Machine Learning, 55, 175–210, 2004

Inducing Multi-Level Association Rules from Multiple Relations

FRANCESCA A. LISI lisi@di.uniba.it
DONATO MALERBA malerba@di.uniba.it
Dipartimento di Informatica, Università degli Studi di Bari, Via Orabona 4, I-70125 Bari, Italy

Editor: Céline Rouveirol and Michèle Sebag

Abstract. Recently there has been growing interest both to extend ILP to description logics and to apply it to knowledge discovery in databases. In this paper we present a novel approach to association rule mining which deals with multiple levels of description granularity. It relies on the hybrid language $\mathcal{AL}$-log which allows a unified treatment of both the relational and structural features of data. A generality order and a downward refinement operator for $\mathcal{AL}$-log pattern spaces is defined on the basis of query subsumption. This framework has been implemented in SPADA, an ILP system for mining multi-level association rules from spatial data. As an illustrative example, we report experimental results obtained by running the new version of SPADA on geo-referenced census data of Manchester Stockport.

Keywords: inductive logic programming, description logics, spatial data mining

1. Introduction

In its original formulation, Inductive Logic Programming (ILP) is concerned with inducing classification rules from examples and background knowledge, all of which are expressed as Prolog programs (Nienhuys-Cheng & de Wolf, 1997). This uniformity of representation is relatively unique within the diverse field of machine learning and has contributed significantly to the identity and coherence of ILP as a field of research. Recently the field has witnessed a twofold evolution, both in the target language and the learning problems.

First, the ILP community is becoming increasingly aware that today's ILP encompasses the application of machine learning methods to domains with flexible nested structures (Flach & Džeroski, 2001). This explains the growing interest in new target languages. *Description Logics* (DLs) are particularly interesting because they have been invented for representing and reasoning with structural knowledge and concept hierarchies (Baader et al., 2003). They represent a function-free first-order fragment allowing a variable-free syntax, which is considered to be important for reasons of readability. Deduction in DLs has been thoroughly investigated. Learning Horn rules also has already reached a mature state in ILP. But inducing DL descriptions from examples has been attacked mostly by heuristic means (Cohen, Borgida, & Hirsh, 1992; Cohen & Hirsh, 1994; Kietz & Morik, 1994) and very recently in a formal manner (Badea & Nienhuys-Cheng, 2000). Horn clausal logic and description logics are incomparable with respect to expressiveness (Borgida, 1996). Furthermore the former are significantly limited being unable to model and reason about

value restrictions in domains with a rich hierarchical structure. Thus there have been several attempts at combining description logics and function-free Horn clausal logic, e.g. $\mathcal{AL}$-log (Donini et al., 1998) and CARIN (Levy & Rousset, 1998). Rouveirol and Ventos (2000) have presented a coverage test and a hypothesis ordering for learning in CARIN-$\mathcal{ALN}$.

Second, many promising applications of ILP to *knowledge discovery in databases* (KDD) have emerged in the literature (Džeroski, 1996, 2001). KDD has been defined as the non-trivial process of discovering valid, novel, potentially useful and ultimately understandable patterns from data (Fayyad, Piatetsky-Shapiro, & Smyth, 1996). This new application area has broadened the range of learning problems of ILP from the traditional predictive tasks to the new descriptive ones. For instance, the ILP system WARMR (Dehaspe & Toivonen, 1999) supports the discovery of frequent DATALOG patterns. WARMR adapts Mannila and Toivonen's *levelwise method* (1997) to the case of spaces of function-free conjunctive formulas which are organized according to θ-subsumption (Plotkin, 1970). Furthermore, it adopts the logical setting of *learning from interpretations* (De Raedt & Džeroski, 1994) which De Raedt and Dehaspe (1997) proved suitable for descriptive data mining and Blockeel et al. (1999) proved good at scaling up ILP algorithms. Frequent patterns are commonly post-processed into rules that exceed given threshold values. In the case of association rules, the measures of support and confidence offer a natural way of pruning weak rules (Agrawal & Srikant, 1994).

Most studies in association rule mining have focused on mining rules at single concept levels, i.e. either at the primitive level or at a rather high concept level. Yet many applications would benefit from concept hierarchies that are often available as part of the domain knowledge (Han & Fu, 1999). Due to the evolution in the expressiveness of target languages mentioned above, we claim that the discovery of multi-level association rules is one of those data mining problems to which ILP can supply an elegant solution. In this paper we propose a novel ILP setting which adopts $\mathcal{AL}$-log as a knowledge representation language. This setting shows to be suitable for mining multi-level association rules from multiple relations because it allows a unified treatment of both the relational and structural features of data. A generality order $\succeq_{\mathcal{B}}$ and a downward refinement operator $\rho_{\mathcal{O}}$ for $\mathcal{AL}$-log pattern spaces is defined on the basis of query subsumption. We prove the monotonicity of $\succeq_{\mathcal{B}}$ with respect to the support of $\mathcal{AL}$-log patterns.

This setting has been implemented in SPADA, an ILP system designed and developed for mining multi-level association rules in spatial databases and applied to geographic data mining (Malerba & Lisi, 2001b). The two-phased discovery of association rules is illustrated by giving an insight into algorithmic issues. As an illustrative example, we report experimental results obtained by running the new version of SPADA on geo-referenced census data of Stockport, one of the ten metropolitan districts of Greater Manchester, UK. Thus the paper updates the algorithmic issues of Malerba and Lisi (2001b) and the experimental results of Malerba and Lisi (2001a).

The paper is organized as follows. Section 2 introduces the task of mining multi-level association rules from multiple relations. Section 3 briefly describes syntax, semantics and reasoning of $\mathcal{AL}$-log adapting them to represent data and patterns in our context. Section 4 is devoted to the presentation of the generality order for organizing spaces of $\mathcal{AL}$-log patterns. Section 5 describes the ILP system SPADA whereas experimental results on geo-referenced census data are discussed in Section 6. Concluding remarks are given in Section 7.

2. The mining task

Most studies in association rule mining have focused on mining rules at single concept levels, i.e., either at the primitive level or at a rather high concept level. Yet *concept hierarchies* are a valuable kind of domain knowledge to be exploited during the pattern discovery for two main reasons. First, it is more likely to discover interesting rules at low concept levels than at high ones. For instance, with reference to the concept hierarchy $\mathcal{H}_1 = \{\texttt{Milk} \sqsubset \texttt{Food}, \texttt{Bread} \sqsubset \texttt{Food}, \texttt{LowFatMilk} \sqsubset \texttt{Milk}, \texttt{ChocoMilk} \sqsubset \texttt{Milk}, \texttt{WhiteBread} \sqsubset \texttt{Bread}, \texttt{WheatBread} \sqsubset \texttt{Bread}\}$[1] over food items in market basket analysis, besides finding 80% of customers that purchase `Milk` may also purchase `Bread`, it could be informative to show that 75% of people buy `WheatBread` if they buy `LowFatMilk`. The association in the latter statement, though it occurs less frequently, carries more specific and concrete information than the former. Second, large support is more likely to exist at high concept levels rather than at low ones. E.g. suppose that the concept hierarchies $\mathcal{H}_2 = \{\texttt{Outerwear} \sqsubset \texttt{Clothes}, \texttt{Shirts} \sqsubset \texttt{Clothes}, \texttt{Jackets} \sqsubset \texttt{Outerwear}, \texttt{SkiPants} \sqsubset \texttt{Outerwear}\}$ and $\mathcal{H}_3 = \{\texttt{Shoes} \sqsubset \texttt{Footwear}, \texttt{HikingBoots} \sqsubset \texttt{Footwear}\}$ are available. One can notice that few people buy jackets with hiking boots, but many people may buy outerwear with hiking boots. Thus the association involving the intermediate category `Outerwear` would not be discovered if the search for large itemsets was restricted to the leaf-level of taxonomies. The need for ad-hoc algorithms has been observed by many researchers. In Han and Fu (1995) a top-down progressive deepening method for mining *multi-level association rules* has been developed by extending the Apriori algorithm (Agrawal & Srikant, 1994) for mining single-level association rules. A major difference between this and others' proposals, e.g. Srikant and Agrawal (1995), is the usage of different thresholds of support and confidence for different concept levels. The method first finds large itemsets at the top-most level and then progressively deepens the mining process towards lower concept levels under the assumption that only the descendants of frequent itemsets are worthy being generated.

A more rigorous formulation of the problem of mining multi-level association rules includes some taxonomic information $\mathcal{T} = \{\mathcal{H}_k\}_{1 \leq k \leq m}$ besides the data set $\mathbf{r}$ to be mined. It is noteworthy that each concept hierarchy $\mathcal{H}_k$ in $\mathcal{T}$ can arrange its concepts according to its own range of concept levels. Furthermore, data is typically available at leaf levels. This makes it hard to generate and evaluate patterns that combine concepts belonging to different hierarchies. For the sake of uniformity, we map concepts to levels of description granularity whose number depends on the data mining problem $\mathcal{P}$ at hand.

Definition 2.1. Let $\Psi = \{1, \ldots, maxG\}$ be the set of levels of description granularity in $\mathcal{P}$. A *granularity assignment* is a relation ψ over $\mathcal{H}_k \times \Psi$ such that $\forall\, (C, h), (D, l) \in \psi$: if $C \sqsubset D$ then $h > l$.

Concepts marked with multiple granularity levels are simply replicated along the hierarchy they belong to, so that a layering of the taxonomy at hand is induced. We denote $\mathcal{T}^l$ the l-th layer, $l \in \Psi$, of a taxonomy $\mathcal{T}$. In figure 1 a three-layered taxonomy $\mathcal{T}$ is illustrated. All concept hierarchies in $\mathcal{T}$ have been rearranged according to the three problem-defined

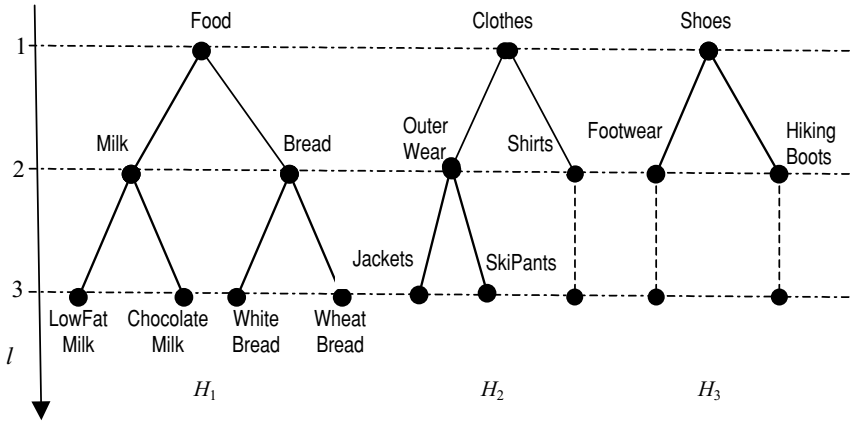

Figure 1. Assigning description granularity levels to concepts.

granularity levels. For instance, the concepts `Footwear` and `HikingBoots` in $\mathcal{H}_3$ have been assigned to both $\mathcal{T}^2$ and $\mathcal{T}^3$.

A pattern is an expression in some language describing a subset of data or a model applicable to that subset (Fayyad, Piatetsky-Shapiro, & Smyth, 1996). Given a taxonomy $\mathcal{T}$, we denote by $\mathcal{L}^l$ the language of patterns involving concepts in $\mathcal{T}^l$. This correspondence between $\mathcal{L}^l$ and $\mathcal{T}^l$ supplies means for getting both coarser-grained and finer-grained descriptions than a given pattern.

Definition 2.2. Let $P \in \mathcal{L}^l$. A pattern $P' \in \mathcal{L}^h$, $h < l$ (resp. $h > l$), is an *ancestor* (resp. *descendant*) of P iff it can be obtained from P by replacing each concept C that occurs in P with a concept $D \in \mathcal{T}^h$ such that $C \sqsubseteq D$ (resp. $D \sqsubseteq C$).

Patterns are evaluated against the data set $\mathbf{r}$. The *support* s of a pattern P in $\mathbf{r}$ is a percentage expressing the relative frequency of P in $\mathbf{r}$. It is computed by means of an evaluation function *supp* whose definition varies according to the chosen representation language. In our context the definition of frequent pattern takes description granularity levels into account.

Definition 2.3. Let $\mathbf{r}$ be a data set and $minsup^l$ the minimum support threshold for $\mathcal{L}^l$. A pattern $P \in \mathcal{L}^l$ with support s is *frequent* in $\mathbf{r}$, denoted as $freq(\mathbf{r}, P)$, if (i) $s \geq minsup^l$ and (ii) all ancestors of P w.r.t. $\mathcal{T}$ are frequent.

Formally, the problem $\mathcal{P}$ of discovering frequent patterns at multiple levels of description granularity can be defined as follows.

Definition 2.4. Given

– a data set $\mathbf{r}$,
– a taxonomy $\mathcal{T}$ where a reference concept $\hat{C}$ and m task-relevant concepts R_k, $1 \leq k \leq m$, are designated,

– a set $\{\mathcal{L}^l\}_{1 \leq l \leq maxG}$ of languages
– a set $\{minsup^l\}_{1 \leq l \leq maxG}$ of support thresholds

the problem of *frequent pattern discovery at l levels of description granularity*, $1 \leq l \leq maxG$, is to find the set $\mathcal{F}$ of all $P \in \mathcal{L}^l$ with $freq(\mathbf{r}, P)$.

As mentioned above, frequent patterns are post-processed into association rules.

Definition 2.5. Let $P, Q \in \mathcal{L}^l$ be such that $P \supset Q$. An *association rule* in $\mathcal{L}^l$ is an implication of the form $Q \rightarrow (P \setminus Q)(s, c)$, where s and c are percentages called the *support* and the *confidence* of the rule.

Given an association rule $Q \rightarrow R\ (s, c)$ and a data set $\mathbf{r}$, the support s is the support of the underlying pattern $Q \cup R$ in $\mathbf{r}$ whereas the confidence c is the probability that the consequent R occurs in $\mathbf{r}$ when the antecedent Q occurs in $\mathbf{r}$.

Definition 2.6. Let $\mathbf{r}$ be a data set and $minconf^l$ the minimum confidence threshold for $\mathcal{L}^l$. An association rule $Q \rightarrow R\ (s, c)$ in $\mathcal{L}^l$ is *highly-confident* if $c \geq minconf^l$. Furthermore, it is called *strong* if it is highly-confident and $Q \cup R$ is frequent.

Confidence is also computed by means of *supp*.

3. Representing data and patterns in $\mathcal{AL}$-log

$\mathcal{AL}$-log is a hybrid knowledge representation system which integrates the description logic $\mathcal{ALC}$ (Schmidt-Schauss & Smolka, 1991) and the deductive database language DATALOG (Ceri, Gottlob, & Tanca, 1990). Therefore it embodies two subsystems, called *structural* and *relational*, respectively. A fragment of $\mathcal{AL}$-log is actually used as a language for representing data and patterns in the context of multi-level association rule mining. Thus we limit the presentation of $\mathcal{AL}$-log to the features of interest to this work. Furthermore we assume the reader to be familiar with DATALOG.

3.1. Syntax

The description logic $\mathcal{ALC}$ allows for the specification of structural knowledge in terms of *concepts*, *roles*, and *individuals*. Individuals represent objects in the domain of interest. Concepts represent classes of these objects, while roles represent binary relations between concepts. Complex concepts can be defined from primitive concepts and roles by applying constructors such as $\sqcap$ (conjunction), $\sqcup$ (disjunction), and $\neg$ (negation). E.g., $C \sqcap D$ is the concept obtained by conjunction of the concepts C and D. The fragment of $\mathcal{ALC}$ of interest to this work contains only primitive concepts.

$\mathcal{ALC}$ knowledge bases consist of an intensional part and an extensional part. As for the *intensional* part, concept hierarchies spanned by is-a relations between concepts are

syntactically expressed as *inclusion statements* of the form

$C \sqsubseteq D$ (read "C is included in D")

where C and D are two arbitrary concepts. Intuitively, the statement says that every instance of C is also an instance of D. As for the *extensional* part, it is possible to specify instance-of relations between individuals and concepts. Syntactically, individuals are symbols of an alphabet $\mathcal{O}$ and instance-of relations are expressed as *membership assertions*, e.g. *concept assertions* of the form

$a : C$ (read "a belongs to C")

where $a \in \mathcal{O}$ and C is a concept. One such assertion says that a is an instance of C.

An $\mathcal{ALC}$ knowledge base Σ is formally the pair $\Sigma = \langle \mathcal{T}, \mathcal{M} \rangle$ where $\mathcal{T}$ is a set of inclusion statements and $\mathcal{M}$ is a set of membership assertions.

In $\mathcal{AL}$-log the $\mathcal{ALC}$ component allows for the definition of DATALOG programs enriched with *constraints* of the form $s : C$ where s is either a constant or a variable, and C is an $\mathcal{ALC}$-concept. Note that the usage of concepts as typing constraints applies only to variables and constants that already appear in the clause. The symbol & separates constraints from DATALOG atoms in a clause.

Definition 3.1. A *constrained* DATALOG *clause* is an implication of the form

$$\alpha_0 \leftarrow \alpha_1, \ldots, \alpha_m \& \gamma_1, \ldots, \gamma_n$$

where $m \geq 0$, $n \geq 0$, α_i are DATALOG atoms and γ_j are constraints.

From now on, we will denote by $head(E)$ the head, and by $body(E)$ the body of a constrained DATALOG clause E. A *constrained* DATALOG *program* Π is a set of constrained DATALOG clauses.

Formally, an $\mathcal{AL}$-log knowledge base $\mathcal{B}$ is defined as the pair $\mathcal{B} = \langle \Sigma, \Pi \rangle$ where Σ is an $\mathcal{ALC}$ knowledge base and Π is a constrained DATALOG program. For a knowledge base to be acceptable, it must satisfy the following conditions:

- The set of DATALOG predicate symbols appearing in Π is disjoint from the set of concept and role symbols appearing in Σ.
- The alphabet of constants in Π coincides with the alphabet $\mathcal{O}$ of the individuals in Σ. Furthermore, every constant occurring in Π appears also in Σ.
- For every clause in Π, every variable occurring in the constraint part occurs also in the DATALOG part.

Note that these properties allow the notion of *substitution* to be straightforwardly extended to constrained DATALOG clauses.

Queries to $\mathcal{AL}$-log knowledge bases are special cases of Definition 3.1, namely they are constrained DATALOG clauses without head. Note that queries are existentially quantified conjunctive formulas.

3.2. Semantics

In $\mathcal{ALC}$ concepts are interpreted as subsets of a domain. More precisely, an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$) which maps each concept to a subset of $\Delta^{\mathcal{I}}$ such that equations like $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, and $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$ are satisfied. In order to assign a precise meaning to membership assertions, the interpretation function $\cdot^{\mathcal{I}}$ is extended to individuals by mapping them to elements of $\Delta^{\mathcal{I}}$ such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$. It is noteworthy that such restriction ensures that different individuals denote different objects in the domain of interest (see *unique names* assumption (Reiter, 1980)).

Models in $\mathcal{ALC}$ are defined as follows.

Definition 3.2. An interpretation $\mathcal{I}$ *satisfies*:

- a concept C if $C^{\mathcal{I}} \neq \emptyset$;
- an inclusion statement $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- a concept assertion $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- a knowledge base $\Sigma = \langle \mathcal{T}, \mathcal{M} \rangle$ if it satisfies both $\mathcal{T}$ and $\mathcal{M}$.

We say that an $\mathcal{ALC}$ knowledge base Σ *logically implies* ξ (denoted as $\Sigma \models \xi$), where ξ is either an inclusion statement or a membership assertion, if every model of Σ satisfies ξ.

By virtue of the unique names assumption, we can impose the following conditions on interpretations (models).

Definition 3.3. An $\mathcal{O}$-*interpretation* for an $\mathcal{ALC}$ knowledge base Σ is an interpretation $\mathcal{I}_{\mathcal{O}}$ such that $\mathcal{O} \subseteq \Delta^{\mathcal{I}}$ and for each $a \in \mathcal{O} : a^{\mathcal{I}} = a$. An $\mathcal{O}$-*model* for Σ is an $\mathcal{O}$-interpretation that is a model.

Note that $\mathcal{O}$-interpretations ($\mathcal{O}$-models) can be considered the $\mathcal{ALC}$ counterpart of Herbrand interpretations (models). Therefore we focus on them.

A model-theoretic semantics of $\mathcal{AL}$-log is determined by the interaction between the structural and the relational component. We call Π_D the set of DATALOG clauses obtained from the clauses of Π by deleting their constraints. An interpretation $\mathcal{J}$ for an $\mathcal{AL}$-log knowledge base $\mathcal{B} = \langle \Sigma, \Pi \rangle$ is the union of an $\mathcal{O}$-interpretation $\mathcal{I}_{\mathcal{O}}$ for Σ and an Herbrand interpretation $\mathcal{I}_{\mathcal{H}}$ for Π_D.

Definition 3.4. Let $\mathcal{B}$ be an $\mathcal{AL}$-log knowledge base. An interpretation $\mathcal{J} = \langle \mathcal{I}_{\mathcal{O}}, \mathcal{I}_{\mathcal{H}} \rangle$ is a *model* of $\mathcal{B}$ if $\mathcal{I}_{\mathcal{O}}$ is a model of Σ, and for each ground instance $\bar{\alpha}'\&\gamma'_1, \ldots, \gamma'_n$ of each clause $\bar{\alpha}\&\gamma_1, \ldots, \gamma_n$ in Π, either there exists one γ'_i, $i \in \{1, \ldots, n\}$, that is not satisfied by $\mathcal{J}$, or $\bar{\alpha}'$ is satisfied by $\mathcal{J}$.

The notion of *logical consequence* paves the way to the definition of answer set for queries.

Definition 3.5. An $\mathcal{AL}$-log knowledge base $\mathcal{B}$ *logically implies*:

- a ground atom α ($\mathcal{B} \models \alpha$), if every model of $\mathcal{B}$ satisfies α
- a ground constraint γ ($\mathcal{B} \models \gamma$), if every model of $\mathcal{B}$ satisfies γ

– a conjunction of ground atoms and ground constraints

$$F = \alpha_1 \wedge \cdots \wedge \alpha_m \& \gamma_1 \wedge \cdots \wedge \gamma_n \ (\mathcal{B} \models F)$$

if $\mathcal{B} \models \alpha_i, \forall\, i \in \{1, \ldots, m\}$, and $\mathcal{B} \models \gamma_j, \forall\, j \in \{1, \ldots, n\}$.

Recalling that a query is an existentially quantified conjunction of atoms and constraints we have:

Definition 3.6. Let $\mathcal{B}$ be an $\mathcal{AL}$-log knowledge base. An *answer* to the query Q is a ground substitution σ for the variables in Q. The answer σ is *correct* w.r.t. $\mathcal{B}$ if $Q\sigma$ is a logical consequence of $\mathcal{B}$ ($\mathcal{B} \models Q\sigma$). The *answer set* of Q in $\mathcal{B}$, denoted as *answerset*$(Q, \mathcal{B})$, contains all the correct answers to Q w.r.t. $\mathcal{B}$.

Query answering mechanisms are sketched in the following section.

3.3. Reasoning

$\mathcal{AL}$-log, like any knowledge representation system, supports specific kinds of reasoning.

The fundamental deduction to be performed in the structural component Σ of an $\mathcal{AL}$-log knowledge base consists of checking whether Σ logically implies an inclusion statement (i.e. $\Sigma \models C \sqsubseteq D$) or a membership assertion (i.e. $\Sigma \models o : C$). The former inference is called *subsumption check*, the latter *instance check*. Since the problem of logical implication can be reformulated as an unsatisfiability problem, we have that $\Sigma \models o : C$ if and only if $\Sigma \cup \{o : \neg C\}$ is unsatisfiable and $\Sigma \models C \sqsubseteq D$ if and only if $\Sigma \cup \{x : C \sqcap \neg D\}$ is unsatisfiable (where x is a variable not appearing in Σ). The technique proposed in Donini et al. (1998) for the general problem of checking the satisfiability of an $\mathcal{ALC}$ knowledge base $\Sigma = \langle \mathcal{T}, \mathcal{M} \rangle$ starts with the tableau branch $S = \mathcal{T} \cup \mathcal{M}$ and adds assertions to S by means of *propagation rules* such as

– $S \rightarrow_{\sqcup} S \cup \{s : D\}$ if

1. $s : C_1 \sqcup C_2$ is in S,
2. $D = C_1$ and $D = C_2$,
3. neither $s : C_1$ nor $s : C_2$ is in S

– $S \rightarrow_{\sqsubseteq} S \cup \{s : C' \sqcup D\}$ if

1. $C \sqsubseteq D$ is in S,
2. s appears in S,
3. C' is the NNF concept equivalent to $\neg C$
4. $s : \neg C \sqcup D$ is not in S

– $S \rightarrow_{\perp} \{s : \perp\}$ if

1. $s : A$ and $s : \neg A$ are in S, or
2. $s : \neg\top$ is in S,
3. $s : \perp$ is not in S

to be applied according to a certain strategy of calculus until either a contradiction is generated or an interpretation satisfying S can be easily obtained from it. It has been proved in Donini et al. (1998) that whatever choice of application (respecting the strategy) is made, the calculus terminates. In particular, it returns a *complete* and *clash-free* tableau branch (i.e. it terminates with a tableau branch that does not does not contain assertions of the form $s : \bot$) if and only if the initial tableau was satisfiable. This provides operational means for proving the satisfiability of an $\mathcal{ALC}$ knowledge base. E.g., coming back to the instance check problem, $\Sigma \models o : C$ if and only if the completion of $\Sigma \cup \{o : \neg C\}$ contains a clash. Analogously, as for the subsumption check problem, $\Sigma \models C \sqsubseteq D$ if and only if the completion of $\Sigma \cup \{x : C \sqcap \neg D\}$ (where x is a variable not appearing in Σ) contains a clash. Both reasoning problems are relevant for this paper. Query answering in $\mathcal{AL}$-log itself requires instance checks to be done as illustrated later on. Also subsumption checks are required by inference mechanisms that will be presented in Section 4.

The main reasoning service for $\mathcal{AL}$-log knowledge bases is *hybrid deduction* which is based on constrained SLD-resolution.

Definition 3.7. Let Q be a query $\leftarrow \beta_1, \ldots, \beta_m \,\&\gamma_1, \ldots, \gamma_n$, E a constrained DATALOG clause $\alpha_0 \leftarrow \alpha_1, \ldots, \alpha_m \& \xi_1, \ldots, \xi_h$, and θ the most general substitution such that $\alpha_0\theta = \beta_j\theta$ where $\beta_j \in \{\beta_1, \ldots, \beta_m\}$. The *resolvent* of Q and E with substitution θ is the query Q' having:

- $(\beta_1, \ldots, \beta_{j-1}, \alpha_1, \ldots, \alpha_m, \beta_{j+1}, \ldots, \beta_m)\theta$ as DATALOG part
- $\gamma'_1, \ldots, \gamma'_k$ as constraints obtained from $\gamma_1\theta, \ldots, \gamma_n\theta, \xi_1, \ldots, \xi_h$ by applying the following simplifications: couples of constraints $t : C$, $t : D$ are replaced by the equivalent constraint $t : C \sqcap D$.

The one-to-one mapping between constrained SLD-derivations and the SLD-derivations obtained by ignoring the constraints can be exploited to extend known results for DATALOG to $\mathcal{AL}$-log. Note that in $\mathcal{AL}$-log a derivation of the empty clause with associated constraints does not represent a refutation. It actually infers that the query is true in those models of $\mathcal{B}$ that satisfy its constraints. This is due to the *open-world assumption* according to which an $\mathcal{ALC}$ knowledge base (in particular, the assertional part) represents possibly infinitely many interpretations, namely its models. Therefore, in order to answer a query, it is necessary to collect enough derivations ending with a constrained empty clause such that every model of $\mathcal{B}$ satisfies the constraints associated with the final query of at least one derivation.

Definition 3.8. Let $\mathcal{B}$ be an $\mathcal{AL}$-log knowledge base and $Q^{(0)}$ the query $\leftarrow \beta_1, \ldots, \beta_m \& \gamma_1, \ldots, \gamma_n$ to $\mathcal{B}$. A *constrained SLD-refutation* for $Q^{(0)}$ in $\mathcal{B}$ is a finite set $\{d_1, \ldots, d_m\}$ of constrained SLD-derivations for $Q^{(0)}$ in $\mathcal{B}$ such that the following conditions hold:

1. for each derivation d_i, $i \in \{1, \ldots, m\}$, the last query $Q^{(n_i)}$ of d_i is a constrained empty clause;
2. for every model $\mathcal{J}$ of $\mathcal{B}$, there exists at least one derivation d_i, $i \in \{1, \ldots, m\}$, such that $\mathcal{J} \models Q^{(n_i)}$.

The notion of constrained SLD-refutation is used for *query answering* in $\mathcal{AL}$-log. It is a complete and sound method for answering *ground* queries (Donini et al., 1998).

Definition 3.9. Let $\mathcal{B}$ be an $\mathcal{AL}$-log knowledge base. An answer σ to a query Q is called a *computed answer* if there exists a constrained SLD-refutation for $Q\sigma$ in $\mathcal{B}$ (denoted as $\mathcal{B} \vdash Q\sigma$). The set of computed answers is called the *success set* of Q in $\mathcal{B}$.

Lemma 3.1. *Let Q be a ground query to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$. It holds that $\mathcal{B} \vdash Q$ if and only if $\mathcal{B} \models Q$.*

Actually, given *any* query Q, the success set of Q in $\mathcal{B}$ coincides with the answer set to Q in $\mathcal{B}$ (Donini et al., 1998). Indeed, it is straightforward to see that the usual reasoning methods for DATALOG allow us to collect in a finite number of steps enough constrained SLD-derivations for Q in $\mathcal{B}$ to construct a refutation—if any. In particular, derivations must satisfy both conditions of Definition 3.8. Checking the former is trivial. Conversely, the latter is verified iff, for every set of assertions $a_1 : C_1, \ldots, a_m : C_m$ such that $a_i : C_i$ appears as a constraint in $Q^{(n_i)}$, $\Sigma \cup \{a_1 : \neg C_1, \ldots, a_m : \neg C_m\}$ is unsatisfiable. Solving this problem requires performing at most k^m unsatisfiability checks on Σ, where k is the maximum number of constraints in each $Q^{(n_i)}$ and m is the number of constrained SLD-derivations constituting the refutation.

3.4. Our $\mathcal{AL}$-log framework

The main feature of our $\mathcal{AL}$-log framework for frequent pattern discovery is the extension of the unique names assumption from the semantic level to the syntactic one. Note that the unique names assumption holds naturally for ground constrained DATALOG clauses because the semantics of $\mathcal{AL}$-log adopts Herbrand models for the DATALOG part and $\mathcal{O}$-models for the constraint part. Indeed in $\mathcal{AL}$-log different constants denote distinct objects of the domain. Conversely the unique names assumption is not guaranteed in the case of non-ground constrained DATALOG clauses, e.g. different variables can be unified. We propose to impose the bias of Object Identity (Semeraro et al., 1998) on the $\mathcal{AL}$-log framework.

Assumption 3.1 (Object Identity)**.** In a formula, terms denoted with different symbols must be distinct, i.e. they represent different entities of the domain.

This bias can be the starting point for the definition of either an equational theory or a quasi-ordering for constrained DATALOG clauses. The latter option has been deeply investigated in Lisi, Ferilli, and Fanizzi (2002) for the case of DATALOG queries and shows to be more suitable for the purposes of this work. It relies on a restricted form of substitution whose bindings avoid the identification of terms:

Definition 3.10. A substitution σ is an *OI-substitution* w.r.t. a set of terms T iff $\forall\, t_1, t_2 \in T$: $t_1 \neq t_2$ yields that $t_1\sigma \neq t_2\sigma$.

From now on, we will assume substitutions to be OI-compliant.

Carrying on the presentation of our $\mathcal{AL}$-log framework, data is represented as an $\mathcal{AL}$-log knowledge base (assuming the restrictions on the $\mathcal{ALC}$ component mentioned in Section 3.1).

Example 3.1. As an illustrative example throughout the paper we consider a knowledge base representing spatial data of the Province of Bari, Italy. Concepts of interest to this example are `LargeTown`, `Road`, and `Water`. The intensional part of Σ contains concept hierarchies rooted in `Road` and `Water`:

```
MotorWay ⊑ Road
MainTrunkRoad ⊑ Road
RegionalRoad ⊑ Road

Sea ⊑ Water
River ⊑ Water
Lake ⊑ Water
```

The extensional part of Σ contains 11 concept assertions for `LargeTown` (e.g. `bari:LargeTown`), and several assertions for the sub-concepts of `Road` and `Water`, e.g.:

```
a14:MotorWay.
ss16:MainTrunkRoad.
adriatico:Sea.
```

The relational subsystem Π contains DATALOG facts such as

```
adjacent_to(bari,adriatico).
intersects(bari,a14).
intersects(bari,ss16).
```

that represent spatial relations between large towns and either roads or water bodies. The intensional part of Π is not available for this example.

Patterns are to be intended as unary conjunctive queries whose answer set contains individuals of the $\mathcal{ALC}$ concept $\hat{C}$ of reference. We have called them $\mathcal{O}$-queries.

Definition 3.11. Given a key constraint $\hat{\gamma} = X : \hat{C}$, an *$\mathcal{O}$-query* Q to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$ is a constrained DATALOG clause of the form

$$Q = q(X) \leftarrow \alpha_1, \ldots, \alpha_m \ \& \ X : \hat{C}, \gamma_2, \ldots, \gamma_n$$

where X is the *distinguished variable* and the remaining variables occurring in body (Q) are the *existential variables*.

From now on we denote by $key(Q)$ the key constraint $X : \hat{C}$ of an $\mathcal{O}$-query Q. A *trivial* $\mathcal{O}$-query is a constrained empty clause of the form $q(X) \leftarrow \& X : \hat{C}$.

Patterns are generated starting from a set $\mathcal{A}$ of atom templates, a key constraint $\hat{\gamma}$, and an additional set Γ of constraint templates. An atom template α specifies name and arity of the predicate and mode of its arguments. An instantiation of α is a DATALOG atom with predicate and arguments that fulfill the requirements specified in α. Constraint templates specify the concept name for concept assertions. In particular, by restricting Γ to constraints derived from $\mathcal{T}^l$ (thus denoted Γ^l), we define the language $\mathcal{L}^l$. Note that, by definition of $\mathcal{O}$-queries, patterns are *connected* (or range-restricted). To be well-formed patterns must be also *linked*. Note that connectedness and linkedness have been originally defined for definite clauses (Helft, 1987) but can be straightforwardly extended to constrained DATALOG clauses, then to $\mathcal{O}$-queries. These conditions of well-formedness guarantee the correctness of query answering, as well known in the fields of logic programming and databases.

Example 3.2. Following Example 3.1, suppose that we are interested in descriptions at two different granularity levels having `LargeTown` as reference concept, and `Road` and `Water` as task-relevant concepts. Let $\mathcal{A} = \{$`intersects(_,_)`, `adjacent_to(_,_)`$\}$, $\hat{\gamma}$ be the key constraint built on `LargeTown`, and Γ^1 and Γ^2 the sets of constraints derived from the two taxonomy layers $\mathcal{T}^1 = \{$`Road`, `Water`$\}$ and $\mathcal{T}^2 = \{$`MotorWay`, `MainTrunkRoad`, `RegionalRoad`, `Sea`, `River`, `Lake`$\}$, respectively. The trivial $\mathcal{O}$-query

$Q_0 =$ `q(X)` $\leftarrow$ `& X:LargeTown`

is valid for both $\mathcal{L}^1$ and $\mathcal{L}^2$. The following $\mathcal{O}$-queries belong to $\mathcal{L}^1$:

$Q_1 =$ `q(X)` $\leftarrow$ `intersects(X,Y)`
`& X:LargeTown, Y:Road`
$Q_2 =$ `q(X)` $\leftarrow$ `intersects(X,Y), intersects(X,Z)`
`& X:LargeTown, Y:Road, Z:Road`
$Q_3 =$ `q(X)` $\leftarrow$ `intersects(X,Y), adjacent_to(X,Z)`
`& X:LargeTown, Y:Road, Z:Water`

The following $\mathcal{O}$-queries belong to $\mathcal{L}^2$:

$Q_4 =$ `q(X)` $\leftarrow$ `intersects(X,Y)`
`& X:LargeTown, Y:Motorway`
$Q_5 =$ `q(X)` $\leftarrow$ `intersects(X,Y), intersects(X,Z)`
`& X:LargeTown, Y:Motorway, Z:MainTrunkRoad`
$Q_6 =$ `q(X)` $\leftarrow$ `intersects(X,Y), adjacent_to(X,Z)`
`& X:LargeTown, Y:Motorway, Z:Sea`

Note that Q_1, Q_2 and Q_3 are ancestors of Q_4, Q_5 and Q_6 respectively.

The evaluation of a pattern Q is based on the computation of answers to Q w.r.t. an $\mathcal{AL}$-log knowledge base $\mathcal{B}$. This requires the following extension of the notions of correct answer and computed answer to the case of $\mathcal{O}$-queries:

Definition 3.12. Let $\mathcal{B}$ be an $\mathcal{AL}$-log knowledge base. An *answer* to the $\mathcal{O}$-query Q is a ground substitution θ for the distinguished variable of Q. The answer θ is *correct* w.r.t. $\mathcal{B}$ if there exists at least one correct answer to $body(Q)\theta$ w.r.t. $\mathcal{B}$. The answer set of Q in $\mathcal{B}$, denoted as $answerset(Q, \mathcal{B})$, contains all the correct answers to Q w.r.t. $\mathcal{B}$.

Definition 3.13. Let $\mathcal{B}$ be an $\mathcal{AL}$-log knowledge base. An answer θ to an $\mathcal{O}$-query Q is called a *computed answer* if there exists at least one computed answer to $body(Q)\theta$ w.r.t. $\mathcal{B}$. The success set of Q in $\mathcal{B}$ contains all the computed answers to Q w.r.t. $\mathcal{B}$.

Note that $body(Q)\theta$ is itself a conjunctive query whose answers can be computed by means of constrained SLD-refutation. Therefore Lemma 3.1 and its consequences can be extended to $\mathcal{O}$-queries.

An example of query answering for $\mathcal{O}$-queries is illustrated in the following.

Example 3.3. Let us consider the $\mathcal{AL}$-log knowledge base $\mathcal{B}$ reported in Example 3.1 and the $\mathcal{O}$-queries reported in Example 3.2. We want to compute a correct answer to Q_1 w.r.t. $\mathcal{B}$. Let us try with $\theta = \{X/\texttt{bari}\}$. For θ to be a correct answer to Q_1 w.r.t. $\mathcal{B}$, we need to find at least one correct answer to

$$Q^{(0)} = \leftarrow body(Q_1)\theta$$
$$= \leftarrow \texttt{intersects(bari,Y)} \ \& \ \texttt{bari:LargeTown, Y:Road}$$

w.r.t. $\mathcal{B}$. Several refutations can be constructed for $Q^{(0)}$. One of them consists of the following single constrained SLD-derivation.

Let $E^{(1)}$ be `intersects(bari,a14)`. A resolvent for $Q^{(0)}$ and $E^{(1)}$ with substitution $\sigma^{(1)} = \{\texttt{Y/a14}\}$ is the constrained empty clause

$$Q^{(1)} = \leftarrow \ \& \ \texttt{bari:LargeTown, a14:Road.}$$

What we need to check now is that $\Sigma \cup \{\texttt{bari:LargeTown, a14:Road}\}$ is satisfiable. This check amounts to two unsatisfiability checks to be performed by applying the tableau calculus sketched in Section 3.3.

The first unsatisfiability check operates on the initial tableau $S^{(0)} = \Sigma \cup \{\texttt{bari:}\neg\texttt{LargeTown}\}$. The application of the propagation rule $\rightarrow_\bot$ to $S^{(0)}$ produces the tableau $S^{(1)} = \{\texttt{bari:}\bot\}$. Computation stops here because no other rule can be applied to $S^{(1)}$. Since $S^{(1)}$ is complete and contains a clash, the initial tableau $S^{(0)}$ is unsatisfiable.

The second unsatisfiability check operates on the initial tableau $S'^{(0)} = \Sigma \cup \{\texttt{a14:}\neg\texttt{Road}\}$. The only propagation rule applicable to $S'^{(0)}$ is $\rightarrow_\sqsubseteq$ with respect to the assertion $\texttt{MotorWay} \sqsubseteq \texttt{Road}$. It produces the tableau $S'^{(1)} = \Sigma \cup \{\texttt{a14:}\neg\texttt{Road}, \texttt{a14:}\neg\texttt{MotorWay}\sqcup\texttt{Road}\}$. By applying $\rightarrow_\sqcup$ to $S'^{(1)}$ with respect to the concept `Road` we obtain $S'^{(2)} = \Sigma \cup \{\texttt{a14:}\neg\texttt{Road}, \texttt{a14:Road}\}$ which presents an evident contradiction. Indeed the application of $\rightarrow_\bot$ to $S'^{(2)}$ produces the final tableau $S'^{(3)} = \{\texttt{a14:}\bot\}$, thus proving the unsatisfiability of $S'^{(0)}$.

These two results together prove the satisfiability of $\Sigma \cup \{\texttt{bari:LargeTown, a14:Road}\}$, then the correcteness of $\sigma = \{\texttt{Y/ a14}\}$ as an answer to $body(Q_1)\theta$ w.r.t. $\mathcal{B}$. Thus we can say that $\theta = \{\texttt{X/bari}\}$ is a correct answer to Q_1 w.r.t. $\mathcal{B}$.

Note that θ is also a correct answer to both Q_0 and Q_4 w.r.t. $\mathcal{B}$. In particular, refutations for Q_0 boil down to checking the implication $\Sigma \models \texttt{bari:LargeTown}$ whereas the refutation for Q_4 with substitution $\sigma^{(1)}$ checks the unsatisfiability of $\Sigma \cup \{\texttt{a14:}\neg\texttt{MotorWay}\}$ instead of $\Sigma \cup \{\texttt{a14:}\neg\texttt{Road}\}$. More precisely, in the case of Q_4, the propagation rule $\rightarrow_\perp$ can be applied to the initial tableau at the first step of calculus. This turns out into a shorter and simpler proof of satisfiability.

The support of a pattern is defined as the ratio between the number of individuals in $\hat{C}$ that satisfy the pattern and the total number of individuals in $\hat{C}$.

Definition 3.14. Let $\mathcal{B}$ be an $\mathcal{AL}$-log knowledge base, $P \in \mathcal{L}^l$. The *support* of P with respect to $\mathcal{B}$ is defined:

$$supp(P, \mathcal{B}) = \frac{|answerset(P, \mathcal{B})|}{|answerset(\hat{P}, \mathcal{B})|}$$

where $\hat{P}$ is the trivial $\mathcal{O}$-query $q(X) \leftarrow \&X : \hat{C}$ for $\mathcal{L}^l$.

Example 3.4. Let us consider the $\mathcal{AL}$-log knowledge base $\mathcal{B}$ reported in 3.1 and the $\mathcal{O}$-queries reported in Example 3.2. In Example 3.3 we have shown the detailed computation of a correct answer to Q_0, Q_1 and Q_4. Analogously all other correct answers can be obtained. We have that $answerset(Q_0, \mathcal{B})$ contains 11 answers (as many as the number of individuals for the concept `LargeTown`), $answerset(Q_1, \mathcal{B})$ contains 11 answers as well (since the conditions in the body of Q_1 are not strong enough to filter the individuals of `LargeTown`) and $answerset(Q_4, \mathcal{B})$ contains 6 answers. Therefore, $supp(Q_1, \mathcal{B}) = 100\%$ and $supp(Q_4, \mathcal{B}) = 54.5\%$.

4. Searching $\mathcal{AL}$-log pattern spaces

In the levelwise method, the space of patterns is searched one level at a time, starting from the most general patterns and iterating between *candidate generation* and *candidate evaluation* phases (Mannila & Toivonen, 1997). Since patterns are represented as $\mathcal{O}$-queries, we intend to characterize the test of generality between two patterns as a *query containment* (or *subsumption*) problem. Given the schema of a database and two queries Q_1 and Q_2, we say that Q_1 is contained in (or is subsumed by) Q_2 if in every possible state of the database the answer set of Q_1 is contained in the answer set of Q_2.

In contrast to subsumption of concepts, subsumption of conjunctive queries has not been extensively studied. In Levy and Rousset (1998) it is treated as a special case of existential entailment. Conversely, Donini et al. do not face the problem. The definition of a subsumption relation for $\mathcal{O}$-queries can not disregard the nature of $\mathcal{O}$-queries as a special case of constrained DATALOG clauses as well as the availability of an $\mathcal{AL}$-log knowledge base with respect to which these $\mathcal{O}$-queries are to be evaluated. Generalized subsumption (Buntine, 1988) has been introduced in ILP as a generality order for Horn clauses with

respect to background knowledge. We propose to adapt generalized subsumption to the $\mathcal{AL}$-log framework as follows.

Definition 4.1. Let Q be an $\mathcal{O}$-query, α a ground atom, and $\mathcal{J}$ an interpretation. We say that Q *covers* α under $\mathcal{J}$ if there is a ground substitution θ for Q ($Q\theta$ is ground) such that $body(Q)\theta$ is true under $\mathcal{J}$ and $head(Q)\theta = \alpha$.

Definition 4.2. Let P and Q be two $\mathcal{O}$-queries to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$. We say that P *$\mathcal{B}$-subsumes* Q if for every model $\mathcal{J}$ of $\mathcal{B}$ and every ground atom α such that Q covers α under $\mathcal{J}$, we have that P covers α under $\mathcal{J}$.

We can define a generality relation $\succeq_{\mathcal{B}}$ between $\mathcal{O}$-queries on the basis of $\mathcal{B}$-subsumption.

Definition 4.3. Let P and Q be two $\mathcal{O}$-queries to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$. We say that P is *at least as general as* Q under $\mathcal{B}$-subsumption, $P \succeq_{\mathcal{B}} Q$, iff P $\mathcal{B}$-subsumes Q. Furthermore, P is *more general than* Q under $\mathcal{B}$-subsumption, $P \succ_{\mathcal{B}} Q$, iff $P \succeq_{\mathcal{B}} Q$ and $Q \succeq_{\mathcal{B}} P$. Finally, P is *equivalent* to Q under $\mathcal{B}$-subsumption, $P \sim_{\mathcal{B}} Q$, iff $P \succeq_{\mathcal{B}} Q$ and $Q \succeq_{\mathcal{B}} P$.

The definition of $\mathcal{B}$-subsumption can be proved equivalent to another formulation, which turns out to be more convenient as an operational means for checking $\mathcal{B}$-subsumption.

Lemma 4.1. *Let P and Q be two $\mathcal{O}$-queries to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$ and σ a Skolem substitution for Q with respect to $\{P\} \cup \mathcal{B}$. We say that $P \succeq_{\mathcal{B}} Q$ iff there exists a ground substitution θ for P such that (i) $head(P)\theta = head(Q)\sigma$ and (ii) $\mathcal{B} \cup body(Q)\sigma \models body(P)\theta$.*

Theorem 4.1. *Let P and Q be two $\mathcal{O}$-queries to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$ and σ a Skolem substitution for Q with respect to $\{P\} \cup \mathcal{B}$. We say that $P \succeq_{\mathcal{B}} Q$ iff there exists a substitution θ for P such that (i) $head(P)\theta = head(Q)$ and (ii) $\mathcal{B} \cup body(Q)\sigma \vdash body(P)\theta\sigma$ where $body(P)\theta\sigma$ is ground.*

Proof: By Lemma 4.1, we have $P \succeq_{\mathcal{B}} Q$ iff there exists a ground substitution θ' for P, such that $head(P)\theta' = head(Q)\sigma$ and $\mathcal{B} \cup body(Q)\sigma \models body(P)\theta'$. Since σ is a Skolem substitution, we can define a substitution θ such that $P\theta\sigma = P\theta'$ and none of the Skolem constants of σ occurs in θ. Then $head(P)\theta = head(Q)$ and $\mathcal{B} \cup body(Q)\sigma \models body(P)\theta\sigma$. Since $body(P)\theta\sigma$ is ground, by Lemma 3.1 we have $\mathcal{B} \cup body(Q)\sigma \vdash body(P)\theta\sigma$, so the thesis follows. □

Note that condition (i) of Theorem 4.1 is guaranteed in the case of $\mathcal{O}$-queries belonging to the same language $\mathcal{L}$. Therefore the test of $\mathcal{B}$-subsumption boils down to query answering.

Example 4.1. With reference to Example 3.2 we want to check whether $Q_1 \succeq_{\mathcal{B}} Q_4$ holds. Let $\sigma = \{\texttt{X/a}, \texttt{Y/b}\}$ a Skolem substitution for Q_4 with respect to $\mathcal{B} \cup \{Q_1\}$ and θ the identity substitution for Q_1. The condition (i) is immediately verified. It remains to verify that (ii) $\mathcal{B} \cup \{\texttt{intersects(a,b)\&a:LargeTown, b:MotorWay}\} \models \texttt{intersects(a,b) \& a:LargeTown, b:Road}$. We try to build a constrained SLD-refutation for

$$Q^{(0)} = \leftarrow \texttt{intersects(a,b) \& a:LargeTown, b:Road}$$

in $\mathcal{B}' = \mathcal{B} \cup \{\texttt{intersects(a,b)\&a:LargeTown,b:MotorWay}\}$.

Let $E^{(1)}$ be $\texttt{intersects(a,b)}$. A resolvent for $Q^{(0)}$ and $E^{(1)}$ with the empty substitution $\sigma^{(1)}$ is the constrained empty clause

$$Q^{(1)} = \leftarrow \texttt{\& a:LargeTown, b:Road}$$

What we need to check is that $\Sigma' \cup \{\texttt{a:LargeTown, b:Road}\}$ is satisfiable. This check amounts to two unsatisfiability checks to be performed by applying the tableau calculus analogously to Example 3.3.

Having proved the satisfiability of $\Sigma' \cup \{\texttt{a:LargeTown, b:Road}\}$, we have proved the existence of a constrained SLD-refutation for $Q^{(0)}$ in $\mathcal{B}'$. Therefore we can say that $Q_1 \succeq_{\mathcal{B}} Q_4$. It is easy to check that $Q_4 \not\succeq_{\mathcal{B}} Q_1$. Analogously we can prove the existence of relations of $\mathcal{B}$-subsumption between the $\mathcal{O}$-queries listed in Example 3.2. In particular, $Q_1 \succeq_{\mathcal{B}} Q_2$ but $Q_2 \not\succeq_{\mathcal{B}} Q_1$ due to the object identity bias.

It can be easily proven that $\succeq_{\mathcal{B}}$ is a quasi-order for $\mathcal{O}$-queries. Furthermore $\succeq_{\mathcal{B}}$ is monotonic w.r.t. the evaluation function *supp*.

Lemma 4.2. *Let Q be an $\mathcal{O}$-query to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$. If $\theta \in answerset(Q, \mathcal{B})$ then, for every model $\mathcal{J}$ of $\mathcal{B}$, Q covers $head(Q)\theta$ under $\mathcal{J}$.*

Proposition 4.1. *Let P and Q be two $\mathcal{O}$-queries to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$. If $P \succeq_{\mathcal{B}} Q$ then $supp(P, \mathcal{B}) \geq supp(Q, \mathcal{B})$.*

Proof: Let $\theta \in answerset(Q, \mathcal{B})$. By Lemma 4.2, for every model $\mathcal{J}$ of $\mathcal{B}$, Q covers $head(Q)\theta$ under $\mathcal{J}$. Note that $P \succeq_{\mathcal{B}} Q$. By Theorem 4, there exists a substitution γ for P such that $head(P)\gamma = head(Q)$. By Definition 4.2, it holds that $\theta \in answerset(P\gamma, \mathcal{B})$. Since $answerset(Q, \mathcal{B}) \subseteq answerset(P\gamma, \mathcal{B})$ and γ simply renames the distinguished variable of P, the thesis follows from Definition 3.14. □

To sum up, the results obtained in this section say that, given a language $\mathcal{L}$ of $\mathcal{O}$-queries, $(\mathcal{L}, \succeq_{\mathcal{B}})$ is a quasi-ordered set. Therefore it can be searched by refinement operators.

Definition 4.4 (Nienhuys-Cheng & de Wolf, 1997). In a quasi-ordered set $(\mathcal{L}, \succeq)$, a *downward* (resp. *upward*) *refinement operator* is a mapping ρ (resp. δ) from $\mathcal{L}$ to $2^{\mathcal{L}}$ such that $\forall P \in \mathcal{L}\ \rho(P) \subseteq \{Q \in \mathcal{L} \mid P \succeq Q\}$ (resp. $\delta(P) \subseteq \{Q \in \mathcal{L} \mid Q \succeq P\}$).

From Proposition 4.1, it follows that downward refinement operators are of greater help in the context of frequent pattern discovery. Indeed, they drive the search towards patterns with decreasing support and enable the early detection of infrequent patterns. Furthermore we are interested in downward refinement operators for searching multiple pattern spaces, each of which corresponds to a different level of description granularity within the same discovery task.

Definition 4.5. Let $P \in \mathcal{L}^l$. The (downward) $\mathcal{AL}$-log refinement operator $\rho_{\mathcal{O}}$ is defined by the following refinement rules:

$\langle Lit \rangle$ Add an instantiation of an atom from $\mathcal{A}$ and instantiations of related constraints from Γ^l to $body(P)$.
$\langle \forall\, C \rangle$ Replace each constraint $\gamma_j = X : C$ in $body(P)$, except for $key(P)$, with a constraint $\gamma'_j \in \Gamma^{l+1}$ such that $\gamma'_j = X : D$ and $D \sqsubseteq C$.

The rule $\langle Lit \rangle$ helps moving within the pattern space $\mathcal{L}^l$ (*intra-space search*) whereas the rule $\langle \forall\, C \rangle$ helps moving from $\mathcal{L}^l$ to $\mathcal{L}^{l+1}$ (*inter-space search*). Both rules are intuitively correct. Given any $P \in \mathcal{L}^l$, they act only on $body(P)$. Thus condition (i) of Theorem 4.1 are satisfied. Furthermore, it is straightforward to notice that the application of $\rho_{\mathcal{O}}$ to P reduces the number of models of P in both cases. In particular, as for $\langle \forall\, C \rangle$, this intuition follows from Definition 3.2. So condition (ii) also is fulfilled.

From now on, we call k-patterns those patterns that have been generated by applying $\langle Lit \rangle$ k times to the trivial $\mathcal{O}$-query in $\mathcal{L}^l$. Also we assume that $minsup^l \leq minsup^{l-1}, l > 1$, as usual in multi-level association rule mining (Han & Fu, 1999). Two pruning conditions for a multi-level search space can be derived from Proposition 4.1.

Corollary 4.1. *Given an $\mathcal{AL}$-log knowledge base $\mathcal{B}$, a k-pattern P in $\mathcal{L}^l$ is infrequent if it is subsumed by either* (*i*) *an infrequent* $(k-1)$*-pattern in $\mathcal{L}^l$ or* (*ii*) *an infrequent k-pattern in $\mathcal{L}^{l-1}$.*

Condition (i) requires to test the containment in queries at the same description granularity level (*intra-space subsumption checks*) whereas condition (ii) demands for testing the containment in coarser-grained queries (*inter-space subsumption checks*). Because of Definition 2.3 the former are to be tested for each level l, while the latter only for $l > 1$.

Example 4.2. Let us consider the portion of space encompassing the $\mathcal{O}$-queries listed in Example 3.2. Note that $Q_0 \succeq_{\mathcal{B}} Q_1$, $Q_0 \succeq_{\mathcal{B}} Q_4$, $Q_1 \succeq_{\mathcal{B}} Q_2$, $Q_1 \succeq_{\mathcal{B}} Q_3$, $Q_1 \succeq_{\mathcal{B}} Q_4$, $Q_4 \succeq_{\mathcal{B}} Q_5$, $Q_4 \succeq_{\mathcal{B}} Q_6$, $Q_2 \succeq_{\mathcal{B}} Q_5$, and $Q_3 \succeq_{\mathcal{B}} Q_6$. In figure 2, edges indicate the direction of search according to the refinement operator $\rho_{\mathcal{O}}$. For instance the query Q_4 can be obtained by applying either $\langle Lit \rangle$ to Q_0 or $\langle \forall\, C \rangle$ to Q_1. Suppose now that Q_4 is a frequent pattern. It is refined into Q_5 by means of $\langle Lit \rangle$. If Q_2 was infrequent, Q_5 should be pruned according to Corollary 4.1(ii).

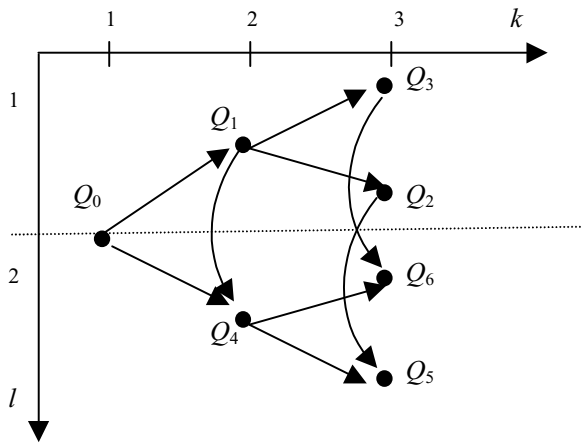

Figure 2. Portion of space searched by $\rho_{\mathcal{O}}$.

5. The ILP system SPADA

The $\mathcal{AL}$-log framework and the subsequent ILP setting have been implemented in SPADA, a system designed and developed for mining multi-level association rules in spatial databases and applied to geographic knowledge discovery (Malerba & Lisi, 2001b). Actually SPADA could tackle problems of multi-level association rule mining in any structured domain, i.e. any domain characterized by the presence of objects, properties of objects and relations between objects. We have focused our attention on the spatial domain because it stresses the potential of our approach from the application side.

A *spatial database* is a database system that offers spatial data types in its data model and query language and supports them in its implementation, providing at least spatial indexing and efficient algorithms for spatial join (Güting, 1994). Thus spatial databases supply an adequate representation of both single objects and spatially related collections of objects. In particular, the abstraction primitives for *spatial objects* are point, line and region. Among the operations defined over spatial objects, *spatial relationships* are the most important because they make it possible, e.g., to ask for all objects in a given relationship with a query object. The explicit location and/or extension of spatial objects also define implicit relations of spatial neighborhood that make knowledge discovery in spatial databases more difficult than in relational databases (Ester et al., 2000). This applies to both the efficiency of algorithms and the complexity of patterns.

With reference to Definition 2.4, the purpose of discovering spatial patterns is to detect associations between *reference objects* (individuals of the reference concept $\hat{C}$) and *task-relevant objects* (individuals of the m task-relevant concepts R_k, $1 \leq k \leq m$) in a given spatial database (the data set **r**). The former are the main subject of the description. The latter are relevant for the task at hand and spatially related to the former. It is noteworthy that specifying these objects enables the application of an Apriori-like algorithm. Indeed, task-relevant objects are like landmarks. They break the continuity of space and

define "transactions" around reference objects. In geographic knowledge discovery, spatial objects are geographical objects modeled in vectorized maps available in Geographical Information Systems (GIS), each R_k is typically a map layer and the taxonomy $\mathcal{T}$ is a collection of spatial hierarchies to be exploited to get geographic descriptions at different granularity levels. Note that spatial hierarchies capture is-a relations among geographical objects. A *spatial pattern* is a pattern P that contains at least one atom representing a spatial relationship. Extending Definition 2.5, a *spatial association rule* is an association rule that can be derived from two patterns P and Q where either P or Q is spatial. An example of association rule mining in geographic data is the discovery of associations between large towns ($\hat{C}$) and spatial objects taken from the layers of road network (R_1), hydrography (R_2) and administrative boundaries (R_3) in the Province of Bari, Italy (Malerba & Lisi, 2001b).

The connection of SPADA to spatial databases is made possible by a middle-layer module for *feature extraction* which selects data of interest and transforms them into ground DATALOG facts. In particular, map features can be either properties of spatial objects or relations that hold between spatial objects, and are extracted according to a predefined semantics (e.g. the 9-intersection model for topological relations (Egenhofer & Herring, 1994)). Furthermore, numerical features with a large domain need to be discretized. This is currently done by applying our implementation of the relative unsupervised discretization algorithm RUDE (Ludl & Widmer, 2000).

5.1. The main features of SPADA

SPADA has been developed in Sicstus Prolog and adopts DATALOG as a language for representing data and patterns.[2]

The main procedure of SPADA is reported in Algorithm 1. It implements a depth-bounded breadth-first search strategy. For each granularity level l (up to a user-defined maximum level $maxG$) and depth level k (up to a user-defined maximum depth $maxD$), the system discovers frequent association patterns by alternating candidate generation (procedure generateCandidates()) and candidate evaluation (procedure evaluateCandidates()). On demand, it turns them into strong association rules (procedure generateStrongRules()). Details of these procedures are given in Sections 5.2 and 5.3.

As for computational complexity, SPADA does not escape the notorious trade-off between expressiveness and efficiency in first-order representations (Džeroski, 1996). Related to efficiency is *scalability*. Studies on learnability theory have shown that current ILP algorithms would scale relatively well as the number of examples or facts in the background knowledge increases. However, they would not scale well with the number of arguments of the predicates (relations) involved, and in some cases with the complexity of the patterns being searched. Thus the use of *declarative bias* is usually suggested to improve scalability (De Raedt & Dehaspe, 1997; Weber, 1999). SPADA exploits the available background knowledge, notably the concept hierarchies of $\mathcal{T}$, and relies on a language bias specification to constrain the search for patterns, e.g. by setting the key constraint $\hat{\gamma}$. Although the atoms in $\mathcal{A}$ are listed, the constraints in Γ^l are derived from $\mathcal{T}$ according to the language bias directives.

mineMultiLevelAssociations($\mathcal{B}$, $maxG$, $maxD$,$\{\mathcal{L}^l, minsup^l, minconf^l\}_{1 \leq l \leq maxG}$)
1. $\mathcal{F} \leftarrow \emptyset$;
2. $\mathcal{R} \leftarrow \emptyset$;
3. $l \leftarrow 1$;
4. **while** $l \leq maxG$ **do**
5. $\quad \mathcal{I}^l \leftarrow \emptyset$;
6. $\quad k \leftarrow 1$;
7. $\quad \mathcal{C}_k^l \leftarrow \{$generateTrivialPattern($\mathcal{L}^l$)$\}$;
8. $\quad \mathcal{F}_k^l \leftarrow \emptyset$;
9. $\quad \mathcal{R}_k^l \leftarrow \emptyset$;
10. $\quad$ **while** $k \leq maxD$ **and** $\mathcal{C}_k^l \neq \emptyset$ **do**
11. $\quad\quad \mathcal{F}_k^l \leftarrow$ evaluateCandidates($\mathcal{B}$, $\mathcal{C}_k^l$, $\mathcal{I}^l$, $minsup^l$);
12. $\quad\quad \mathcal{F} \leftarrow \mathcal{F} \cup \mathcal{F}_k^l$;
13. $\quad\quad$ **if** $patterns2rules = yes$ **then**
14. $\quad\quad\quad \mathcal{R}_k^l \leftarrow$ generateStrongRules($\mathcal{F}_k^l$, $minconf^l$);
15. $\quad\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_k^l$;
16. $\quad\quad$ **endif**
17. $\quad\quad k \leftarrow k + 1$;
18. $\quad\quad \mathcal{C}_k^l \leftarrow$ generateCandidates($\mathcal{F}_{k-1}^l$, $\mathcal{L}^l$, $\mathcal{I}^l$);
19. $\quad$ **endwhile**
20. $\quad l \leftarrow l + 1$;
21. **endwhile**
return $\mathcal{F}$, $\mathcal{R}$

Algorithm 1. Main procedure of SPADA.

5.2. *From data to patterns*

In SPADA each search stage generates, then evaluates patterns. For a given l the candidate generation phase builds the set $\mathcal{C}_k^l$ of candidate k-patterns starting from the set $\mathcal{F}_{k-1}^l$ of frequent $(k-1)$-patterns and the language $\mathcal{L}^l$ by taking the set $\mathcal{I}^l$ of infrequent patterns into account. Candidate generation consists of a refinement step followed by a pruning step. The former applies one of the two rules of $\rho_{\mathcal{O}}$ to patterns previously found frequent by preserving the properties of linkedness and safety. The pruning step allows some infrequent patterns to be detected and discarded prior to evaluation. Note that the pruning conditions of Corollary 4.1 require a high number of subsumption checks to be performed. This makes candidate generation computationally expensive. So, we propose an implementation of the refinement operator $\rho_{\mathcal{O}}$ which uses a graph of backward pointers to be updated while searching in order to keep track of both intra-space and inter-space search stages. Figure 3 gives an example of such graph for the portion of space reported in figure 2. Here nodes, dotted edges and dashed edges represent patterns, intra-space parenthood and inter-space parenthood, respectively.

Algorithm 2 reports candidate generation in SPADA. It consists of two computation branches. The former concerns the case of search in $\mathcal{L}^1$. It applies either $\langle Lit \rangle$ (procedure

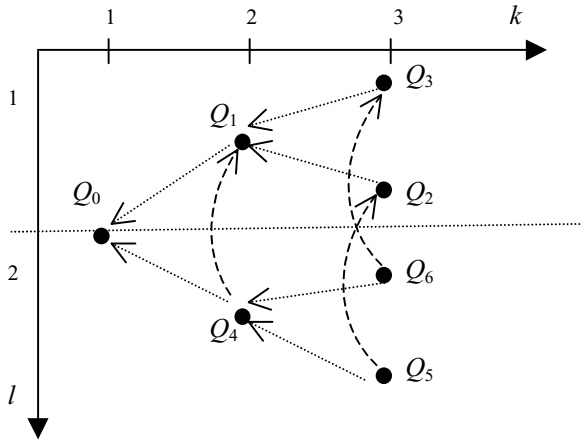

Figure 3. Graph of intra-space and inter-space backward pointers.

intraRefine()), performs the pruning step (procedure prune()) and inserts an intra-space backward pointer for each retained candidate (procedure `setIntraSpaceEdge()`). Note that, since the fragment of $\mathcal{AL}$-log of interest to us can be reduced to the aforementioned extended DATALOG, the rule $\langle Lit \rangle$ boils down to adding literals to the body of queries.

The other branch in Algorithm 2 concerns the case of search at levels of finer description granularity. One can expect that it is enough to simply replace the procedure intraRefine() with a procedure interRefine() which implements the refinement rule $\langle \forall\, C \rangle$. But things are more complicated. Let us suppose that the current space to be searched is $\mathcal{L}^l$, $l > 1$. On one side, searching $\mathcal{L}^l$ with only $\langle Lit \rangle$ implies to restart from scratch. Rather we would like to capitalize on the computational effort made when searching $\mathcal{L}^{l-1}$ and minimize the number of inter-space subsumption checks. On the other side, searching $\mathcal{L}^l$ indirectly, i.e. by applying $\langle \forall\, C \rangle$ to $\mathcal{O}$-queries found frequent in $\mathcal{L}^{l-1}$, implies the loss of the useful information that could be collected if $\mathcal{L}^l$ was searched directly. E.g., when generating $\mathcal{C}_k^l$, $l, k > 1$, it happens that intraRefine($\mathcal{F}_{k-1}^l$) $\subseteq$ interRefine($\mathcal{F}_k^{l-1}$). This means that a blind application of $\langle \forall\, C \rangle$ causes an increment of intra-space subsumption checks.

It is necessary to find a compromise between these two apparently irreconcilable solutions. Our solution requires that $\mathcal{C}_k^l$, $l, k > 1$, is computed taking both $\mathcal{F}_{k-1}^l$ and $\mathcal{F}_k^{l-1}$ into account. In particular, the expansion of a node P in $\mathcal{F}_{k-1}^l$ is done as follows:

- retrieve the inter-space parent node P' of P by following the inter-space backward pointer (step (13));
- retrieve the set $\mathcal{Q}' \subseteq \mathcal{F}_k^{l-1}$ of intra-space children nodes of P' by navigating intra-space backward pointers in reverse sense (step (14));
- generate the set $\mathcal{Q}$ of $\mathcal{O}$-queries obtained by applying $\langle \forall\, C \rangle$ to each Q' in $\mathcal{Q}'$ (step (16))

```
Procedure generateCandidates(F^l_{k-1}, L^l, var I^l)
1. C^l_k ← ∅;
2. if l = 1 then
       /* search in L^1 */
3.     foreach pattern P in F^l_{k-1} do
4.         Q ← intraRefine(P, L^l);
5.         Q ← prune(Q, I^l);
6.         foreach pattern Q in Q do
               /* set the intra-space edge from Q to P */
7.             setIntraSpaceEdge(Q, P)
8.         endforeach
9.         C^l_k ← C^l_k ∪ Q
10.    endforeach
11. else
       /* search in L^l, l > 1*/
12.    foreach pattern P in F^l_{k-1} do
13.        P' ← getInterSpaceParent(P);
14.        Q' ← getIntraSpaceChildren(P');
15.        foreach pattern Q' in Q' do
16.            Q ← interRefine(Q');
17.            Q ← prune(Q, I^l);
18.            foreach pattern Q in Q do
                   /* set the inter-space edge from Q to Q' */
19.                setInterSpaceEdge(Q, Q');
                   /* set the intra-space edge from Q to P
20.                setIntraSpaceEdge(Q, P);
21.            endforeach
22.            C^l_k ← C^l_k ∪ Q
23.        endforeach
24.    endforeach
return C^l_k
```

Algorithm 2. Candidate generation in SPADA.

This not only avoids a blind application of the refinement rule $\langle \forall\, C \rangle$ but also lightens the computational load during the pruning step.

Example 5.1. With reference to Example 4.2, note that the refinement of Q_4 into Q_5 via $\langle Lit \rangle$ is performed by applying $\langle \forall\, C \rangle$ to Q_2. We emphasize that the inter-space backward pointer from Q_4 to Q_1 enables the access to search stages of success in $\mathcal{L}^1$. This assures that Q_2 does not produce a certainly infrequent pattern because of Corollary 4.1(ii).

Candidate evaluation also benefits from backward pointers. They link any candidate Q to the frequent pattern P from which Q has been derived by applying the refinement operator.

We remind that $P \succeq_{\mathcal{B}} Q$, namely a relation of query containment holds between P and Q. In database practice, P is usually a *view* (i.e. a query whose answers are precomputed, stored and maintained up-to-date) and the relation $answerset(Q, \mathcal{B}) \subseteq answerset(P, \mathcal{B})$ is exploited to speed up query evaluation by filtering the answers of P instead of computing the answers of Q from scratch. We store the answerset of frequent patterns and use backward pointers to access this data quickly. Furthermore the evaluation of constraints that would trigger the deductive engine of $\mathcal{AL}$-log on the structural subsystem has been efficiently implemented as saturation of $\mathcal{T}^l$ against $\mathcal{M}$ to be performed at the beginning of each search stage.

5.3. *From patterns to association rules*

For given l and k, the procedure generateStrongRules() takes the set $\mathcal{F}_k^l$ of frequent patterns as input and returns the set $\mathcal{R}_k^l$ of strong association rules (see Algorithm 3). In particular, given a frequent pattern P, the procedure generateAntecedents() outputs antecedents suitable for rules being derived from P and the procedure `combine()` is responsible for using these antecedents to build rules according to Definition 2.5.

It is noteworthy that the generation of "good" rule antecedents is crucial. A naïve implementation of the procedure generateAntecedents() would consist of a combinatorial computation step followed by a pruning step. The former would output combinations of atoms occurring in P while the latter would discard those that are not well-formed. Backward pointers can be also exploited to speed up the generation of association rules instead. In SPADA the procedure generateAntecedents() recursively retrieves the predecessors of a frequent pattern. Only those yielding to strong rules are considered. This eliminates the need for evaluating rules a posteriori.

Example 5.2. Adapting the semantics of query extensions for association rules (Dehaspe & Toivonen, 1999) to our $\mathcal{AL}$-log framework, the following association rule having Q_4 as antecedent:

Procedure generateStrongRules($\mathcal{F}_k^l$, $minconf^l$)
1. $\mathcal{R}_k^l \leftarrow \emptyset$;
2. **foreach** pattern P in $\mathcal{F}_k^l$ **do**
3. $\quad \mathcal{Q} \leftarrow$ generateAntecedents(P);
4. $\quad$ **foreach** pattern Q in $\mathcal{Q}$ **do**
5. $\quad\quad$ **if** getSupport(P)/getSupport(Q)$\geq$ $minconf^l$ **then**
6. $\quad\quad\quad \mathcal{QE} \leftarrow$ combine(P, Q);
7. $\quad\quad$ **else** $\mathcal{QE} \leftarrow \emptyset$
8. $\quad\quad \mathcal{R}_k^l \leftarrow \mathcal{R}_k^l \cup \mathcal{QE}$
9. $\quad$ **endforeach**
10. **endforeach**
return $\mathcal{R}_k^l$

Algorithm 3. Rule generation in SPADA.

```
intersects(X,Y) & X:LargeTown, Y:Motorway⇝
     adjacent_to(X,Z) & Z:Sea
```

can be generated from Q_6. It is strong with support 36.36% and confidence 66.7%. Remember from Example 3.4 that the query Q_4 is frequent with support 54.5%.

5.4. *Related work*

The closest work to ours is WARMR (Dehaspe & Toivonen, 1999). But although it has been presented as a system able to use is-a hierarchies, WARMR is not a system for mining multi-level association rules because it lacks of mechanisms for dealing properly with structural knowledge. E.g., with reference to the problem in Example 3.1 and assuming that $\mathcal{L}$ is defined with the following WRMODE specification[3]:

```
{largeTown(-o), intersects(+o,-r), is_a(+r,road), is_a(+r,motorway),
..., adjacent_to(+o,-w), is_a(+w,water), is_a(+w,sea), ...}
```

WARMR can discover patterns such as the following DATALOG queries Q_1 and Q_2

```
?- largeTown(A), intersects(A,B), is_a(B,road)
?- largeTown(A), intersects(A,B), is_a(B,motorway)
```

Intuitively, looking at the concepts involved in the two queries, we can say that Q_1 is more general than Q_2 but θ-subsumption is not strong enough to catch this generality relation. It would be necessary to adopt a semantic generality relation instead of a syntactic one. A drawback of this is the inability of WARMR to perform any form of *taxonomic reasoning*. E.g., it generates the following DATALOG queries

```
?- largeTown(A), intersects(A,B), is_a(B,road)
?- largeTown(A), intersects(A,B), is_a(B,road), is_a(B,motorway)
```

by simply adding `is_a` atoms from $\mathcal{L}$ without detecting semantic redundancies. Furthermore, WARMR differs from SPADA as concerns the transformation of frequent patterns into association rules. In order to limit the number of possible rules, WARMR expects that the user provide a list of possible patterns that can occur in the antecedent or conseguent of a rule. This is done via the language bias directive `classes`. More specifically, given a frequent pattern P, if there are subsets Q and R such that $Q \subset P$ is listed in `classes` and $R = P \setminus Q$, then WARMR generates the rules $Q \rightarrow R$ and $R \rightarrow Q$. FARMER (Nijssen & Kok, 2001) is a faster version of WARMR. Here patterns are formulated as unary conjunctive queries in the form of rules as done in this paper and the search does not use θ-subsumption but manipulates a tree data structure to generate the queries which are defined by the language bias. Also FARMER does not support any form of taxonomic reasoning.

The work presented in Rouveirol and Ventos (2000) is the first attempt at learning in hybrid languages. In particular, the chosen language is CARIN-$\mathcal{ALN}$. Algorithms for testing

example coverage and subsumption between two hypotheses in the logical setting of *discriminant* induction from interpretations are proposed. They are based on the existential entailment algorithm studied in Levy and Rousset (1998). Unfortunately, no implementation is available yet for the method of Rouveirol and Ventos, so we could not compare it with SPADA.

The problem of mining multi-level association rules in geographic data has been originally formulated and solved in Koperski and Han (1995). Koperski and Han propose a method that has been obtained by adapting (Han & Fu, 1995) to spatial data mining and implemented in the module GeoAssociator of GeoMiner (Han, Koperski, & Stefanovic, 1997). When applied to geographic data, SPADA can be considered an upgrade of GeoAssociator to first-order logic. To the best of our knowledge, very few ILP applications to geographic data mining have been reported in the literature. GwiM (Popelinski, 1998) can solve several geographic data mining tasks, though no insight into the algorithmic issues has been provided. INGENS (Malerba et al., 2001) is an inductive geographic information system with learning capabilities which currently support classification tasks in topographic map interpretation.

6. An application to geo-referenced census data

In some works on spatial representation from the social scientist's perspective, socio-economic phenomena have been conceptualized as spatial objects in the sense of entities having both spatial location and spatially independent attribute characteristics (Martin, 1999). Population data is among the potentially spatial socio-economic data: it is usually geo-referenced with respect to areal spatial objects such as census zones, electoral constituencies, local government areas, or regular grid squares. In the UK, for instance, the geo-referencing areal units are ED (Enumeration District), Ward, District, and County. They form a hierarchy based on the inside relationship among locations. Thus the ED is the smallest unit for which census data is now published. Furthermore, the digital ED boundaries produced for the 1991 UK census enable the spatial representation of census data in the computer databases. Generally speaking, population censuses of the 1990s provided an added impetus to the application of GIS to socioeconomic uses. One of the most interesting topic areas for identifying potential users of such GIS applications is the public debate over Unitary Development Plans (UDP) in the UK. The district chosen for investigation is Stockport, one of the ten Metropolitan Districts of Greater Manchester, UK. It is divided into twenty-two wards for a total of 589 EDs. In particular, census data is extremely important for policy analysis and, once geo-referenced and conceptualized as spatial objects with numerical aspatial properties, supply a good test-bed to SPADA. The study of the Stockport district is expected to show the potential benefit of data mining methods and techniques to one or more potential users, in particular urban planners who are used to analyze jointly socioeconomic data and topographic maps. Indeed population and economic census data overlapped to geographic data can be the key indicator of 'optimum' locations for public services, thus supporting a good public policy. For instance, population census data such as car availability and topographic maps such as public transport networks are core ingredients in studies of accessibility of public services. This has been the practice in urban planning

environments for centuries. Nowadays it can be supported by powerful computer tools such as spatial data mining systems.

Data available for this application encompasses 1991 census data in text format (89 tables, each with 120 attributes on average) and map layers in vector format (including digital ED boundaries). This data has been loaded into an Oracle Spatial™ database, i.e. an object-relational DBMS extended with spatial data handling facilities. The ED code allows the joining of the two kinds of data and the generation of test data. Census data is available only at the ED level and is all numeric (in fact, integer-valued).

6.1. A study of commuting habits

Studying commuting habits of the population of a certain area is one of the main preliminary activities in transportation planning. Let us suppose that some decision-making process about the motor-way M63 is ongoing. Describing the area of Stockport served by the M63 (i.e. the wards of Brinnington, Cheadle, Edgeley, Heaton Mersey, South Reddish) may be of support to the planners. In this section we report the results obtained by applying SPADA to the task of discovering multi-level spatial association rules having EDs intersected by the M63 as reference objects (individuals of the concept `M63ED`) and all EDs in the area served by the M63 as task-relevant objects (individuals of the concept `M63AreaED`). The latter are to be characterized with respect to data about commuting. We call $\mathcal{B}_1$ the $\mathcal{AL}$-log knowledge base for this task.

6.1.1. The knowledge base. The structural component of $\mathcal{B}_1$ expresses a concept hierarchy for the Stockport ED layer which has been obtained by grouping EDs on the basis of the ward they belong to. In figure 4, both inclusion statements and membership assertions are graphically represented. E.g. the ED with code *fa01* is one of the EDs in the ward of Bredbury (`fa01:BredburyED`) and the EDs in the ward of Bredbury are EDs (`BredburyED` $\sqsubseteq$ `ED`). Further concept hierarchies could be derived by resorting to clustering algorithms.

The relational component of $\mathcal{B}_1$ contains both census and geographic data. In particular, the census attributes that we have selected for this experiment (see Table 1) refer to residents aged 16 and over, thus they have been normalized with respect to the total number of residents aged 16 and over (i.e. the attribute $s820001$). Each couple of consecutive cut points a and b has generated an interval of the kind $[a..b]$. Geographic data concerns

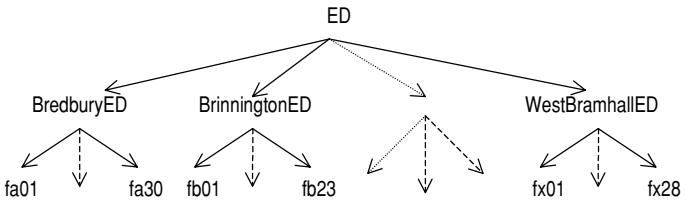

Figure 4. An is-a hierarchy for the Stockport ED layer.

Table 1. Census attributes in the experiment on Stockport commuting habits.

Attribute	Description	Cut points in the attribute domain
*s*820161	Persons who work out of the district of usual residence and drive to work	0.0, 6.25, 8.333, 12.973, 17.241, 19.048, 20.943, 23.529, 25.0, 25.926, 27.586, 29.032, 29.865, 31.25, 33.333, 34.375, 36.182, 38.235, 40.0, 42.105, 45.455, 46.667, 48.194, 50.0, 51.515, 52.632, 54.167, 56.0, 57.143, 58.333, 58.824, 60.0, 60.714, 61.538, 63.889, 65.217, 66.667, 67.742, 69.565, 71.429, 72.902, 100.0
*s*820213	Employees and self-employed who reside in households with 3 or more cars and drive to work	0.0, 2.222, 15.385, 28.0, 29.521, 31.034, 33.333, 35.068, 37.5, 38.095, 38.889, 41.043, 42.857, 48.387, 72.727
*s*820221	Employees and self-employed who reside in households with 3 or more cars and work out of the district of usual residence	0.0, 2.222, 4.762, 9.091, 10.345, 13.636, 18.182, 19.355, 21.131, 23.529, 25.0, 28.571

the relations of intersection between EDs and motorways (`intersect`) and adjacency between EDs (`adjacent_to`) that have been extracted from the database by means of spatial computation. On the contrary the relations of accessibility and closeness have been intensionally defined by means of spatial qualitative reasoning:

```
linked_to(X,Y)←intersect(X,m63),intersect(Y,m63)
    & X:M63AreaED, Y:M63AreaED
close_to(X,Y)←adjacent_to(X,Z), adjacent_to(Z,Y)
```

starting from the computed relations.

6.1.2. The declarative bias. To complete the problem statement, we specified a declarative bias both to constrain the search space and to filter out some uninteresting spatial association rules. In particular, we asked for rules containing only the predicates `linked_to` and `close_to`. This way, we ruled out all the computed spatial relations and focused on the ones derived by spatial qualitative reasoning. As to the inclusion of census attributes, we restricted our attention on some intervals for each attribute.

6.1.3. Analysis of results. SPADA has been run on $\mathcal{B}_1$ with thresholds $minsup^1 = 0.7$ and $minconf^1 = 0.9$ at the first level, and $minsup^2 = 0.5$ and $minconf^2 = 0.8$ at the second level. The whole discovery process has taken 24.61 sec on a PC Pentium III with 128 Mb RAM (21.86 sec for level 1, and 2.75 sec for level 2). It has returned 744 frequent patterns out of 3767 candidate patterns and 1862 strong rules out of 1862 generated rules. Table 2 reports quantitative results of this experiment level by level (l) and step by step (k) both for frequent pattern discovery (Phase I) and rule generation (Phase II). Each couple of slash-separated figures indicate the ratio between the number of frequent patterns (resp. strong rules) and the number of candidate patterns (resp. candidate rules). The figure in

Table 2. Quantitative results of the experiment on Stockport commuting habits.

		Phase I		Phase II		
l	k	No. patterns	Time	No. rules	Time	Total time
1	2	3/3	11.92	3/3	0.33	12.25
	3	6/108		6/6		
	4	12/210		30/30		
	5	27/710		36/36		
	6	45/1089		153/153		
2	2	3/3 (3)	9.94	3/3	2.42	12.36
	3	16/18 (120)		16/16		
	4	41/54 (572)		85/85		
	5	165/279 (2876)		304/304		
	6	426/569 (11203)		1226/1226		

round brackets is the number of candidate patterns generated in a previous experiment on the same data (Malerba & Lisi, 2001a). The fact that the number of candidate patterns at the second level of description granularity has lost one order of magnitude while the number of frequent patterns has remained unchanged ensures the correctness of our approach.

Some interesting patterns have been discovered. For instance, at level $l = 2$ of description granularity, the following candidate Q:

```
q(X)←close_to(X,Y), linked_to(X,Z), s820161(Z,[52.632..54.167]),
    & X:M63ED, Y:SouthReddishED, Z:CheadleED
```

has been generated after $k = 6$ refinement steps and evaluated with respect to $\mathcal{B}_1$. The pattern is a large one with 60% support and says that "*60% of EDs intersected by the M63 are close to a South Reddish ED and are linked via the M63 to a Cheadle ED where 52–54% residents aged 16 and over work out of the district of usual residence and drive to work*."

For the sake of clarity, we emphasize that Q has been actually obtained from the following pattern P at level $l = 1$:

```
q(X)←close_to(X,Y), linked_to(X,Z), s820161(Z,[52.632..54.167]),
    & X:M63ED, Y:M63AreaED, Z:M63AreaED
```

by means of the refinement rule $\langle \forall\, C \rangle$. It is straightforward to notice that P is coarser-grained than Q. Indeed it is supported by 90% of EDs intersected by the M63 and says that *"90% of EDs intersected by the M63 are close to an ED served by the M63 and are linked via M63 to another ED in the same area where 52–54% residents aged 16 and over work out of the district of usual residence and drive to work"*.

One of the strong rules that have been derived from Q is:

```
close_to(X,Y) & X:M63ED, Y:SouthReddishED⇝
    linked_to(X,Z), s820161(Z,[52.632..54.167]) & Z:CheadleED
```

with 60% support and 100% confidence. It says that "***IF*** *an ED intersected by the M63 is close to a South Reddish ED* ***THEN*** *WITH CONFIDENCE 100% it is linked via the M63 to a Cheadle ED where 52-54% residents aged 16 and over work out of the district of usual residence and drive to work.*"

Note that this rule would have not been discovered if we had limited the search to $\mathcal{L}^1$. Indeed the corresponding rule at the first level of description granularity does not exceed *minconf*[1]. Other examples of strong rules at the second level are:

```
close_to(X,Y), s820221(Y,[10.345..13.636]) & X:M63ED⇝
    linked_to(X,Z) & Z:BrinningtonED (60%, 86%)
```

"***IF*** *an ED intersected by the M63 is close to an ED where 10–13% residents aged 16 and over are employees and self-employed who reside in house-holds with 3 or more cars and work out of the district of usual residence* ***THEN*** *WITH CONFIDENCE 86% it is linked via the M63 to a Brinnington ED distinct from the previous one.*"

```
close_to(X,Y), s820221(Y,[19.355..21.131]) & X:M63ED⇝
    & Y:HeatonMerseyED (70%, 100%)
```

*"**IF** an ED intersected by the M63 is close to an ED where 19–21% residents aged 16 and over are employees and self-employed who reside in households with 3 or more cars and work out of the district of usual residence **THEN** WITH CONFIDENCE 100% the latter ED belongs to the ward of Heaton Mersey".*

A naïve interpretation of results in this experiment might lead the experts to state that, among Stockport EDs that are served by the motorway M63, those that are characterized by a high percentage of car commuters more urgently need some improvement of the road network.

6.2. A study of accessibility of health care services

In this section we present an application of SPADA to a case study of urban accessibility of health care services. Once again the area under analysis is Stockport. In particular, we study the access to Stepping Hill, where a big hospital is located. We have chosen this case study for two reasons. First, it relies heavily on the aforementioned joint analysis of census and geographic data. Second, it is interesting from the twofold perspective of transportation planning and health care planning. Indeed, accessibility of health care services is an issue for both planning activities (Gatrell & Senior, 1999). Many different definitions of accessibility and many ways to measure it can be found in the literature. In this work, we are concerned with urban accessibility, which refers to local (inner city) daily transport opportunities. A great effort has been made to define urban accessibility indices, which can be used to assess/compare transportation facilities within different regions of an urban area or between

urban regions (Bhat et al., 2000). In this study, we are interested in the urban accessibility "to" the Stepping Hill Hospital "from" the actual residence of people living within the area served by the hospital. Since (micro) data on the actual residence of each involved household are not available, we study the accessibility at the ED level. Moreover, our study does not aim to synthesize a new accessibility index, but to discover human interpretable patterns that can also contribute to directing resources for facility improvement in areas with poor transport accessibility.

In this section, we present the results obtained by applying SPADA to the task of mining association rules having EDs within a distance of 10 Km from the hospital (`RefED`) as reference objects, and EDs in the area of Stepping Hill (`SteppingHillED`) as task-relevant objects. This area has been defined as the one covering five EDs, three of which belong to the ward of Great Moor (`GreatMoorED`), one to the ward of Davenport (`DavenportED`) and the fifth one to the ward of Hazel Grove (`HazelGroveED`). The goal is to understand which reference EDs have access to the task-relevant EDs. We call $\mathcal{B}_2$ the $\mathcal{AL}$-log knowledge base for this experiment.

6.2.1. The knowledge base. The structural component of $\mathcal{B}_2$ encompasses two concept hierarchies, one for the ED layer (see figure 4 restricted to the Stepping Hill area) and the other one for the transport network layer (see figure 5). Both hierarchies have depth three and are straightforwardly mapped into three granularity levels.

The relational component of $\mathcal{B}_2$ contains facts expressing the existence of topological relations between EDs and objects of the map layers of roads, railways and bus priority lines, e.g.

```
external_touches_to(ed_03bsfq29, bus_priority_line_1).
crosses(ed_03bsfc13, road_12245).
along(ed_03bsfg25, rail_2453).
```

The first fact states that the relation `external_touches_to` holds between the ED with identifier `ed_03bsfq29` and the bus priority line identified by `bus_priority_line_1`. The other two facts are to be interpreted analogously. Besides these geographic data, census data is used to define the accessibility of the Stepping Hill area. Indeed, even though some roads connect a reference ED X with a task-relevant ED Y, people living in X might have problems reaching Y because they do not drive. This means that sociological data

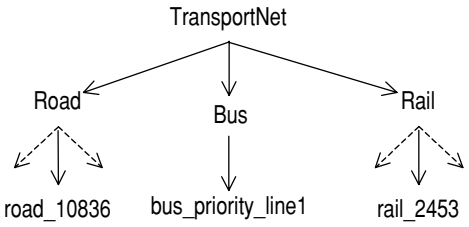

Figure 5. An is-a hierarchy for the Stockport transport network layer.

available in the census data tables can be profitably used to give an improved definition of accessibility. We selected four attributes on the percentage of households with zero, one, two, and three or more cars, we discretized them with RUDE and generated the following four binary predicates for SPADA: `no_car, one_car, two_cars, three_more_cars`. The first argument of the predicate refers to an ED, while the second argument is an interval returned by RUDE.

Despite the availability of geographic and census data, the resulting knowledge base does not provide a satisfactory definition of accessibility yet. Indeed, we are interested in relationships between EDs, such as those stating that two EDs are 'connected' by the same bus priority line or the same road or the same railway. To solve this problem we added rules such as:

```
crossed_by_bus_line(X)
    ←external_touches_to(X,bus_priority_line_1)
connected_by_bus_line(X,Y)
    ←crossed_by_bus_line(X), crossed_by_bus_line(Y)

crossed_by_road(X,Z)←crosses(X,Z) & Z:Road
connected_by_road(X,Y)
    ←crossed_by_road(X,Z), crossed_by_road(Y,Z)

crossed_by_rail(X,Z)←along(X,Z) & Z:Rail
connected_by_rail(X,Y)
    ←crossed_by_rail(X,Z), crossed_by_rail(Y,Z)
```

to Π. They supply an intensional definition of connectivity limited to the cases of direct accessibility of an ED from another ED by means of only one road or railway or bus line. To express more complex cases of accessibility, we added intensional definitions of `can_reach_by_road, can_reach_by_rail, can_reach_by_bus, can_reach_by_rail_bus`, and `can_reach_by_public_transport` on the accessibility by means of public transport. Also we provided the definition of `can_reach_only_by_road` to account for accessibility by means of roads alone.

6.2.2. The declarative bias. To complete the problem statement we specified a declarative bias both to constrain the search space and to filter out some uninteresting spatial association rules. In particular, we asked for rules containing only the following predicates: `can_reach_by_public_transport, can_reach_only_by_road, no_car, one_car, two_cars`, and `three_more_cars`. In this way, we ruled out all spatial relations—both the ones computed and the ones derived by spatial qualitative reasoning—that helped defining the most interesting ones, namely the accessibility by public transport and the accessibility only by roads. Moreover, the specification of the following filter:

```
pattern_constraint([no_car(_,_), one_car(_,_), two_cars(_,_),
  three_more_cars(_,_)], 1).
```

prevents the generation of association rules with purely spatial patterns, that is, patterns showing only spatial relations between spatial objects. Purely spatial patterns are indeed of no interest to the expert in transport planning, since it is very likely that they convey no additional information to what he/she already knows.

6.2.3. Analysis of results. After some tuning of the minimum thresholds of support and confidence for each granularity level, we decided to run the system with the following parameter values: $minsup^1 = 0.2$ and $minconf^1 = 0.5$ for the level 1, $minsup^2 = 0.1$ and $minconf^2 = 0.4$ for the level 2, and $minsup^3 = 0.1$ and $minconf^3 = 0.3$ for the level 3. Despite the declarative bias, SPADA generated 944 rules in 88 secs from a set of 39,830 extracted or inferred facts. More precisely, the system generated 28 rules in 38 secs at granularity level 1, 215 rules in 17 secs at level 2, and 701 rules in 33 secs at level 3. Two of the rules returned by SPADA at the first level are the following:

```
can_reach_only_by_road(A,B) & A:RefED, B:SteppingHillED
  ⇝no_car(A,[0.228..0.653]) (38.15%, 56.31%)
can_reach_by_public_transport(A,B) & A:RefED, B:SteppingHillED
  ⇝no_car(A,[0.228..0.653]) (21.71%, 61.11%)
```

The spatial pattern of the first rule occurs in fifty-eight distinct EDs. This means that from fifty-eight distinct EDs within a distance of 10 Km from Stepping Hill Hospital, it is possible to reach the hospital only by road and the percentage of households with no car is quite high (between 22.8% and 65.3%). Moreover, if from an ED `A` around Stepping Hill Hospital it is possible to reach one of the five task-relevant EDs only by road, then the confidence that `A` has a high percentage of households with no car is 56.31%. The spatial pattern of the second rule occurs in thirty-three distinct EDs. This means that from thirty-three reference EDs whose percentage of households with no car is quite high it is possible to reach the area of the Stepping Hill Hospital by public transport. The confidence in the second rule is a little higher than the first association rule. At granularity level 2, SPADA specializes the concept `SteppingHillED` considered at level 1. The first association rule above is specialized as follows:

```
can_reach_only_by_road(A,B) & A:RefED, B:GreatMoorED
  ⇝no_car(A,[0.228..0.653]) (38.15%, 56.31%)
can_reach_only_by_road(A,B) & A:RefED, B:DavenportED
  ⇝no_car(A,[0.228..0.653]) (21.71%, 50.76%)
can_reach_only_by_road(A,B) & A:RefED, B:HazelGroveED
  ⇝no_car(A,[0.228..0.653]) (21.71%, 50.76%)
```

As expected, the support of some rules has decreased. However, since both support and confidence are greater than the corresponding user-defined thresholds, all the three rules are output by SPADA. Similar considerations apply to granularity level 3, where specific task-relevant EDs are reported.

Association rules found by SPADA in this application can be of interest to urban planners, since they relate data on the transport network with data on sociological factors.

However, this study has three main limitations due to the nature of available data. First, we considered 1991 Census data, which are now obsolete. Second, the crossing of a railway does not necessarily mean that there is a station in an ED. Similar considerations can be made for bus priority lines and roads. Third, digital maps made available by the Ordnance Survey are devised for cartographic reproduction purposes and not for data analysis. Hence, a road may appear to be 'blocked' in the digital map, because it runs under a bridge.

7. Conclusions and future work

Recent extensions of ILP witness a growing interest in description logics and KDD applications. In this paper we have shown that $\mathcal{AL}$-log is suitable for inducing multi-level association rules from multiple relations. We have proposed an $\mathcal{AL}$-log framework and a novel ILP setting to work within it. The trade-off between efficiency and expressive power is well known in the machine learning community. So we have shown that efficient algorithms can be designed for coping with descriptive data mining tasks where multiple levels of description granularity are required. In particular we have proposed an implementation of the refinement operator $\rho_{\mathcal{O}}$ which is based on a graph of backward pointers. The current version of SPADA admits only $\mathcal{ALC}$ primitive concepts in the structural knowledge. Roles and complex concepts have been disregarded. Yet we are confident that results obtained on this subset of $\mathcal{AL}$-log are still valid for full $\mathcal{AL}$-log.

For the future we plan to investigate the properties of $\rho_{\mathcal{O}}$. Also we intend to design a constraint-based language bias more suitable for searching $\mathcal{AL}$-log pattern spaces. The method can be further improved by investigating proper measures of interestingness (e.g. pruning conditions for the rule space) and the issue of robustness (e.g. techniques for handling approximated spatial relations). Other issues, such as visual representation of knowledge at multiple levels, should also be studied in depth. Furthermore, with the advent of data warehousing and OLAP technologies, arranging data at *multiple levels of abstraction* has become a common practice and boosted the research on multi-dimensional databases (Ferri et al., 2000; Krogel & Wrobel, 2001). It would be interesting to extend our method in this direction.

On the application side, we plan to carry on experiments in geographic data mining and ask domain experts for evaluation of results. Furthermore we would like to test SPADA as a tool for the exploration of maps. As such it could be integrated into INGENS and used in stages preliminary and/or complementary to map interpretation. We intend also to tackle new applications among which ontology learning seems quite promising.

Notes

1. The symbol $\subset$ is to be read as 'sub-concept of'.
2. Reserved atoms (e.g., $is\_a/2$ for concept assertions and $\neq$ for object-identity) and program-defined mechanisms implement our $\mathcal{AL}$-log framework.
3. These are directives for the candidate generation phase. Here the signs $+$ and $-$ stand for input and output variable, respectively. See Dehaspe and Toivonen (1999) for further details.

References

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In J. Bocca, M. Jarke, & C. Zaniolo (Eds.), *Proceedings of 20th International Conference on Very Large Data Bases* (pp. 487–499). Morgan Kaufmann.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (Eds.). (2003). *The description logic handbook: Theory, implementation and applications*. Cambridge University Press.

Badea, L., & Nienhuys-Cheng, S.-W. (2000). A refinement operator for description logics. In J. Cussens & A. Frisch (Eds.), *Inductive logic programming*, vol. 1866 of Lecture Notes in Artificial Intelligence (pp. 40–59) Springer-Verlag.

Bhat, C., Handy, S., Kockelman, K., Mahmassani, H., Chen, Q., & Weston, L. (2000). Urban accessibility index: Literature review. Technical Report TX-01/7-4938-1, Texas Dept. of Transportation, University of Texas, Austin.

Blockeel, H., De Raedt, L., Jacobs, N., & Demoen, B. (1999). Scaling up inductive logic programming by learning from interpretations. *Data Mining and Knowledge Discovery, 3*, 59–93.

Borgida, A. (1996). On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence, 82:1/2*, 353–367.

Buntine, W. (1988). Generalized subsumption and its application to induction and redundancy. *Artificial Intelligence*, *36:2*, 149–176.

Ceri, S., Gottlob, G., & Tanca, L. (1990). *logic programming and databases*. Springer.

Cohen, W., Borgida, A., & Hirsh, H. (1992). Computing least common subsumers in description logics. In *Proc. of the 10th National Conf. on Artificial Intelligence* (pp. 754–760). The AAAI Press/The MIT Press.

Cohen, W., & Hirsh, H. (1994). Learning the CLASSIC description logic: Thoretical and experimental results. In *Proc. of the 4th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'94)* (pp. 121–133). Morgan Kaufmann.

De Raedt, L., & Dehaspe, L. (1997). Clausal discovery. *Machine Learning, 26:2/3*, 99–146.

De Raedt, L., & Džeroski, S. (1994), First order jk-clausal theories are PAC-learnable. *Artificial Intelligence, 70*, 375–392.

Dehaspe, L., & Toivonen, H. (1999). Discovery of frequent DATALOG patterns. *Data Mining and Knowledge Discovery, 3*, 7–36.

Donini, F., Lenzerini, M., Nardi, D., & Schaerf, A. (1998). $\mathcal{AL}$-log: Integrating datalog and description logics. *Journal of Intelligent Information Systems, 10:3*, 227–252.

Džeroski, S. (1996). Inductive logic programming and knowledge discovery in databases. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (Eds.), *Advances in knowledge discovery and data mining*. (pp. 117–152). AAAI Press/The MIT Press.

Džeroski, S. (2001). Relational data mining applications: An overview. In S. Džeroski & N. Lavrač (Eds.), *Relational data mining* (pp. 339–364). Springer.

Egenhofer, M., & Herring, J. (1994). Categorizing binary topological relations between regions, lines, and points in geographic databases. In M. Egenhofer, D. Mark, & J. Herring (Eds.), *The 9-intersection: Formalism and its use for natural-language spatial predicates* (pp. 183–271). Technical Report 94-1, U.S. NCGIA.

Ester, M., Frommelt, A., Kriegel, H.-P., & Sander, J. (2000). Spatial data mining: Database primitives, algorithms and efficient DBMS support. *Data Mining and Knowledge Discovery*, *4:2/3*, 193–216.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery: An overview. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (Eds.), *Advances in knowledge discovery and data mining*, (pp. 1–34). AAAI Press/The MIT Press.

Ferri, F., Pourabbas, E., Rafanelli, M., & Ricci, F. (2000). Extending geographic databases for a query language to support queries involving statistical data. In G (Ed.), *Proceedings of the 12th Int. Conf. on Scientific and Statistical Database Management*.

Flach, P., & Džeroski, S. (2001). Editorial: Inductive logic programming is coming of age. *Machine Learning*, *44:3*, 207–209.

Gatrell, A., & Senior, M. (1999). Health and health care applications. In P. Longley, M. Goodchild, D. Maguire, & D. Rhind (Eds.), *Geographical information systems, vol. 2, Principles and technical issues*, 2nd edn. vol. 2, (pp. 925–938). John Wiley & Sons.

Güting, R. (1994). An introduction to spatial database systems. *VLDB Journal*, *3:4*, 357–399.

Han, J., & Fu, Y. (1995). Discovery of multiple-level association rules from large databases. In U. Dayal, P. Gray, & S. Nishio (Eds.), *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases*, Sept. 11–15, 1995, (pp. 420–431). Zurich, Switzerland. Morgan Kaufmann.

Han, J., & Fu, Y. (1999). Mining multiple-level association rules in large databases. *IEEE Transactions on Knowledge and Data Engineering*, *11:5*.

Han, J., Koperski, K., & Stefanovic, N. (1997). GeoMiner: A system prototype for spatial data mining. In J. Peckham (Ed.), *Proceedings ACM SIGMOD International Conference on Management of Data* (pp. 553–556). ACM Press.

Helft, N. (1987). Inductive generalization: A logical framework. In I. Bratko, & N. Lavrač (Eds.), *Progress in Machine Learning—Proceedings of EWSL87: 2nd European Working Session on Learning* (pp. 149–157). Wilmslow, U.K.: Sigma Press.

Kietz, J.-U., & Morik, K. (1994). A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, *14:1*, 193–217.

Koperski, K., & Han, J. (1995). Discovery of spatial association rules in geographic information databases. In M. Egenhofer and J. Herring (Eds.), *Advances in spatial databases*, vol. 951 of Lecture Notes in Computer Science (pp. 47–66). Springer.

Krogel, M.-A., & Wrobel, S. (2001). Transformation-based learning using multirelational aggregation. In C. Rouveirol & M. Sebag (Eds.), *Inductive logic programming*, vol. 2157 of Lecture Notes in Artificial Intelligence (pp. 142–155). Springer.

Levy, A., & Rousset, M.-C. (1998). Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, *104*, 165–209.

Lisi, F.A., Ferilli, S., & Fanizzi, N. (2002). Object identity as search bias for pattern spaces. In F. van Harmelen (Ed.), *ECAI 2002. Proceedings of the 15th European Conference on Artificial Intelligence* (pp. 375–379). Amsterdam: IOS Press.

Ludl, M.-C., & Widmer, G. (2000). Relative unsupervised discretization for association rule mining. In D. Zighed, H. Komorowski, & J. Zytkow (Eds.), *Principles of data mining and knowledge discovery*, vol. 1910 of Lecture Notes in Artificial Intelligence (pp. 148–158). Springer.

Malerba, D., Esposito, F., Lanza, A., & Lisi, F. A. (2001). Machine learning for information extraction from topographic maps. In H. Miller, & J. Han (Eds.), *Geographic data mining and knowledge discovery* (pp. 291–314). Taylor and Francis.

Malerba, D., & Lisi, F. A. (2001a). Discovering associations between spatial objects: An ILP application. In C. Rouveirol & M. Sebag (Eds.), *Inductive logic programming*, vol. 2157 of Lecture Notes in Artificial Intelligence (pp. 156–163). Springer.

Malerba, D., & Lisi, F. A. (2001b). An ILP method for spatial association rule mining. In A. Knobbe, & D. van der Wallen (Eds.), *Notes of the ECML/PKDD 2001 workshop on multi-relational data mining* (pp. 18–29).

Mannila, H., & Toivonen, H. (1997). Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery, 1:3*, 241–258.

Martin, D. (1999). Spatial representation: The social scientist's perspective. In P. Longley, M. Goodchild, D. Maguire, & D. Rhind (Eds.), *Geographical information systems,* vol. 1, *principles and technical issues*, 2nd edn., vol. 1. (pp. 71–80). John Wiley & Sons.

Nienhuys-Cheng, S., & de Wolf, R. (1997). *Foundations of inductive logic programming*, vol. 1228 of Lecture Notes in Artificial Intelligence. Springer.

Nijssen, S., & Kok, J. (2001). Faster association rules for multiple relations. In B. Nebel (Ed.), *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* (pp. 891–896). Morgan Kaufmann.

Plotkin, G. (1970). A note on inductive generalization. *Machine Intelligence*, *5*, 153–163.

Popelinski, L. (1998). Knowledge discovery in spatial data by means of ILP. In J. Zytkow, & M. Quafalou (Eds.), *Principles of data mining and knowledge discovery, second European symposium, PKDD '98*, vol. 1510 of Lecture Notes in Artificial Intelligence (pp. 185–193). Springer.

Reiter, R. (1980). Equality and domain closure in first order databases. *Journal of ACM*, *27*, 235–249.

Rouveirol, C., & Ventos, V. (2000). Towards learning in CARIN-$\mathcal{ALN}$. In J. Cussens & A. Frisch (Eds.), *Inductive logic programming*, vol. 1866 of Lecture Notes in Artificial Intelligence (pp. 191–208). Springer.

Schmidt-Schauss, M., & Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial Intelligence, 48:1*, 1–26.

Semeraro, G., Esposito, F., Malerba, D., Fanizzi, N., & Ferilli, S. (1998). A logic framework for the incremental inductive synthesis of datalog theories. In N. Fuchs (Ed.), *Proceedings of 7th International Workshop on Logic Program Synthesis and Transformation*, vol. 1463 of Lecture Notes in Computer Science (pp. 300–321). Springer.

Srikant, R., & Agrawal, R. (1995). Mining generalized association rules. In U. Dayal, P. Gray, & S. Nishio (Eds.), *Proceedings of 21th International Conference on Very Large Data Bases* (pp. 407–419). Morgan Kaufmann.

Weber, I. (1999). A declarative language bias for levelwise search of first-order regularities. In Z. Ras, & A. Skowron (Eds.), *Foundations of intelligent systems*, vol. 1609 of Lecture Notes in Artificial Intelligence (pp. 253–261). Springer-Verlag.

Received May 10, 2002
Revised April 2, 2003
Accepted November 4, 2003
Final manuscript December 22, 2003