ Q \rightarrow R \}$ equivalent to Σ'.
c) For no $Q \rightarrow R$ in Σ' and proper subset S of Q is $\Sigma' - \{ Q \rightarrow R \} \cup \{ S \rightarrow R \}$ equivalent to Σ'.

Intuitively, condition (b) guarantees that no rule is redundant, while condition (c) guarantees that no attribute-value pair on any left side is redundant (i.e., left sides are maximally general). Condition (a) and (c) may help the user focus on relevant rules; condition (b) allows well-founded more-succint representations of the set of rules. In the next section we describe a procedure for computing a minimal set of implication rules according to definition 2.

3. Inferring a minimal cover for implication rules

The core of our method is an algorithm, described in section 3.1, for finding a reduced cover that satisfies property a) and c) of Definition 2, while property b) is handled by an additional procedure, described in section 3.2, that applies to the reduced cover.

3.1 Finding maximally general rules

For each attribute-value pair (a,v) present in the relation, we want to find the set LHS (left-hand side) containing all possible maximally general rules of the form $Q \rightarrow (a,v)$. Since all rules $Q \rightarrow R$ can be derived from rules with a single attribute-value pair on the right-hand side, the total set of rules generated this way is a cover for the relation.

A simple method to infer the set LHS is the following. Since each object *o* will typically support some rules (among those whose right side is equal to a subset of the attribute-value pairs present in *o*) as well as prevent some other rules (among those whose right side is *not*

[1] A thorough discussion of the differences between implication rules and association rules, as well as of the relation between implication rules and functional dependencies, is contained in Carpineto and Romano (submitted).

equal to any of the attribute-value p[illegible] present in *o*), we could maintain a set containing all possible *lhs* candidates and update it [illegible]xamining one object at a time. Candidates contradicted by the object shoul[illegible] set, candidates confirmed should be marked. After all obj[illegible] marked rules is a cover (although not maximally general). This approach, [illegible]ever, would require to examine p^m candidates for each object, where p is th[illegible] m is the number of attributes.

The algorithm we present is ba[illegible]n partitioning[illegible]f obj[illegible]ion in two classes, i.e, the set of “positive” o[illegible] such tha[illegible]” objects such that a(o) v. The main loo[illegible] the a[illegible]thm iterat[illegible] the positive [illegible]ts. The algorithm keeps and updates for each positive object [illegible]les that can b[illegible]oretically generated from it; i.e., all the possible 2^{m-1} subs[illegible]e [illegible]alue pai[illegible]ent in the object, excep[illegible]e [illegible]ithm [illegible] fi[illegible]rules that are contradi[illegible] some negative o[illegible]ual to any other rules p[illegible]ed by the other[illegible]e exa[illegible]nally[illegible]ollect only the most general rules [illegible]e remaining ones. These ar[illegible]tiona[illegible]lt problems.

Th[illegible]y idea of the algorithm[illegible]red s[illegible]resenting the candidate rule sp[illegible]ver which the th[illegible]nd in an integrated manner. C[illegible] an ordered[illegible] power set o[illegible] attribute-value pairs present in the o[illegible]dard set inclusi[illegible]elation (i.e., x ≥ y if x__y). This ordered set is [illegible] form[illegible]ith re[illegible] task[illegible]d. In particular, if an element is ruled out by so[illegible]egativ[illegible]ct the[illegible] its g[illegible] elements are also ruled out; conversely, if an element [illegible] valid[illegible] then all its smaller elements are also valid rules. Using { (o); ≥} allow[illegible]dle pr[illegible]g due to negati[illegible] manner. For each negative objec[illegible] algo[illegible] finds the i[illegible]ection be[illegible] the negative object and the current positive obj[illegible]all the elements that ar[illegible]an or equal to the intersection from { (o); ≥}. Deali[illegible]th the rule duplication[illegible]ue[illegible]er positiv[illegible] objects is also easy. If we lexicographically order the set of po[illegible] obje[illegible]we [illegible] the current candidate space all those rules that will be conside[illegible]when [illegible]ining subsequent positive objects. The “deferred” candidates are those element[illegible] are g[illegible]r than or equal to the intersection between the current positive object and any [illegible]eque[illegible]sitive object. This way we can avoid generating rules more than once. The [illegible]l ope[illegible]n involved in each iteration is the collection of the most general rules, perfo[illegible] a specific-to-general breadth-first search through the elements of { (o); ≥} that [illegible] neither been pruned nor deferred.

A complete description of the algorithm is given in Table 1.

TABLE 1. The algorithm for inferring a rule c[illegible]

```
Find-LHS
Input: a relation (O, A, V, I), an attr[illegible]a, v) such tha[illegible] a, v) ∈ I for some o∈O.
Output: the set LHS relative [illegible]

    LHS := ∅
    O+ := { o ∈ O | a(o) = v}        set[illegible]ive[illegible]
    O- := { o ∈ O | a(o)   v}        ; se[illegible]
    Loop for i ∈ [1, || O+||[illegible]
        Loop for o- ∈ O[illegible]
            int- := o_i^+   o-[illegible]
            Loop for c ∈ {  (o[illegible]a,v}) [illegible]ch[illegible] do
                label (c) = “de[illegible]
        Loop for j [illegible]
            int+ [illegible]o_j^-[illegible]
            Loop for c ∈ [illegible]≥[illegible]hat [illegible]+ and label (c) [illegible]’ do
                    labe[illegible]
        Loop  fo[illegible] (o_i^+[illegible]h that [illegible])   “del”  and  label [illegible]  “def”   ; collect[illegible]
                wh[illegible] all pa[illegible]s of [illegible](c)) = “del” do
            LHS :[illegible]S ∪ {c[illegible]
```

We have to em[illegible]view it is convenient to implemen[illegible](o); ≥} using a vec[illegible]d access fa[illegible] More precisely, we coded each[illegible]nent of (o) by a bit v[illegible]sition *i* is equal[illegible] or 0 depending on whether the[illegible]bute-value pair *i* is prese[illegible]he element. Th[illegible]it vectors are used as indices to [illegible] length 2^{m-1} cell v[illegible]r that [illegible]infor[illegible]bout the elements of (o). The addr[illegible]f the parents (acc[illegible]g to [illegible]an[illegible]puted from the address of the element itself through bit mas[illegible]. Thu[illegible]s implementation allows direct acces to each element of (o) and to its parents.

To illustrate t[illegible]orkin[illegible] the algorithm, we refer to a simple database consisting of four objects descr[illegible]by fo[illegible]ributes (see Table 2). Let d_1 be the current *rhs* and consider the step relative to the [illegible]ive object ($a_1b_1c_1d_1$). For the sake of clarity, we represent the set (($a_1b_1c_1d_1$); ≥[illegible] a graph rather than as a vector (see Figure 1). Examinaton of the first negative object ($a_1b_1c_2d_2$) causes pruning of a_1b_1 (intersection between $a_1b_1c_2d_2$ and the current positive object) and of its ancestors (a_1 and b_1) from the graph. Object $a_2b_2c_1d_2$ causes pruning of element c_1. Finally, the effect of the positive example $a_1b_2c_1d_1$ is deferment of element a_1c_1. In Figure 1, pruned elements are encircled, while deferred elements are boxed. Once all three objects have been examined the unlabeled elements in the graph are visited ($a_1b_1c_1$ and b_1c_1) and the most general of them are collected. The result is the rule $b_1c_1 \rightarrow d_1$.

The cover returned by the algorithm described in Table 1 satisfies condition (a) and (c) of Definition 2, but it will not, in general, satisfy condition (b) because whenever the cover contains the rules $Q \rightarrow R$ and $R \rightarrow S$ it will also contain the rule $Q \rightarrow S$.

Table 2: An example database.

a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_2
a_2	b_2	c_1	d_2
a_1	b_2	c_1	d_1

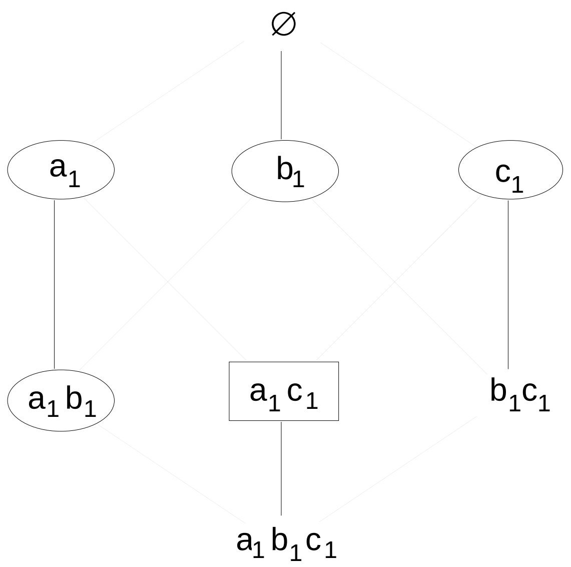

Figure 1: The ordered set of the candidate *lhs* rules associated with the first object of the database shown in Table 2. Encircled elements are pruned by negative objects, boxed elements are deferred by positive objects.

3.2 Removing cover redundancy

In order to generate a nonredundant cover for a set Σ of implication, we must delete from Σ each $Q \rightarrow R$ such that $Q \rightarrow R$ is “member” of $\Sigma - \{Q \rightarrow R\}$. To determine if the rule $Q \rightarrow R$ can be derived from the set $\Sigma - \{Q \rightarrow R\}$, it is sufficient to compute the closure of Q under $\Sigma - \{Q \rightarrow R\}$, denoted Q^+, and then check whether $R \subseteq Q^+$. Maier (1983) describes an efficient algorithm for computing Q^+ for functional dependencies that can be easily adapted to implication rules. The algorithm has $O(k)$ time complexity, where k is the cardinality of Σ. Therefore the time complexity of the procedure for removing redundant rules from Σ is $O(k^2)$, which may be exceedingly hard for many real problems. We will see in section 5 that when the input (redundant) cover is large the procedure for removing redundancies may easily run into computational barriers. In the next section we discuss the computational complexity of IR inference and analyse the complexity of the algorithm for finding maximally general rules.

4. Complexity of IR inference

We first consider how the growth of the two main problem parameters, i.e., the number of objects *n* and the number of attributes *m*, 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., $P \rightarrow Q$ may be replaced by $PR \rightarrow Q$ and $PS \rightarrow Q$); thus, the size of the cover may increase even while the set of IRs that hold in a relation 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 *functional dependency* 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 3). In many situations, however, this problem may not become very serious, because in real domains it is often the case that the number of attributes in a relation is relatively small. Furthermore, the growth of the number of IRs may be dramatically reduced by allowing for statistical sig[illegible] the holding rul[illegible]s mentioned in the last section.

We now analyse the complexit[illegible]he *F*[illegible]*LHS* [illegible]scribed in Table 1. For each positive object we must check all [illegible] obje[illegible]nd update the candidate-rule space. Since each update, including the final colle[illegible] requ[illegible]visiting distinct nodes and testing their parents, the time complexity involved is [illegible]ded b[illegible] maximum branching factor times the cardinality of the candidate-rule space. In f[illegible] we ex[illegible]entally checked that for each positive object the number of nodes visited is const[illegible]2^{m-1}, where m/2 is just the average branching factor of the elements in ((o); $\geq$). Thus[illegible] time complexity for each positive object cannot exceed $O(n+(m/4)2^m)$, and the time complexity of the Find-LHS algorithm is therefore < $O(n_p(n+m2^m))$, where n_p is the number of positive objects. In practice, $n<<m2^m$; furthermore, n_p is usually a small fraction of the number of objects. As a consequence, in many practical applications, the complexity is nearly $O(m2^m)$.

5. Related work

IRs, sometimes called implications (Guiges and Duquenne 1986) or simply rules (Ziarko and Shan 1996), have not been deeply investigated until recently.

Ziarko and Shan (1996) have presented an IR-finding method based on the computation of prime implicants of Boolean expressions. Similar to our approach, the cover output by Ziarko and Shan's contains only maximally general rules. However, the complexity of the method, as also noted by the authors, seems to be prohibitively high even for databases of limited size; in fact, no evidence is reported in the paper that in some real domains such an approach would be feasible.

The only other IR-finding method that we are aware of has been presented by Godin and Missaoui 1994, and later improved by Carpineto and Romano (submitted). This method is based on a particular representation of the input relation called concept lattice, from which the rules are then extracted. The found rules are not maximally general; the cover satisfies a weaker property than condition (b) of Definition 2, explained in Carpineto and Romano (submitted). For the scope of this paper, suffice it to say that the cover returned by the concept lattice method is a superset of the cover returned by Ziarko and Shan's method or by the algorithm described in this paper.

We now compare the Find-LHS algorithm with the concept lattice method, in particular with Carpineto and Romano (submitted)'s version. A theoretical comparison is difficult because the complexity of the concept lattice method cannot be directly expressed in terms of the main problem parameters; it would also require an in-depth analysis of the concept lattice method which is outside the scope of this paper. Our goal is rather to get some indications about the behaviour of the Find-LHS algorithm; therefore we compare the performance of the two algorithms on some real domains.[2]

For the experiment, we used five machine learning benchmarks available from the repository at University of California at Irvine, whose main features are described in Table 3. Numeric values were discretized into ten equal-length intervals.[3] 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. We ran the concept lattice method and the Find-LHS algorithm on each data set, computing the following variables: (a) CPU time necessary to find the cover, (b) size of the cover,.

The results are shown in Table 3. "CPU time" columns labeled "ET" indicate that the algorithm exceeded a time limit of 1000 seconds, while "NA" in the "cover size" column indicates that the procedure ran into computational barriers, and therefore the datum was not available.

Table 3. Experimental evaluation of the Find-LHS algorithm versus the concept lattice method

Dataset				**Concept Lattice Method**		**Find-LHS**	
Name	#objects	#attributes	missing values	CPU time	Number of rules	CPU time	Number of rules
Bridges 2	108	12	Y	23	28399	74	7151
Breast Cancer	286	10	Y	76	38678	48	14739
Liver Disorders	345	7	N	29	7775	11	5326
Breast Cancer W.	699	11	Y	174	110391	259	55394
Tic Tac Toe	958	10	N	ET	NA	290	27491

The results of the experiments show that the cover returned by the Find-LHS algorithm are significantly smaller than those produced by the concept lattice method. As for time performance, Find-LHS beat the concept lattice method on three of the five benchmarks, and lost on the remaining two. The results show that as the number of objects increase, the number of attributes staying stable, the performance of Find-LHS tends to improve over that of the concept lattice method. This may suggest that the number of objects may be a less critical factor for the Find-LHS algorithm's complexity than for the concept lattice method's complexity. This

[2] All the programs tested in this paper are written in Common Lisp. For the experiments we used a SUN Ultra 2 equipped with 248 Mbytes of RAM.

indication, however, is partially contradicted by the behaviour of the Breast-cancer-Winsconsin data set. In fact, an increase in the number of objects may sometimes be more harmful for Find-LHS than for the concept lattice method. This happens when the objects in the relation have very few similarities (i.e., their attributes values are different). In this case, while the intermediate representation used by the concept lattice method is small and easy to compute, the Find-LHS algorithm must be invoked many times because there are plenty of values taken on by each attribute.

We have also to emphasize that all data sets used in the experiment have a limited number of attributes. While the number of attributes represents an inherent limitation for the performance of the Find-LHS algorithm, the concept lattice method may work with data sets described by tens of attributes (Carpineto and Romano; submitted). On the other hand, the comparison between the CPU time of the two algorithms can only be taken as indicative because the returned covers have different characteristics. In particular, the time necessary to remove the rules that are not maximally general from the cover returned by the concept lattice method may significantly worsen its performance.

In order to obtain a better estimate of the relative performance of the two algorithms, in Table 4 we show the CPU times normalised with respect to the time necessary to find a minimal cover. The results show a marked superiority of the Find-LHS algorithm and give some insights into the cover reduction due to redundancy removal.

Table 4. Experimental comparison of time performance normalised with respect to redundancy removal

Dataset	**Concept Lattice Method**			**Find-LHS**		
Name	Size of redundant cover	CPU time for minimal cover	Size of minimal cover	Size of redundant cover	CPU time for minimal cover	Size of minimal cover
Bridges 2	28399	275	1240	7151	146	1236
Breast Cancer	38678	ET	NA	14739	489	3462
Liver Disorders	7775	76	2011	5326	57	2062
Breast Cancer W.	110391	ET	NA	55394	ET	NA
Tic Tac Toe	NA	ET	NA	27491	ET	NA

[3] In another series of experiments numeric attributes were treated as if they were nominal. The results were different but consistent with those reported below.

5. Conclusion

In this paper we considered the problem of finding a minimal cover for the set of implication rules that hold in a relation. We presented an algorithm for finding a cover containing only maximally general rules, from which a minimal cover can then be extracted. We analysed the complexity of the proposed approach and compared it with another IR-finding algorithm. Our approach performed favourably, especially for data sets with a limited number of attributes.

Work on analysis, computation and application of implication rules is still in its infancy. Among several issues that need be investigated, extraction of significant subsets of the set of rules inferred is important for practical applications. Following Definition 1, an IR holds even if there is only one object that satisfies the rule; consequently, we may discover many irrelevant or spurious rules. One way to cope with this problem is to restrict the set of valid rules to those rules that are supported by a given percentage of the objects. The introduction of such a parameter does not adversely affect reasoning in that inference axioms hold also for statistically-supported IRs; furthermore, it dramatically reduces the number of valid rules in a data set (Carpineto and Romano; submitted).

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 meaningless rules due to the presence of erroneus attribute values in some objects, although the use of the support threshold mentioned above 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, with the goal of handling noise while keeping reasoning capabilities. This is an avenue for future research.

Acknowledgements

Work carried out in 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.

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., G. ROMANO, and P. d'ADAMO (submitted). Inferring dependencies from relations: a conceptual clustering approach. Submitted for publication.

GODIN, R., and R. MISSAOUI. 1994. 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.

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

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

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

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

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.

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 First International Conference on Knowledge Discovery and Data Mining, Montreal, Canada, AAAI Press, pp. 263-268.

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.

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

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