Intelligent Data Analysis 5 (2001) 445–461
IOS Press

Inductive learning from numerical and symbolic data: An integrated framework

Floriana Esposito[a], Donato Malerba[a] and Vittorio Marengo[b]
[a]*Department of Informatics, University of Bari, Italy*
E-mail: {esposito, malerba}@di.uniba.it
[b]*Department of Statistics, University of Bari, Italy*
E-mail: vmarengo@dss.uniba.it

Received 21 March 2001
Revised 20 May 2001
Accepted 26 June 2001

Abstract. Numerical induction from data is one of the statistical data analysis tasks, which uses a tabular model, with almost exclusively numerical features, as data representation formalism. The output representations are different: from functions to probability distributions, from surface equations to tables of indexes. One approach to extend the classical data analysis techniques to symbolic objects is the Symbolic Data Analysis: the input and the output of classical techniques are expressed in a symbolic way, so guaranteeing the comprehensibility of both the observations and the results, while the processing techniques, although appropriately adapted, maintain the efficiency of the classical statistical inferential models. Also in the field of Machine Learning several methods have been proposed to extend some inductive approaches from statistical data analysis to data represented as attribute-value couples. Sometimes these approaches transform ideas and principles coming from numerical induction to handle propositional calculus descriptions, otherwise they combine different techniques in order to treat numerical-continuous data and algebraic-symbolic data differently. The aim here is to improve the efficiency and to preserve the expressive power of the representations during the learning process, and to save the accuracy and flexibility of the numerical techniques during the recognition phase. This kind of integration is more and more complex when using first-order computational learning models, which are useful for handling object descriptions in structured domains, when not only the properties of objects but also the relations between different objects must be considered. The necessity arises from integrating different computational strategies, different knowledge representations and different processing methods in a naive combination of classifiers or, more meaningfully, in a real integration within a unique theoretical framework. When building machine learning systems increasing attention is given to handling symbolic and numerical information inside the same system. In the paper, we face the problem of handling both numerical and symbolic data in first-order models, distinguishing the phase of model generation from examples, and the phase of model recognition by means of a flexible probabilistic subsumption test.

1. Introduction

Extracting knowledge from data with the aims of understanding and explaining phenomena and of decision making is the common goal of statistical Data Analysis and of Machine Learning. Both the research communities are aware that:

– analyzing complex aggregated data requires the handling of new set-valued variables in order to extract significant regularities and association rules;
– "understanding" the data requires an interpretation that is generally easier when the results of the analysis are expressed in a conceptual-symbolic way;

– processing a very large collection of data requires very efficient methods such as numerical ones.

Numerical and symbolic approaches to empirical learning share the common objectives of producing predictive knowledge from observations, but do not share the same semantics, thus hindering the possibility of using the different approaches in an integrated way. When the knowledge representation is in a numerical form the links among the objects must be defined in terms of operators having the semantics of numbers (*less-equal* or *twice* x); when the representation is symbolic, the constraints on and the relations among the objects must be expressed as predicates, related to the application domain (color $=$ [white, yellow]), and the semantics is directly expressed as theorems. The well-known consequence of using meaningful but heavy symbolic representations is the difficulty in handling numerical relations, loosing efficiency and precision in computation. On the other hand, numerical representation prevents the creation of chains of logical reasoning which permit explanations.

These considerations suggest integrating the approaches in a single theoretical framework, focusing the attention on learning classification theories, expressed in logic languages, and at the same time handling numerical information efficiently.

Extending classical data analysis with new tools of formalization in order to deal with symbolic data has been the goal of Symbolic Data Analysis since the end of the 1980s [10].

The theory demands an extension of the classical rectangular arrays, whose rows are the feature vectors characterizing an object or an observation: each element may contain an interval or several values instead of a single value as usual, and objects are not necessarily defined by the same variables. As a matter of fact, several techniques which have been developed for exploratory data analysis and multidimensional classification (factor analysis, clustering, multiple regression, canonical correlation etc.) are able to handle almost exclusively numerical features. Studies on categorical data analysis [1] are an exception and mainly concern the analysis of non numerical-categorical dependent/independent variables, paying little attention to the problem of how to search for good models. The main goal is that of testing whether a model adequately fits the sample data, while strategies for model generation and selection are only superficially examined. The problem of defining a generality order of models that organizes the space of inductive hypotheses has received little attention in classical data analysis. The first attempt to define the algebraic structure of the model space to construct efficient search strategies is made in Symbolic Data Analysis, where partially ordered assertion objects are dealt with [1].

Also in the field of Machine Learning several methods have been proposed to extend some inductive approaches from statistical data analysis to data represented as attribute-value couples. Sometimes these approaches transform ideas and principles coming from numerical induction to handle propositional calculus descriptions. Sometimes they combine different techniques in order to deal with numerical-continuous data and algebraic-symbolic data differently, with the aim of improving the efficiency and of maintaining the expressive power of representations during the learning process, and saving the accuracy and flexibility of the numerical techniques during the recognition phase. This kind of integration is more and more complex when using first-order computational learning models, which are useful for handling object descriptions in structured domains, when not only the properties of objects, but also relations between different objects must be considered. The need arises to integrate different computational strategies, different knowledge representations and different processing methods in a naive combination of classifiers or, more meaningfully, in a real integration within a single theoretical framework. According to this multistrategy learning view initially attention was focused on the integration of different reasoning methods; later different paradigms were integrated, including symbolic, connectionist and genetic ones. Recently increasing attention has been given to handling symbolic and numerical information inside the same system in First Order Logic (FOL).

In this paper, the previous work done by the authors is extended into two dimensions: First, the way to handle both numerical and symbolic information in a multistrategy view during the learning process and second, the definition of a flexible matching function for first-order definite clauses which are useful during the recognition process. As to the inductive inference from numerical and symbolic data, an information-theoretic specialization operator has been defined: it can specialize a clause by adding literals of the type

$$f(x_1, \ldots, x_n) \in [a \ldots b],$$

where $f(x_1, \ldots, x_n)$ is a function taking values in a numeric domain, while $[a \ldots b]$ denotes a closed interval. It is worthwhile observing that the arity of the function may be greater than one, that is, the operator is applicable to numerical relations like *distance* (x, y). This operator has been embedded in two proprietary first-order learning systems, INDUBI/CSL [19] and ATRE [20], which automatically generate knowledge in the form of Horn clauses. The problem of using this kind of knowledge for classification or prediction tasks is mainly in matching new observations, which may be noisy or incomplete, against rules. It is necessary to extend the canonical matching function in order to measure the similarities of two descriptions rather then their equality or inclusion, by an approximate subsumption algorithm. Both systems have been tested on the problem of classifying and understanding images of single-page documents [16]. The results have demonstrated the importance of handling both numerical and symbolic descriptions, since the former increase the sensitivity, while the latter increase the stability of the internal representation of visual objects.

2. Symbolic data analysis

Most statistical methodologies for data analysis were designed for handling almost simple representations: the unit for statistical analysis is an individual (i.e. a person or an object) described by a well-defined set of random variables (either qualitative or quantitative), each of which results in just one single value. Nowadays, data analysts are required to process data that go beyond the classical framework, as in the case of data concerning more or less homogeneous classes or groups of individuals instead of single individuals. A typical situation is that of census data, that is, a collection of information on the size and characteristics of a population, as well as on the number and characteristics of dwelling units, business enterprises and government agencies. The statistical unit of census is the household: each household receives a census questionnaire and is asked to answer the questions. Census data are needed to make decisions on matters of national and local policy concerning education, employment, transportation and so on. In order to guarantee privacy, aggregated data are made available: typical aggregations are formed by census tracts (almost 50.000 in US) or by city blocks (almost 7 millions in US). Another example of a source of aggregated data is offered by data warehouses, that are databases built to provide information and knowledge needed by decision-makers. Unlike operational decisions, which are based on individual transactions, management or strategic decisions are made by analyzing snapshots of performance of the whole corporation. Also in these on-line analytical processing (OLAP) applications, new data analysis methodologies dealing with aggregated data are required.

Aggregated data describe a group of individuals by set-valued or modal variables [5]. A variable Y defined for all elements k of a set E is termed set-valued with the domain Ω, if it takes values $Y(k)$ in $P(\Omega) = \{U | U \subseteq \Omega\}$, that is, the powerset of Ω. A single-valued variable is a special case of a set-valued variable for which $|Y(k)| = 1$ for each k. When $|Y(k)| = \infty$ for each k, then Y is called multi-valued.

An example of a multi-valued variable available in the census data is race, which aggregates information of ethnic origin of household members in a given city block. When an order relation $<$ is defined on Ω, then the value returned by a set-valued variable can be expressed by an interval $[\alpha, \beta]$, with some $\alpha, \beta \in \Omega$, and $\alpha < \beta$. In this case, Y is termed an *interval* variable. An example of an interval variable in OLAP application is age, which aggregates information on the age of customers that have bought beer from a given supermarket each week-end. In the last example it is clear that when only two customers buy beer, the former aged eighteen and the latter aged eighty, the data aggregation returns an interval [18,80] for a given week-end k. There is a clear loss of information that can be partially recovered by associating a distribution function with the interval. More generally, a *modal* variable is a set-valued variable with a measure or a (frequency-, probability- or weight-) distribution π_k associated to $Y(k)$.

A class of individuals described by a number of set-valued or modal variables is termed *symbolic data*. Symbolic data lead to more complex data tables called symbolic data tables, where each cell does not necessarily contain, as usual, just a single numeric or categorical value, but several values which can be weighted and also linked by taxonomies. It is important to observe that symbolic data aggregate data on individuals: henceforth, the term aggregation will be given a different meaning from the term summarization, which is well represented by two measures typically used in classical statistics, namely mean and standard deviation. Summarization leads to classical data tables, while aggregation returns symbolic data tables. The extension of classical data analysis techniques to such tables is termed Symbolic Data Analysis and has been the goal of both the European Esprit project SODAS 20821 and the IST-1999-25161 project ASSO, that have produced a formal theory [5] and a software package for Symbolic Data Analysis. However, these methods have not yet been extended to handle structured descriptions, that is, to symbolic objects representing not only the properties of objects but also relations among them.

3. Machine Learning for numeric and symbolic induction

The steady growth of Machine Learning within the area of Artificial Intelligence has introduced alternative representations and methods for analyzing the data. One major dimension in which to differentiate machine learning techniques is the complexity of the representation language they adopt. Most machine learning studies involve propositional attribute-value languages, which are comparable to tabular representations adopted in statistics, but which differ in the possibility of mixing both numeric and symbolic data.[1]

A typical example is represented by decision tree induction methods that work on continuous, nominal, ordinal and tree-structured attributes [2]. The main weakness of such methods derives from a lack of expressive power when relations among sub-parts have to be described. Indeed, even in the case of a fixed number of sub-parts and binary relations, the idea of defining a distinct attribute for each possible relation between two sub-parts would be realizable only if sub-parts could be totally ordered according to some criterion. Otherwise, the association of an attribute to pairs of sub-parts would not be deterministic and hence inexpressible in tabular form.

To overcome these limitations of propositional attribute-value representations, it is necessary to move to first-order logic (FOL). However, the change of language is not altogether easy, since the intractability

[1] Henceforth, the term numeric will denote integer and ratio-level measurements, while symbolic data will correspond to nominal and ordinal level measurements.

of predicate calculus and the NP-completeness of several decision problems raise new problems that can only be solved by imposing appropriate restrictions. In other words, a trade-off between expressiveness and computational tractability is necessary. These aspects have been widely investigated in machine learning, particularly by researchers working in the field of inductive logic programming (ILP) [20]. A great deal of research in the field of ILP concerns generalization models and the corresponding generalization/specialization operators. However, only recently studies have been performed on the problem of inductive inference from both numerical and symbolic data, although no studies have faced the problem of dealing with rules that near-miss the correct entailment of positive examples.

A first attempt to deal with continuous-valued attributes in FOL was made by Bergadano and Bisio [4], who proposed a method for automatically setting some parameters of predicate "schemes" with parametric semantics. A similar idea was proposed in ML-Smart [6] where a two-step approach is implemented. First a tentative numerical parameter is learned and then a standard genetic algorithm is applied to refine the numerical knowledge. Esposito et al. [14] proposed a different approach to the problem of handling both numerical and symbolic data. It was based on the integration of statistical data analysis with symbolic concept learning methods. More precisely, the authors combined a discriminant analysis technique for linear classification with a first-order learning method, so that the numerical information was handled by linear classifiers, while the symbolic attributes and relations were used by the first-order learning system.

One limit of all these approaches is that they have been defined for clauses with nullary predicates in the head, that is, with predicates corresponding to propositional classes. This means that such rules can be used to predict the membership class of an observation as a whole, but they cannot entail properties of sub-parts of an observation. For instance, the following clause:

$$\text{House} \leftarrow \text{red}(x),\ \text{ontop}(x, y),\ \text{triangle}(x),\ \text{block}(y).$$

can be used to classify houses with a red triangular sub-part on top of a square, but cannot be applied to the problem of understanding which part of the house is the roof. In the latter case, non-nullary predicates are necessary in the head of a clause, as in the following definite clause:

$$\text{roof}(x) \leftarrow \text{red}(x),\ \text{ontop}(x, y),\ \text{triangle}(x),\ \text{square}(y).$$

The two-phased approach implemented in ML-Smart has been upgraded to full, first-order definite clauses in the systems FONN [7] and NTR [8], where either neural networks or a gradient descent were used to refine numerical constants occurring in FOL classification theories.

In the framework of Inductive Logic Programming (ILP), a number of methods have been devised to solve the problem: transformation of relational problems into equivalent propositional ones as in LINUS [18], use of a priori knowledge either in a procedural form as in FOIL [25] or in a declarative form as in Progol [22].

The first class of methods includes the work by Dzeroski et al. [12] who propose the transformation of first-order representations into propositional form, in order to handle real numbers by means of techniques already tested in decision tree induction systems. This method can be applied only when all variables appearing in the body of a clause also appear in the head. Different methods of propositionalization have been implemented in REPART [30] and in ICP [28]. In the former case the propositionalization pattern is provided by the user, while in the latter case it is built from the training set. More recently, a lazy propositionalization method has been proposed for the system PROPAL, which selectively propositionalizes the FOL training set by interleaving attribute value reformulation and algebraic resolution [3].

The main representative of the second class of methods is FOIL 6.0 [26], which automatically produces comparative literals of type $V_i > k$, $V_i \geqslant k$, $V_i > V_j$, $V_i \leqslant V_j$, where V_j are numerical variables already present in other non-comparative literals and k is a numerical threshold. The semantics of the built-in relational predicates, as well as the heuristics for the selection of the best threshold k, are defined procedurally and embedded in the code of the system.

On the contrary, Progol [22] is the representative of the third class of methods, where the capability of numerical reasoning is explicitly "coded" in a declarative form as background knowledge. More recently, two implementation extensions of Progol have been proposed in order to make possible the use of any statistical and numerical analysis procedures [29]. This approach is quite general and allows an ILP program to exploit more fully the power afforded by the theoretical framework of ILP.

Our approach to the problem of learning "numerical" knowledge concerns the on-line discretization of numerical attributes and relations. It is characterized by the following features:

1. On-line discretization of numerical attributes and relations should be performed by a specialization operator, since the learning algorithms in which the operator is used, that is INDUBI/CSL and ATRE, perform a general-to-specific (or top-down) search.
2. The operator should always guarantee that the seed example that guides the induction process in both systems will be covered.
3. The heuristic function used to choose among different discretizations should satisfy some property that reduces the computational complexity of the operator.

This approach falls in the second class presented above, since it is quite a basic mechanism that allows ILP systems to work efficiently on numerical data. This is not in contrast with the definition in the background knowledge of domain-specific procedures for statistical and numerical analysis, as made by Progol.

Once a classification logic theory has been learned some rules may "near-miss" the correct entailment of observations. The problem becomes particularly relevant when numerical descriptions are involved. Indeed, a comparative literal of the type $V_i > k$ is falsified for any value slightly smaller than or equal to k. Therefore, a classical subsumption test should be replaced by an approximate subsumption test. Little attention has been paid to this research issue in inductive learning of FOL theories. Esposito et al. [13] proposed a solution to the problems raised by noise during the classification phase, based on a probabilistic interpretation of the matching predicate. The result of the matching process is no longer a true/false answer, but a probability that two well-formed formulae of a variable-valued logic matched, because of a change in one of them. However, one limitation of the proposed flexible matching process is that it has been defined for rules with nullary predicates in the head. An extension of this work will be presented in Section 5.

4. Model generation: Induction of first-order models

Models considered in this paper are defined by means of a first-order logic language, whose basic component is the literal or selector, which has two distinct forms:

$$f(t_1, \ldots, t_n) = \text{Value } (simple\ literal) \quad \text{and} \quad f(t_1, \ldots, t_n) \in \text{Range } (set\ literal),$$

where f is an n-ary function symbol, called *descriptor*, t_i's can be either variable or constant terms, and *Range* is a set of possible values taken by f. Some examples of literals are the following: color(x_1)

$=$ red, height$(x_2) \in [1.1 \ldots 2.1]$, and ontop$(x_1, x_2) =$ true. Literals can be combined to form *definite clauses*, which can be written as:

$$L_0 \leftarrow L_1, L_2, \ldots, L_m,$$

where the simple literal L_0 is called *head* of the clause, while the conjunction of simple or set literals $L_1, L_2, \ldots, L_m$ is named *body*. A clause with literal $f(t_1, \ldots, t_n) =$ Value in the head defines conditions that should be satisfied by the arguments $t_1, \ldots, t_n$, so that the function f can take the value Value when applied to $t_1, \ldots, t_n$. For instance, the clause

$$\text{identifier}(x) = \text{roof} \leftarrow \text{color}(x) = \text{red},\ \text{ontop}(x, y) = \text{true},$$
$$\text{shape}(x) = \text{triangle},\ \text{shape}(y) = \text{square}$$

defines the conditions that should be satisfied by x, so that *identifier* can take the value *roof* when applied to x. A *function definition* is a set of clauses that define f for all possible values in its domain. A *value definition* is a set of clauses that define the conditions that should hold for the arguments of f, so that f can take a certain value in its domain. Models considered in this paper are value definitions. Henceforth, we will concentrate our attention on value definitions of functions taking values in finite unordered domains.

Some particular definite clauses are obtained by imposing different constraints:

- *linkedness:* conditions in the body should be directly or indirectly related to the arguments of the function f defined by the clause;
- *range-restrictedness:* all arguments of the function f defined by the clause must be constrained by at least a condition in the body of the clause.

The clause reported above is both linked and range-restricted, while the following clause

$$f(x, y) = a \leftarrow g(y) = b,\ h(z) = c$$

is neither linked (condition $h(z) = c$ is not related to either x or y) nor range-restricted (we have no condition for x).

The first-order models generated by the learning systems INDUBI/CSL [19] and ATRE [20] are expressed as sets of linked, range-restricted definite clauses. These models are induced from a set of training (positive and negative) examples for a given function value, each of which is described as a single ground, linked and range-restricted definite clause. The generation proceeds according to a separate-and-conquer search strategy.[2] The separate stage is a loop that checks for the completeness of the current model, that is checks that all examples are explained (or *covered*) by the generated model. If this check fails, another clause to be added to the partial model is searched for. The new clause has to cover some of those examples still unexplained by the partial model. For instance, if we have the following positive examples for the value a for the function f:

$$e_1 :\ f(x, y) = a \leftarrow g(x) = b,\ h(y) = c,\ r(x, z) = d,\ g(z) = b$$
$$e_2 :\ f(u, v) = a \leftarrow g(u) = b,\ g(v) = b$$

[2] Actually, ATRE implements a parallel separate-and-conquer search strategy to learn multiple dependent value definitions.

and first-order model[3]

$$H: \ f(X,Y) = a \leftarrow g(X) = b, \ h(Y) = c$$

then it is easy to see that H explains e_1 and not e_2, thus we have to complete it by adding a further clause that covers at least e_2. In the separate stage both learning systems search in the space of all value definitions for a complete model.

The separate-and-conquer search strategy is adopted in other well-known learning systems, such as FOIL [25]. The main difference with our systems is the use of a *seed* example, whose function is that of guiding the learning process. Since each positive example should be covered by at least one clause of the model returned by the procedure separate-and-conquer, each example becomes a valuable source of information on the structure of the covering clause. Indeed, if e^+ is a positive example to be explained by an induced model H, then H should contain at least one clause C that covers e^+. Therefore, the systems start with a seed example e^+ and generate a set of at least M distinct range-restricted clauses, which are consistent and cover e^+. Then, the best generalization is selected from such a set according to a preference criterion. Finally, positive examples covered by the best generalization are removed from the set of positive examples and a new clause is generated, if the set of remaining positive examples is not empty.

In the conquer stage, both learning systems perform a beam-search in the space of definite clauses, looking for a linked, range-restricted definite clause that explains some positive examples but no negative example (*consistency* property). The search starts with the most general clause:

$$f(t_1, \ldots, t_n) = \text{Value} \leftarrow$$

and proceeds from general-to-specific, or top-down, by adding literals to the body until the obtained clause becomes consistent. In order to specialize a clause G, the systems have to choose some literals to be added. Both numerical and symbolic data are handled in exactly the same way, that is, they have to comply with the property of linkedness and should be sorted according to the very same preference criterion. The only difference is that numeric literals already present in the generalization can be reconsidered, in order to specialize the interval. It is worthwhile to observe that all selected literals are generalizations of literals in the seed example e^+, obtained by applying a simple inverse substitution [27] that replaces all occurrences of a term t_i by the same variable X_i. Thus, the seed example provides the learner with useful information on the structure of the generalizations. The computation of the interval for numerical set literals is described in Fig. 1.

A table associated to the function $f(X_1, X_2, \ldots, X_n)$ is built by matching the specialized clause

$$G, f(X_1, X_2, \ldots, X_n) \in [-\infty \ldots + \infty]$$

against positive and negative examples. Then an information-theoretic heuristic is used to select the best interval of values. The table, initially empty, contains pairs <Value, Class>, where *Class* can be either $+$ or $-$ according to the sign of the example e from which *Value* is taken. The *Value* is determined by considering the literal of the example e that matched against $f(X_1, X_2, \ldots, X_n) \in [-\infty \ldots + \infty]$.

Now the problem is that of finding the interval that best discriminates positive from negative examples. Any threshold value α, lying between two consecutive distinct values, will have the effect of producing

[3]Henceforth, we will follow the usual Prolog convention of starting variables with a capital letter, all other atoms being constants.

```
procedure determine_range([]I(X1, X2, ..., Xn)=Seed_value],e+,G,E+,E-)
  initialize table T[I(X1, X2, ..., Xn)]
  foreach example e in E+ do
    foreach substitution θ such that G,I(X1, X2, ..., Xn)∈ [-∞ .. +∞] covers e do
      select the literal [I(X1, X2, ..., Xn)=Value]θ of e
      add the tuple ⟨Value ,+⟩ to the table T[I(X1, X2, ..., Xn)]
    endforeach
  endforeach
  foreach example e in E- do
    foreach substitution θ such that G,I(X1, X2, ..., Xn)∈ [-∞ .. +∞] covers e do
      select the literal [I(X1, X2, ..., Xn)=Value]θ of e
      add the tuple ⟨Value ,-⟩ to the table T[I(X1, X2, ..., Xn)]
    endforeach
  endforeach
  sort table T[I(X1, X2, ..., Xn)] on the Value field
  Cut := determine_all_cut-points(T[I(X1, X2, ..., Xn)])
  MinWE := +∞
  foreach cut-point α in Cut do
    determine the left and right intervals [l1,l2], [r1,r2] with l2<α<r1
    if Seed_value∈ [l1 .. l2] then Admissible_interval := [l1, l2] else Admissible_interval := [r1, r2] endif
    WE := weighted_entropy(T[I(X1, X2, ..., Xn)], Admissible_interval)
    if WE < MinWE then
      Best_interval:=Admissible_interval
      MinWE := WE
    endif
  endforeach
if MinWE ≠ +∞ then return [I(X1, X2, ..., Xn)∈ Best_interval] else return nil endif
```

Fig. 1. Choice of the best range of numerical descriptors.

two disjoint intervals: the left interval $[l_1, l_2]$ and the right interval $[r_1, r_2]$. The lower bound of the left interval is the smallest value in the table with sign $+$, while the upper bound in the same interval is the largest value in the table that does not exceed the threshold α. On the contrary, the lower bound of the right interval is the smallest value in the table that exceeds α, while the upper bound is the largest value with sign $+$. When one of the two intervals contains no positive value, then it is set to *undefined*. However, at least one of the two intervals must be defined, since the table contains at least one value $+$ corresponding to the *Seed_value*, that is the value taken by $f(X_1, X_2, \ldots, X_n)$ in the seed example.

Not all definite intervals have to be considered, since the specialized clause $G, f(X_1, X_2, \ldots, X_n) \in$ Range for a given *Range* might no longer cover the seed example e^+. Those definite intervals that include the *Seed_value* are said to be *admissible*, because they guarantee that the corresponding specializations still cover the seed example e^+. When the condition of admissibility holds for an interval, the weighted entropy is computed and compared to the minimum weighted entropy found up to that moment.

By looking at the table as a source of messages labelled $+$ and $-$, then the expected information on the class membership conveyed from a randomly selected message is:

$$\text{info}(n^+, n^-) = -\frac{n^+}{n^+ + n^-} \log_2 \frac{n^+}{n^+ + n^-} - \frac{n^-}{n^+ + n^-} \log_2 \frac{n^-}{n^+ + n^-}$$

where n^+ and n^- are the number of values in the table with positive and negative signs, respectively.

Now consider a similar measurement after $T[f(X_1, X_2, \ldots, X_n)]$ has been partitioned into two subsets, S_1 and S_2, the former containing $n_1^+ + n_1^-$ values falling within an admissible interval, the latter containing

the remaining values. The information provided by S_1 will be close to zero when almost all cases have the same sign, $+$ or $-$. However, we are interested in intervals with a low entropy but a high number of positive examples. The following weighted entropy:

$$E(n_1^+, n_1^-) = \frac{n_1^-}{n_1^+}\text{info}(n_1^+, n_1^-)$$

does actually penalize those admissible intervals with a low percentage of positive examples. A good rule of thumb would be to choose the admissible interval that minimizes $E(n_1^+, n_1^-)$. As a concrete illustration, let us reconsider the table reported below

Value	0.5	0.7	0.9	1.0	1.5	1.5	1.5	1.7	2.5	2.5
Sign	$+$	$-$	$-$	$-$	$-$	$-$	$+$	$+$	$-$	$+$

There are four possible cut points that generate the following intervals:

α	0.60	1.25	1.60	2.10
$[l_1, l_2]$	[0.50 . . . 0.50]	[0.50 . . . 1.00]	[0.50 . . . 1.50]	[0.50 . . . 1.70]
$[r_1, r_2]$	[0.70 . . . 2.50]	[1.50 . . . 2.50]	[1.70 . . . 2.50]	[2.50 . . . 2.50]

Let us suppose that *Seed_value* equals 1.50. Then only those intervals including 1.50 are admissible, since they allow the specialized clause to cover the seed example. The weighted entropy for each of them is given in the table below.

Adm. interval	[0.70 . . . 2.50]	[1.50 . . . 2.50]	[0.50 . . . 1.50]	[0.50 . . . 1.70]
E	1.836592	1.000000	2.157801	1.590723

Thus, the best interval is the second one, with a weighted entropy equal to 1.0.

When the table is huge, there could be numerous cut-points. Only some of them will actually be considered, namely those between two consecutive distinct values with different signs (*boundary points*). This explains why the cut points 0.80 and 0.95 have not been considered. On the contrary, the cut-point 1.25 has been considered because the value 1.0 has a negative sign, while there is a value 1.5 with a positive sign. The reason for this choice is due to the following:

Theorem 1 (**Best cut-point**) If a cut-point α minimizes the measure, then α is a boundary point.

A proof of this theorem is reported in [16]. This result helps to discard several computations of the weighted entropy by considering only boundary points, so improving the efficiency of the procedure *determine_range*.

5. Model application: Recognition by probabilistic subsumption

Models induced by the system can be used to predict function values for new observations. More precisely, each clause

$$C : f(X_1, \ldots, X_n) = \text{Value} \leftarrow L_1, L_2, \ldots, L_m$$

of an induced model can be used forward as an inference rule: If conditions in the body are satisfied by the new observation, according to a grounding substitution θ, we can conclude that the function f takes the value *Value* when applied to $X_1\theta, \ldots, X_n\theta$. It is worthwhile observing that $X_1\theta, \ldots, X_n\theta$ are ground terms, since the property of range-restrictedness guarantees that when conditions in the body of a clause are satisfied, then the arguments of f are completely determined. For instance,

the body of the clause $f(X,Y) = a \leftarrow g(X) = b,\ h(Y) = c$ is satisfied by the observed facts $g(x) = b,\ h(y) = c,\ r(x,z) = d,\ g(z) = b$, according to the substitution $\theta = \{X \leftarrow x,\ Y \leftarrow y\}$ that grounds all literals in the clause. Thus, we can conclude that f takes the value a when applied to x and y.

The matching procedure adopted in this deductive step requires that all conditions in the body of a clause are satisfied by the observed facts, or, more technically, that the body of the clause θ-subsumes [4] the set of observed facts [24].

Definition 1 (**θ-subsumption**). *Let C and D be two clauses.*[5] *Then C θ-subsumes D, denoted as $C \leqslant \theta D$, if and only if there exists a substitution θ such that $C\theta \subseteq D$.*

The result of a θ-subsumption test is either true or false. If C denotes the space of clauses, then

θ-subsumption: $C \times C \rightarrow \{\text{false, true}\}$

However, this requirement might be too strict for real-world problems, because of their inherent vagueness. The presence of noise or the imprecision of the measuring instruments or the variability of the phenomenon described by the induced model often cause the subsumption test to fail. For this reason it becomes necessary to rely on a more flexible definition of subsumption that aims at comparing two descriptions in order to identify their similarities rather than their equality. The result of a flexible subsumption should produce a number in the unit interval [0,1] that indicates a degree of similarity between two clauses:

flexible-subsumption: $C \times C \rightarrow [0,1]$

such that, for any two clauses C and D,

i) flexible-subsumption$(C,D) = 1$ if C θ-subsumes D,
ii) flexible-subsumption$(C,D) \in [0,1)$ otherwise.

The definition of such a similarity measure should be based on a theory which is able to reason about chance and uncertainty, such as the probability theory. A fuzzy-logic approach would also be possible, although we are not aware of fuzzy pattern matching algorithms working on structural or relational representations like those dealt with in this paper.

In the probabilistic approach we can interpret the result of the flexible-subsumption function as the probability of C θ-subsuming D, provided that a change is made in D. More precisely, let D' be a ground clause obtained from D by means of some syntactic changes, such that C θ-subsumes D'. We can associate a probability to D', $P(D|D')$, representing the likelihood of observing D, given that the original observation was D'. Then, we can set

$$\text{flexible-subsumption}(C,D) = \max_{\text{def } D'\theta\text{-subsumed by } C} P(D|D')$$

[4] The canonical definition of θ-subsumption given by Plotkin [24] is appropriate for simple literals. However, it can be straightforwardly extended to set literals by requiring that the range of values of literals in C is a subset of the range of values of the corresponding literals in D.

[5] A clause $C = L_1 \vee L_2 \vee \ldots L_m$ is also considered as the set of its literals, that is, $C = \{L_1, L_2, \ldots, L_m\}$.

that is, *flexible-subsumption*(C, D) equals the maximum value of the likelihood computed over the space of clauses $D'\theta$-subsumed by C.

Therefore, one of the main problems is that of estimating $P(D|D')$ for all clauses $D'\theta$-subsumed by C. For instance, let us consider the following definite clause:

$$f(X, Y) = a \leftarrow g(X) = b,\ h(Y) = c$$

and the following set of observed facts:

$$D:\ g(x) = b,\ h(y) = d.$$

Let C denote the body of the definite clause. It can immediately be seen that C does not θ-subsume D. On the contrary,

$$D':\ g(x) = b,\ h(y) = c$$

is θ-subsumed by C, so that we can draw the conclusion $f(x, y) = a$ with probability equal to $P(D|D')$. This probability is the likelihood that the original observation was D', but we measured D because of noise. Before explaining how to compute $P(D|D')$ let us observe that there are many other clauses $D'\theta$-subsumed by C, therefore we have to search in the (possibly infinite) space of clauses θ-subsumed by C for that clause D' that maximizes $P(D|D')$. REFLEX, the system that implements a flexible-subsumption test, performs a branch-and-bound search that expands the least-cost partial path. The cost function is given by $1 - P(D|D')$.

Let D be the set of literals $\{L_1, L_2, \ldots, L_m\}$. Under the assumption that all facts $L_1, L_2, \ldots, L_m$ are conditionally independent, given D', the probability $P(D|D')$ can be defined as follows:

$$P(D|D') = \prod_{i=1}^{m} P(L_i|D')$$

where $P(L_i|D')$ denotes the probability of observing the ground fact L_i given D'. Suppose that L_i is $f(t_1, \ldots, t_n) =$ *Value*, then, if D' contains the literal $f(t_1, \ldots, t_n) =$ Value$'$, $P(L_i|D')$ is the probability that the real value was *Value*$'$, but we observed *Value*. We relate this probability to the type of domain of the function f (unordered, partially ordered, or totally ordered), as well as to the probability distribution of values in the function domain. When no information is available on the probability distribution of values, we make an assumption of uniform distribution. Formulae for the computation of $P(L_i|D')$ under this assumption are reported in [11, Section V], for several types of domains.

When D' does not contain a literal $f(t_1, \ldots, t_n) =$ Value$'$ we can say that the information on $f(t_1, \ldots, t_n)$ is missing or unknown. In this case $P(L_i|D')$ is computed as the expected value of $P(L_i|D' \cup \{f(t_1, \ldots, t_n) =$ Value$'\})$, where *Value*$'$ is the generic domain value, that is,

$$P(L_i|D') = \sum_{\text{Value}'} P(f(t_1, \ldots, t_n) = \text{Value}') \cdot P(L_i|D' \cup \{f(t_1, \ldots, t_n) = \text{Value}'\})$$

Once again, formulae for the computation of the expected value, under the assumption of uniform distribution are reported in [13, Section VII].

We conclude this section by observing that, under the assumption of uniform distribution, we have $P(D|D') > 0$ for any D'. This means that any ground instance of the head of a definite clause can be

Table 1
Experimental results for a set of 30 real business letters

Average number of errors		p-value wilcoxon signed	Number of clauses		average learning time	
Symbolic	Mixed	ranks test	Symbolic	Mixed	Symbolic	Mixed
3.6	2.9	0.3114	28.0	11.5	20:24	19:17

probabilistically entailed, although some instances are more likely than others. If we are interested in logical entailment with probability at least p, then the definition of p-subsumption can be considered.

Definition 2 (p-subsumption). *Let C and D be two clauses. Then C p-subsumes D, denoted as $C \leqslant pD$, if and only if flexible-subsumption$(C, D) \geqslant p$.*

In the application presented in the next section, a p-subsumption test to identify the logical components of some real business letters will be used.

6. Application to document image understanding

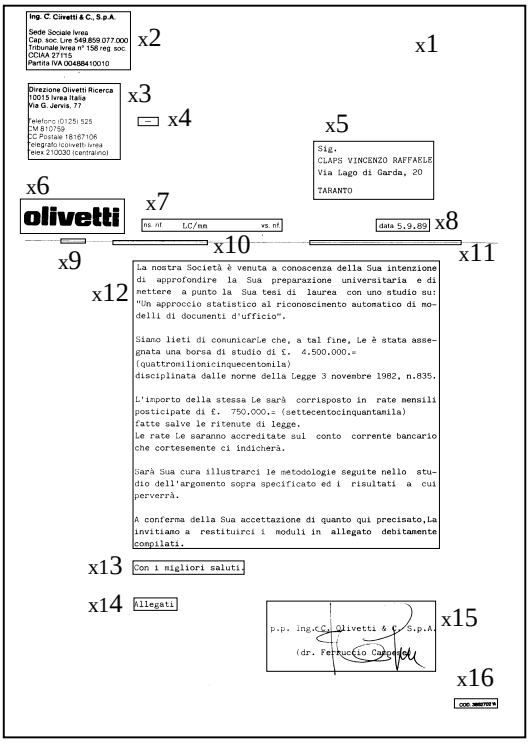
In this section the application of this approach to the field of document understanding is presented. The term *document image understanding* denotes the process of identification of logical components of a document and the subsequent extraction of relationships between them, such as the reading order based on spatial information and layout analysis results. Indeed, a document can be understood by means of its layout structure: this happens when the document has a standard format, as, for instance, business letters sent by a certain company. In this case, the date, the logo, the receiver of the letter and other semantically relevant parts can be easily located. Nevertheless, writing down the models useful to understand a particular type of document can be a demanding task, thus we adopted a different approach: We learn the models from a set of training documents [15]. In previous experiments we used only symbolic descriptors by discretizing numeric attributes, such as height, width and position of a block. Preliminary results were encouraging, but not totally satisfactory. One of the main issues seemed to be the prior discretization of numeric attributes. Since the current release of the learning systems are able to handle numerical descriptors as well, we decided to organize an experiment to test the improvement of the generated rules in terms of accuracy, learning time and simplicity.

In [16] the results obtained by INDUBI/CSL on a set of 112 single-page documents were reported. In this work, we present results concerning the application of ATRE to a set of 30 single-page documents, namely copies of business letters sent by a company. The page layout of each document is described with only symbolic descriptors or mixed numeric/symbolic descriptors (see Fig. 2). The logical components we are interested in recognizing are logo, sender, receiver, date, reference number, body and (possibly) signature of the sender.

Experimental results for a 10-fold cross-validation are summarized in Table 1. The significance test used is a non-parametric test, namely the Wilcoxon signed-ranks test [23], pairing across the folds for the cross-validations. The table shows that the average number of errors decreases, although not significantly, when numerical attributes are discretized on-line.

By decomposing the average number of errors into omission and commission errors[6] we can conclude

[6] Omission errors are made when some logical components in the test document are not identified, while commission errors are made when some layout components are given a wrong logical meaning.



part_of(x1,x2)=true, part_of(x1,x3)=true, ..., part_of(x1,x16)=true,
width(x2)= medium, width(x3)= medium, ..., width(x16)= medium_small,
height(x2)= medium_small, height(x3)= medium_small, ..., height(x16)= smallest,
type(x2)= text, type(x3)= text, ..., type(x16)= text, position(x2)= top_left,
position(x3)= top_left, ..., position(x8)= top_right, ..., position(x16)= bottom_right,
ontop(x2,x3)=true, ontop(x5,x8)=true, ..., ontop(x13,x14)=true,
toright(x6,x7)=true, toright(x3,x4)=true, ..., toright(x9,x10)=true,
aligned(x2,x3)=only_left_col,aligned(x6,x7)=only_lower_row, ...,
aligned(x13,x14)=only_left_col

part_of(x1,x2)=true, ..., part_of(x1,x16)=true,
width(x2)= 120.0, ..., width(x8)= 57.0, ..., width(x16)= 44.0,
height(x2)= 63.0, ..., height(x16)= 5.0,
type(x2)= text, ..., type(x16)= text,
x_pos_centre(x2)= 89.0, ..., x_pos_centre(x8)=478.0, ..., x_pos_centre(x16)=568.0,
y_pos_centre(x2)= 40.0, ..., y_pos_centre(x8)= 263.0, ..., y_pos_centre(x16)= 836.0,
ontop(x2,x3)=true, ..., ontop(x13,x14)=true,
toright(x6,x7)=true, ..., toright(x9,x10)=true,
aligned(x2,x3)=only_left_col, ..., aligned(x13,x14)=only_left_co

Fig. 2. A business letter and its layout descriptions, symbolic (up) and mixed (down).

that rules generated from numeric/symbolic descriptions made a significantly lower number of commission errors (0.3 vs. 1.5, p-value $= 0.0277$), and slightly increased the number of omission errors (2.6 vs. 2.1, p-value $= 0.7353$). Since, in our application, commission errors are considered more serious than omission errors, we can conclude that the handling of numerical attributes was actually beneficial. As to the other parameters, we observe that the introduction of numerical descriptors simplified the models (see the average number of clauses) and reduced the learning time (expressed in minutes).

The increase in omission errors is due to the presence of literals of the type $f(X_1, \ldots, X_n) \in [a \ldots b]$, that often miss the match against an instance, since the value taken by f is either a little higher than b or a little lower than a. For instance, in one of the ten trials of the previous experiment, the system produced the following definition of *logic_type* for the value *date*:

$$\begin{aligned}
&\text{logic_type}(X_1) = \text{date} \leftarrow \text{width}(X_1) \in [42.0 \ldots 97.0],\ x\text{-pos-centre}(X_1) \in [480.0 \ldots 525.0] \\
&\text{logic_type}(X_1) = \text{date} \leftarrow y\text{-pos-centre}(X_1) \in [262.0 \ldots 279.0],\ \text{aligned}(X_3, X_1) = \text{both_rows}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{aligned}(X_2, X_3) = \text{both_rows} \\
&\text{logic_type}(X_1) = \text{date} \leftarrow \text{aligned}(X_3, X_1) = \text{only_lower_row},\ \text{aligned}(X_2, X_1) = \text{both_rows} \\
&\text{logic_type}(X_1) = \text{date} \leftarrow x\text{-pos-centre}(X_1) \in [453.0 \ldots 525.0], \\
&\qquad\qquad\qquad\qquad\qquad\qquad y\text{-pos-centre}(X_1) \in [266.0 \ldots 276.0]
\end{aligned}$$

However, none of the bodies of the four clauses above θ-subsumes the numeric/symbolic description of the document in Fig. 2. In fact, the first clause misses the test because the function *x-pos-centre* takes the value 478.0 for the layout component $x8$ corresponding to the logical component date, while the range of possible values is $[480.0 \ldots 525.0]$. On the contrary, the fourth clause misses the test because the function *y-pos-centre* takes the value 263.0 for the argument $x8$, while the range of possible values is $[266.0 \ldots 276.0]$.

For this reason, we decided to match the induced models against test cases using a p-subsumption test instead of the traditional θ-subsumption. The results obtained with a 0.99-subsumption test are the following: 0.4% of commission errors and 2.2% of omission errors. Thus, during the recognition phase, the system has been able to reduce the rate of omission errors to that obtained with symbolic descriptions, with a small increase of commission errors. Such an increase is mainly due to the proximity of layout components labeled as reference number (see block $x7$ in Fig. 2) to the other layout components without any logical meaning (see block $x10$ in Fig. 2). We can conclude that the threshold of a probabilistic subsumption test depends on the value of the function to be predicted. In this experiment we have defined a unique threshold for all p-subsumption tests, but for the future we plan to learn the best p automatically from the same training set used to build the models.

7. Conclusions

Handling both numerical and symbolic data is an important issue for the successful application of first-order learning systems to real world problems. In the paper an operator for the on-line discretization of a homeric attribute/relation and a flexible matching predicate between definite clauses have been presented and tested in the field of document understanding. These techniques increase the sensitivity of the learner, reducing the commission errors without simultaneously increasing omission errors.

For the future, we plan to investigate an extension of current ILP techniques to aggregated data, as done in symbolic data analysis. The main difficulty met in applying current ILP techniques to aggregated data is that variables are set-valued or modal. Indeed, in the case of symbolic data, each observation is actually the description of a class of individuals, while ILP studies follow the traditional assumption of all machine learning studies, according to which observations are points of a feature space while discrimination rules define regions that enclose them. This work will give a new perspective to Diday's initial work on *hordes* [10] and might be a distinct step forward both in ILP and in Symbolic Data Analysis.

Acknowledgments

Thanks to Lynn Rudd for her help in reading the paper.

References

[1] A. Agresti, *Categorical Data Analysis*, Wiley, New York, NY, 1990.
[2] H. Almuallin, Y. Akiba and S. Kaneda, On handling tree-structured attributes in decision tree learning, *Proceedings of the 12th International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, 1995, pp. 12–20.
[3] É. Alphonse and C. Rouveirol, Lazy Propositionalisation for Relational Learning, in: *Proceedings of the 14th European Conference on Artificial Intelligence ECAI'00*, W. Horn, ed., IOS Press, 2000, pp. 256–260.
[4] F. Bergadano and R. Bisio, Constructive learning with continuous-valued attributes, in: *Uncertainty and Intelligent Systems*, B. Bouchon, L. Saitta and R.R. Yager, eds, Lecture Notes in Computer Science, 313, Springer-Verlag, Berlin, 1988, pp. 154–162.
[5] H.H. Bock and E. Diday, *Analysis of Symbolic Data. Exploratory methods for extracting statistical information from complex data*, Series: Studies in Classification, Data Analysis and Knowledge Organization, vol. 15, Springer-Verlag, 2000.
[6] M. Botta and A. Giordana, Learning quantitative features in a symbolic environment, in: *Methodologies for Intelligent Systems*, Z.W. Ras and M. Zemankova, eds, Lecture Notes in Artificial Intelligence, 542, Springer-Verlag, Berlin, 1991, pp. 296–305.
[7] M. Botta, A. Giordana and R. Piola, FONN: Combining first order logic with connectionist learning, *Proceedings of the 14th International Conference on Machine Learning ICML'97*, 1997, pp. 48–56.
[8] M. Botta and R. Piola, Refining Numerical Constants in First Order Logic Theories, *Machine Learning* **38**(1/2) (2000), 109–132.
[9] P. Brito, Order structure of symbolic assertion objects, *IEEE Trans. on Knowledge and Data Engineering* **6**(5) (1994), 830–835.
[10] E. Diday, Knowledge representation and symbolic data analysis, in: *Knowledge, Data and Computer-Assisted Decisions*, M. Schader and W. Gaul, eds, Springer-Verlag, Berlin, 1990, pp. 17–34.
[11] E. Diday, Des objets de l'analyse des données à ceux de l'analyse des connaissance, in: *Induction symbolique et numérique à partir de données*, Y. Kodratoff and E. Diday, eds, Cépaduès-Éditions, Toulouse, 1991.
[12] S. Dzeroski, L. Todorovski and T. Urbancic, Handling real numbers in ILP: A step towards better behavioural clones (extended abstract), in: *Machine Learning: ECML95, Lecture Notes in Artificial Intelligence*, (Vol. 912), N. Lavrac and S. Wrobel, eds, Springer, Berlin, 1995, pp. 283–286.
[13] F. Esposito, D. Malerba and G. Semeraro, Classification in noisy environments using a distance measure between structural symbolic descriptions, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-14**(3) (1992), 390–402.
[14] F. Esposito, D. Malerba and G. Semeraro, Incorporating statistical techniques into empirical symbolic learning systems, in: *Artificial Intelligence Frontiers in Statistics*, D.J. Hand, ed., Chapman & Hall, London, 1993, pp. 168–181.
[15] F. Esposito, D. Malerba and G. Semeraro, Multistrategy Learning for Document Recognition, *Applied Artificial Intelligence* **8**(1) (1994), 33–84.
[16] F. Esposito, D. Malerba and F.A. Lisi, Machine Learning for Intelligent Processing of Printed Documents, *Journal of Intelligent Information Systems* **14**(2/3) (2000), 175–198.
[17] N. Lavrac, S. Dzeroski and M. Grobelnik, Nonrecursive definitions of relations with LINUS, in: *Machine Learning: EWSL-91, Lecture Notes in Artificial Intelligence*, (Vol. 482), Y. Kodratoff, ed., Springer-Verlag, Berlin, 1991, pp. 265–281.
[18] N. Lavrac and S. Dzeroski, *Inductive Logic Programming: techniques and applications*, Ellis Horwood, Chicester, UK, 1994.
[19] D. Malerba, G. Semeraro and F. Esposito, A multistrategy approach to learning multiple dependent concepts, in: *Statistics and Machine Learning: The Interface*, C. Taylor and R. Nakhaeizadeh, eds, Wiley, London, 1996, pp. 87–106.
[20] D. Malerba, F. Esposito and F.A. Lisi, Learning recursive theories with ATRE, in *Proc. of ECAI '98*, Brighton, UK, 1998, pp. 434–439.
[21] S. Muggleton, ed., *Inductive Logic Programming*, Academic Press, London, 1992.
[22] S. Muggleton, Inverse Entaiment and Progol, *New Generation Computing* **13** (1995), 245–286.
[23] M. Orkin and R. Drogin, *Vital Statistics*, McGraw Hill, New York, NY, 1990.
[24] G.D. Plotkin, *Automatic methods of inductive inference*, PhD thesis, Edinburgh University, August 1971.
[25] R. Quinlan, Learning logical definitions from relations, *Machine Learning* **5** (1990), 239–266.

[26] J.R. Quinlan and R.M. Cameron-Jones, FOIL: A midterm report, in: *Machine Learning: ECML-93, Lecture Notes in Artificial Intelligence,* (Vol. 667), P.B. Brazdil, ed., Springer-Verlag, Berlin, 1993, pp. 3–20.
[27] C. Rouveirol, Flattening and saturation: Two representation changes for generalization, *Machine Learning* **14** (1994), 219–232.
[28] M. Sebag and C. Rouveirol, Constraint inductive logic programming, in: *Advances in ILP,* L. de Raedt, ed., IOS Press, 1996.
[29] A. Srinivasan and R.C. Camacho, Numerical reasoning with an ILP system capable of lazy evaluation and customised search, *Journal of Logic Programming* **40** (1999), 185–214.
[30] J.-D. Zucker and J.-G. Ganascia, Learning structurally indeterminate clauses, in: *Proc. of the 8th Int. Conference on Inductive Logic Programming, Lecture Notes in Artificial Intelligence 1446,* D. Page, ed., Springer-Verlag, 1998, pp. 235–244.