Efficient Discovery of Multiple-level Patterns*

Francesca A. Lisi and Donato Malerba

Dipartimento di Informatica, Università degli Studi di Bari
Via Orabona 4, I-70125 Bari, Italy
{lisi, malerba}@di.uniba.it

Abstract. In the context of frequent pattern discovery, the availability of concept hierarchies over objects of interest requires the development of ad-hoc algorithms for dealing with multiple levels of description granularity. We present a novel framework that allows a unified approach to both relational and structural features of data. Patterns are intended as unary conjunctive queries and ordered according to the relation of query subsumption. A refinement operator for searching these pattern spaces is defined and efficiently implemented in the candidate generation phase of SPADA, a system for mining multiple-level association rules from spatial data. Experimental results show a remarkable improvement of the overall performance of the system.

1 Introduction

The design of algorithms for frequent pattern discovery has turned out to be a popular topic in data mining. The blueprint for most algorithms proposed in the literature is the levelwise method that is based on a breadth-first search in the lattice spanned by a generality order between patterns [15]. 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. Frequent patterns are commonly post-processed into rules that exceed given threshold values. In the case of association rules [1], the measures of support and confidence offer a natural way of pruning weak rules.

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 $\mathcal{H}_1$=$\{Milk \sqsubset Food$, $Bread \sqsubset Food$, $LowFatMilk \sqsubset Milk$, $ChocoMilk \sqsubset Milk$, $WhiteBread \sqsubset Bread$, $WheatBread \sqsubset Bread\}$ 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

* This paper is in partial fulfillment of the research objectives set by the IST project SPIN! (Spatial Mining for Data of Public Interest) funded by the European Union (http://www.ccg.leeds.ac.uk/spin/).

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 $\mathcal{H}_2 = \{Outerwear \sqsubset Clothes, Shirts \sqsubset Clothes, Jackets \sqsubset Outerwear, SkiPants \sqsubset Outerwear\}$ and $\mathcal{H}_3 = \{Shoes \sqsubset Footwear, HikingBoots \sqsubset Footwear\}$ 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 [9] a top-down progressive deepening method for mining *multi-level association rules* has been developed by extending the Apriori algorithm [1] for mining single-level association rules. A major difference between this and others' proposals, e.g. [20], 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. An adaptation of this method to spatial association rule mining has been presented in [11].

Recent extensions of the levelwise method witness a growing interest in more expressive languages for representing data and patterns. For instance, the system WARMR supports the discovery of frequent Datalog patterns [4]. But although it has been presented as a system able to use is-a hierarchies, WARMR is not a system for mining multiple-level association rules because it lacks of mechanisms for dealing properly with structural knowledge. In this paper, we present a framework for frequent pattern discovery at multiple levels of description granularity. In particular, we resort to $\mathcal{AL}$-log [6] that allows a unified approach to both the relational and structural features of data. Patterns are intended as unary conjunctive queries and ordered according to the relation of query subsumption. This generality order is monotonic with respect to the evaluation function in frequent pattern discovery, i.e. support, and yields to a downward refinement operator that performs the search for patterns both at the same granularity level and at finer granularity levels. We propose a graph-based implementation of this refinement operator. It exploits backward pointers to previous successful search stages in order to speed up the discovery process. This new approach to candidate generation is implemented in the latest version of SPADA, a system for spatial association rule mining, and evaluated on spatial data of an Italian province, thus updating results reported in [13].

The paper is organized as follows. Section 2 introduces the task of discovering frequent multiple-level patterns. In Section 3, syntax and semantics of $\mathcal{AL}$-log is briefly reported and adapted to the representation of data and patterns in the context of frequent pattern discovery. Section 4 is devoted to the presentation of the generality order for organizing spaces of $\mathcal{AL}$-log patterns. An efficient implementation of a $\mathcal{AL}$-log refinement operator is discussed in Section 5. Concluding remarks are given in Section 6.

2 The Mining Task

The distinguishing feature of the class of data mining problems of interest is the availability of some taxonomic information $\mathcal{T} = \{\mathcal{H}_k\}_{1 \leq k \leq m}$ besides the database R to be mined. It is noteworthy that each concept hierarchy $\mathcal{H}_k$ in $\mathcal{T}$ can arrange its concepts $\{C_k^h\}_{1 \leq h \leq n_k}$ according to its own range of concept levels. Furthermore, data are 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 problem $\mathcal{P}$ at hand. Given $\Psi = \{1, \ldots, maxlevel(\mathcal{P})\}$ the set of levels of description granularity in $\mathcal{P}$, a *granularity assignment* is a relation ψ over $\mathcal{H}_k \times \Psi$. 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 granularity levels. For instance, the concepts C_1^2 and C_1^3 in $\mathcal{H}_1 = \{C_1^2 \sqsubset C_1^1, C_1^3 \sqsubset C_1^1\}$ have been assigned to both $\mathcal{T}[2]$ and $\mathcal{T}[3]$.

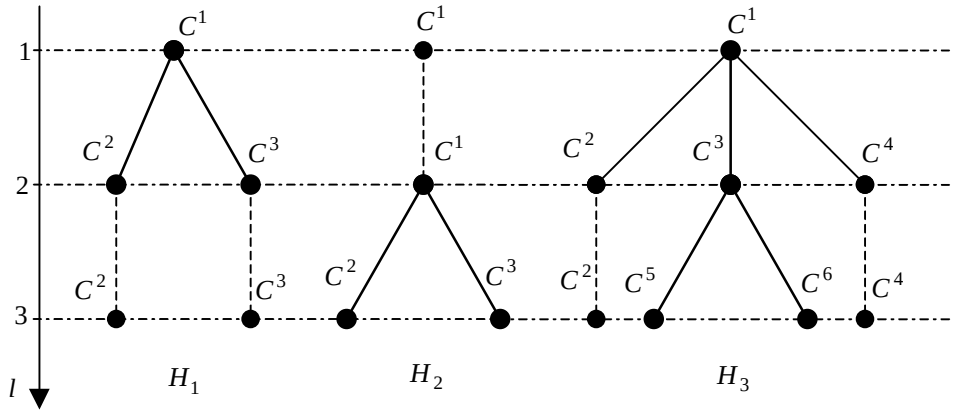

Fig. 1. Assigning description granularity levels to concepts

A pattern is an expression in some language describing a subset of data or a model applicable to that subset [7]. Given a taxonomy $\mathcal{T}$, we denote by $\mathcal{L}[l]$ the language of patterns involving concepts in $\mathcal{T}[l]$. A pattern $P' \in \mathcal{L}[l']$, $l' < l$ (resp. $l' > l$), is an *ancestor* (resp. *descendant*) of $P \in \mathcal{L}[l]$ iff it can be obtained from P by replacing each concept C that occurs in P with a concept $D \in \mathcal{T}[l']$ such that $C \sqsubseteq D$ (resp. $D \sqsubseteq C$). 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.

The problem $\mathcal{P}$ of mining multi-level association rules is formally defined as:

Given

- a database R,
- a taxonomy $\mathcal{T}$,
- a set Ψ of description granularity levels,
- two thresholds, $minsup[l]$ and $minconf[l]$, for each level l in Ψ,

find strong multi-level association rules.

Thus patterns, then association rules, are evaluated by taking their own level of description granularity into account. In particular, candidate patterns that fulfill the following requirements are retained:

Definition 1. *A pattern $P \in \mathcal{L}[l]$ with support s is* frequent *if (i) $s \geq minsup[l]$ and (ii) all ancestors of P w.r.t. $\mathcal{T}$ are frequent.*

Frequent patterns are post-processed into association rules.

Definition 2. *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 called the* support *and the* confidence *of the rule.*

Definition 3. *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.*

Support and confidence are computed by means of the evaluation function σ that varies according to the chosen representation language.

3 Representing Data and Patterns in $\mathcal{AL}$-log

$\mathcal{AL}$-log is a hybrid knowledge representation language which integrates the description logic $\mathcal{ALC}$ [19] and the deductive database language Datalog [3]. A fragment of it is actually used as a language for representing data and patterns in our context. Thus we limit the presentation of $\mathcal{AL}$-log to the features of interest to this work.

3.1 Syntax

The system $\mathcal{AL}$-log embodies two subsystems, called structural and relational. The former allows one to express knowledge about concepts, roles and individuals by resorting to $\mathcal{ALC}$. The latter provides the user with a suitable extension of Datalog in order to express relational knowledge.

The *structural* subsystem of $\mathcal{AL}$-log is itself a two-component system. As for the definition of the *intensional* part of an $\mathcal{ALC}$-knowledge base, it consists of concept hierarchies spanned by is-a relations between concepts. Syntactically, they are 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 definition of the *extensional* part of an $\mathcal{ALC}$-knowledge base, we are concerned with 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* of the form $o : C$ (read "o is a member of C") where $o \in \mathcal{O}$, and C is a concept. Intuitively, the assertion says that o is an instance of C. Given a set $\mathcal{T}$ of inclusion statements and a set $\mathcal{M}$ of membership assertions, the pair $\Sigma = \langle \mathcal{T}, \mathcal{M} \rangle$ denotes an *$\mathcal{ALC}$-knowledge base* Σ.

The *relational* subsystem of $\mathcal{AL}$-log allows one to define 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 4. *A* constrained Datalog clause *has 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. Furthermore α_0 and $\exists \alpha_1, \ldots, \alpha_m \& \gamma_1, \ldots, \gamma_n$ are called head *and* body *of the clause, respectively. A* constrained Datalog program *Π is a set of constrained Datalog clauses.*

Given an $\mathcal{ALC}$-knowledge base Σ and a constrained Datalog program Π, an *$\mathcal{AL}$-log knowledge base* $\mathcal{B}$ is the pair $\mathcal{B} = \langle \Sigma, \Pi \rangle$ that satisfies 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 $\mathcal{O}$. 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.

These properties allow us to extend the notion of ground substitution to constrained Datalog clauses. We call *$\mathcal{O}$-substitution* a mapping σ from the set V of variables to the set $\mathcal{O}$ of constants (individuals) in an $\mathcal{AL}$-log knowledge base.

Queries to $\mathcal{AL}$-log knowledge bases are special cases of Definition 4. In particular, unary conjunctive queries whose answer set contains individuals of an $\mathcal{ALC}$ concept $\hat{C}$ of interest (*key constraint*) deserve special attention.

Definition 5. *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 the body of Q are the* existential variables.

From now on we denote by $key(Q)$ the key constraint, by $head(Q)$ the head, and by $body(Q)$ the body of an $\mathcal{O}$-query Q.

In our framework, data and patterns are represented as an $\mathcal{AL}$-log knowledge base and $\mathcal{O}$-queries respectively. Patterns are generated starting from a set $\mathcal{A}$ of Datalog atoms, a key constraint $\hat{\gamma}$, and an additional set Γ of constraints. In particular by restricting Γ to constraints derived from $\mathcal{T}[l]$ (thus denoted $\Gamma[l]$) we define the language $\mathcal{L}[l]$. Furthermore, patterns must satisfy the properties of linkedness [10] and safety [3]. These conditions of well-formedness guarantee the correct evaluation of patterns.

Example 1. Let $\mathcal{A}$=$\{intersects(\_,\_)$, $adjacent\_to(\_,\_)\}$ and $\hat{\gamma}$ be the key constraint built on the concept $LargeTown$. Suppose that the concept hierarchies $\mathcal{H}_1$=$\{Motorway \sqsubseteq Road$, $MainTrunkRoad \sqsubseteq Road$, $RegionalRoad \sqsubseteq Road\}$ and $\mathcal{H}_2$ =$\{Sea \sqsubseteq Water$, $River \sqsubseteq Water$, $Lake \sqsubseteq Water\}$ are assigned. Suppose also that we are interested in descriptions at two different granularity levels. Thus the taxonomy $\mathcal{T}$ consists of the two layers $\mathcal{T}[1]$=$\{Road$, $Water\}$ and $\mathcal{T}[2]$=$\{Motorway$, $MainTrunkRoad$, $RegionalRoad$, Sea, $River$, $Lake\}$ from which the sets $\Gamma[1]$ and $\Gamma[2]$ of constraints are derived. 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.

Candidate evaluation is based on the computation of the answer set of a query Q w.r.t. a $\mathcal{AL}$-log knowledge base $\mathcal{B}$. Before discussing issues of query answering, we need to introduce the semantics of $\mathcal{AL}$-log.

3.2 Semantics

In $\mathcal{ALC}$-log knowledge bases 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 $\Delta^{\mathcal{I}}$ and each role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that some equations are satisfied [6]. 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 $a \neq b$. Such restriction ensures that different individuals denote different objects in the domain of interest (see *unique names* assumption [17]). Interpretations that are compliant with this restriction are called $\mathcal{O}$-interpretations.

Definition 6. *An $\mathcal{O}$-interpretation $\mathcal{I}$* satisfies*:*

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

As for the semantics of an $\mathcal{AL}$-log knowledge base $\mathcal{B} = \langle \Sigma, \Pi \rangle$, it is noteworthy that the interaction between the structural and the relational part of $\mathcal{B}$ is determined by the constraints specified in the clauses of Π. We call Π_D the set of Datalog clauses obtained from the clauses of Π by deleting their constraints. We define an interpretation $\mathcal{J}$ for $\mathcal{B}$ as the union of an $\mathcal{O}$-interpretation $\mathcal{I}_{\mathcal{O}}$ for Σ and an Herbrand interpretation $\mathcal{I}_{\mathcal{H}}$ for Π_D.[1]

Definition 7. *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 $\mathcal{O}$-queries. The reader is referred to [6] for an insight into query answering mechanisms in $\mathcal{AL}$-log.

Definition 8. *An $\mathcal{AL}$-log knowledge base $\mathcal{B}$* logically implies *an existentially quantified conjunctive formula $F = \exists \alpha_1, \ldots, \alpha_m \& \gamma_1, \ldots, \gamma_n$ if there exists a ground instance $F' = \alpha'_1, \ldots, \alpha'_m \& \gamma'_1, \ldots, \gamma'_n$ of F such that $\mathcal{B} \models F'$.*

Definition 9. *Let $\mathcal{B}$ be a $\mathcal{AL}$-log knowledge base. An* answer *to the $\mathcal{O}$-query Q is a ground $\mathcal{O}$-substitution θ for the distinguished variable. The answer θ is* correct *w.r.t. $\mathcal{B}$ if $\mathcal{B} \models body(Q)\theta$. The set answerset$(Q, \mathcal{B})$ contains all the correct answers to Q w.r.t. $\mathcal{B}$.*

During *candidate evaluation* we are concerned with computing the support of a pattern. This is defined as the ratio between the number of individuals in $\hat{C}$ that satisfy the candidate pattern and the total number of individuals in $\hat{C}$.

[1] We assume the reader to be familiar with the Herbrand model-theoretic semantics of Datalog [3].

Definition 10. *Let $\mathcal{B}$ be a $\mathcal{AL}$-log knowledge base, $P \in \mathcal{L}[l]$. The* support *of P with respect to $\mathcal{B}$ is defined:*

$$\sigma(P, \mathcal{B}) = \frac{\mid answerset(P, \mathcal{B}) \mid}{\mid answerset(\widehat{P}, \mathcal{B}) \mid}$$

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

Further details of the evaluation function and also its relevance to candidate generation are given in the next section.

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 [15]. The choice of the generalization model for a space of $\mathcal{AL}$-log patterns affects both its algebraic structure and, as a consequence, the definition of refinement operators to work on it. 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 the answer set of Q_1 is contained in the answer set of Q_2. This characterization turns out to be profitable in candidate evaluation.

Subsumption is the most important form of reasoning in DLs [5]. In contrast to subsumption of concepts, subsumption of conjunctive queries has not been extensively studied. In [12] it is treated as a special case of existential entailment for $\mathcal{ALCNR}$. We propose to extend generalized subsumption [2] to $\mathcal{O}$-queries. The approach is inspired by the hypothesis ordering proposed in [18] for learning from interpretations in CARIN-$\mathcal{ALN}$. The adaptation is correct since $\mathcal{AL}$-log is less powerful than languages in the CARIN family.

Definition 11. *Let Q_1 and Q_2 be two $\mathcal{O}$-queries to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$. We say that Q_1* subsumes *Q_2 iff (i) $key(Q_1) = key(Q_2)$, (ii) $head(Q_1) = head(Q_2)$ and (iii) $body(Q_2) \models_{\mathcal{B}} body(Q_1)$. Furthermore, Q_1 is* equivalent *to Q_2 if Q_1 subsumes Q_2 and viceversa.*

The completeness and soundness of both existential entailment (proved in [12]) and forward chaining yields a complete and sound inference mechanism for checking subsumption between two $\mathcal{O}$-queries. Note that conditions (i) and (ii) of Definition 11 are guaranteed in the case of $\mathcal{O}$-queries belonging to the same language $\mathcal{L}$ whereas condition (iii) poses an existential entailment problem.

Subsumption can be adopted to induce a generality order over pattern spaces.

Definition 12. *Let $P, Q \in \mathcal{L}$. We say that P is* more general than *Q, $P \succeq Q$, iff P subsumes Q. Furthermore, $P \succ Q$ iff $P \succeq Q$ but $Q \succeq P$ does not hold. Finally, $P \sim Q$ iff $P \succeq Q$ and $Q \succeq P$.*

It can be easily proven that $\succeq$ is a quasi-order. Quasi-ordered sets can be searched by refinement operators [16].

Definition 13. *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.* $\delta(P) \subseteq \{Q \in \mathcal{L} \mid Q \succeq P\}$*).*

Furthermore it can be proven that $\succeq$ is monotonic with respect to the evaluation function σ.

Proposition 1. *Let* $\mathcal{B}$ *be an* $\mathcal{AL}$*-log knowledge base and* P *and* Q *two patterns in* $\mathcal{L}$*. If* $P \succeq Q$ *then* $\sigma(P, \mathcal{B}) \geq \sigma(Q, \mathcal{B})$*.*

From this proposition 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 14. *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 atom* $\alpha \in \mathcal{A}$ *and related constraints* $\gamma \in \Gamma[l]$ *to* $body(P)$*.*
$\langle \forall C \rangle$ *Replace each constraint* $\gamma_j = X : C$ *in* $body(P)$ *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 conditions (i)-(ii) of Definition 11 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 6. So condition (iii) 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 $\mathcal{L}[l]$. Under the assumption that $minsup[l] \leq minsup[l-1]$, $l > 1$, two pruning conditions for a multi-level search space can be defined.

Corollary 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 1 the former are to be tested for each level l, while the latter only for $l > 1$.

Example 2. Let us consider the portion of space encompassing the $\mathcal{O}$-queries listed in Example 1. Note that $Q_0 \succeq Q_1$, $Q_0 \succeq Q_4$, $Q_1 \succeq Q_2$, $Q_1 \succeq Q_3$, $Q_1 \succeq Q_4$, $Q_4 \succeq Q_5$, $Q_4 \succeq Q_6$, $Q_2 \succeq Q_5$, and $Q_3 \succeq Q_6$. 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 1(ii).

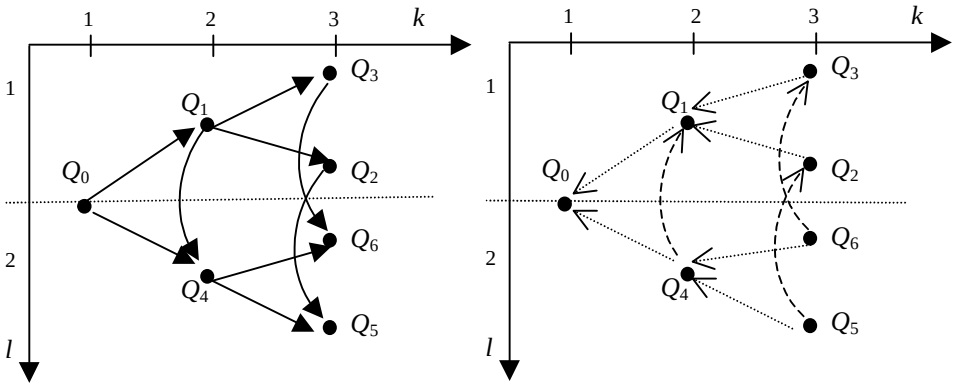

Fig. 2. Search graph (left) and related graph of backward pointers (right).

5 Implementing the $\mathcal{AL}$-log Refinement Operator $\rho_{\mathcal{O}}$

In the levelwise method each search stage generates, then evaluates patterns. 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 1 require a high number of subsumption checks to be performed. This makes candidate generation computationally expensive. So, we propose an implementation of the $\mathcal{AL}$-log 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 2 (right) gives an example of such graph, where nodes, dotted edges and dashed edges represent patterns, intra-space parenthood and inter-space parenthood, respectively. Following backward pointers allows us to save much computational effort, more precisely, to capitalize on the computational effort made when searching $\mathcal{L}[1]$. Indeed a k-pattern Q is generated from a $(k-1)$-pattern P in $\mathcal{L}[l]$, $l > 1$, by retrieving the result of the corresponding search stage in $\mathcal{L}[l-1]$

and casting it to the level l of description granularity. This makes inter-space subsumption checks and equivalence checks unnecessary.

Example 3. With reference to Example 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 1(ii).

This efficient approach to candidate generation has been implemented in SPADA [13], a system for mining multiple-level association rules from spatial data which can be considered an upgrade of [11] to Datalog-based representations. As mentioned before, association rule mining is usually performed in two steps. Here we focus our attention on the first step, i.e. frequent pattern discovery. The purpose of discovering spatial patterns is to detect associations between *reference objects* and *task-relevant objects* in a given spatial database. The former are the main subject of the description. The latter are spatially related to the former. 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 our context, a *spatial pattern* is a pattern P that contains at least one atom representing a spatial relationship. As usual in data mining, frequent patterns can be presented in the form of rules by filtering out those with low confidence. We call $Q \rightarrow R$ a *spatial association rule* if Q is a spatial pattern. The problem $\mathcal{P}$ of mining multi-level association rules in spatial data can be formulated as follows:

Given

- a spatial database, a set S of reference objects and some sets R_k, $1 \leq k \leq m$, of task-relevant objects,
- a taxonomy $\mathcal{T}$ involving objects in R_k,
- a set Ψ of description granularity levels,
- two thresholds, $minsup[l]$ and $minconf[l]$, for each level l in Ψ,

find strong multi-level spatial association rules.

In the case of geo-referenced data each R_k is typically a map layer and the taxonomy $\mathcal{T}$ is a collection of spatial hierarchies to be exploited to get descriptions of a given geographic area at different granularity levels. Spatial hierarchies capture is-a relations among locations on the basis of their geometry. For example, a problem instance $\imath(\mathcal{P})$ is the discovery of associations between large towns (S) 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. Preliminary experimental results on $\imath(\mathcal{P})$ are reported in [13]. Another problem instance and related results are illustrated in [14].

The implementation of $\rho_{\mathcal{O}}$ within SPADA has yielded a new release of the system. From now on we refer to SPADA 1.0 and SPADA 2.0 as the old and the new release of SPADA, respectively. The evaluation of the new candidate generation mechanism has been conducted by comparing the performance of SPADA

1.0 and SPADA 2.0 on $\imath(\mathcal{P})$ with thresholds $minsup[1] = 0.3$, $minsup[2] = 0.25$ and $minsup[3] = 0.2$. Table 1 reports quantitative results of this experiment level by level (l) step by step (k). Each couple of slash-separated figures indicate the ratio between the number of frequent patterns and the number of candidate patterns. No improvement is registered for frequent pattern discovery in $\mathcal{L}[1]$. As search proceeds towards finer description granularity levels SPADA 2.0 saves more and more computational effort. Note that the number of candidate patterns is one order of magnitude lower while the number of frequent patterns remains unchanged. This ensures the correctness of our approach. Results of frequent pattern discovery in $\mathcal{L}[2]$ and $\mathcal{L}[2]$ are emphasized in Figure 3.

Table 1. SPADA 1.0 vs SPADA 2.0: comparative evaluation.

		SPADA 1.0		SPADA 2.0	
$\mathcal{L}[l]$	k	*No. Patterns*	*Time (sec)*	*No. Patterns*	*Time (sec)*
1	2	3/3	27.69	3/3	27.47
	3	3/33		3/33	
	4	8/24		8/24	
	5	8/122		8/122	
	6	30/104		30/104	
	7	35/644		35/644	
	8	132/715		132/715	
2	2	3/3	328.73	3/3	7.08
	3	6/54		6/10	
	4	18/48		18/18	
	5	25/360		25/45	
	6	96/325		96/101	
	7	114/2487		114/230	
	8	481/2322		481/558	
3	2	3/3	246.78	3/3	14.56
	3	9/93		9/18	
	4	24/72		24/30	
	5	21/666		21/130	
	6	81/273		81/98	
	7	89/2757		89/454	
	8	355/1946		355/416	

6 Conclusions and Future Work

The trade-off between efficiency and expressive power is well known in the database community. Yet, recent extensions of the levelwise method for frequent pattern discovery witness a growing interest in more expressive representation languages. In this paper, we show that efficient algorithms can be designed for

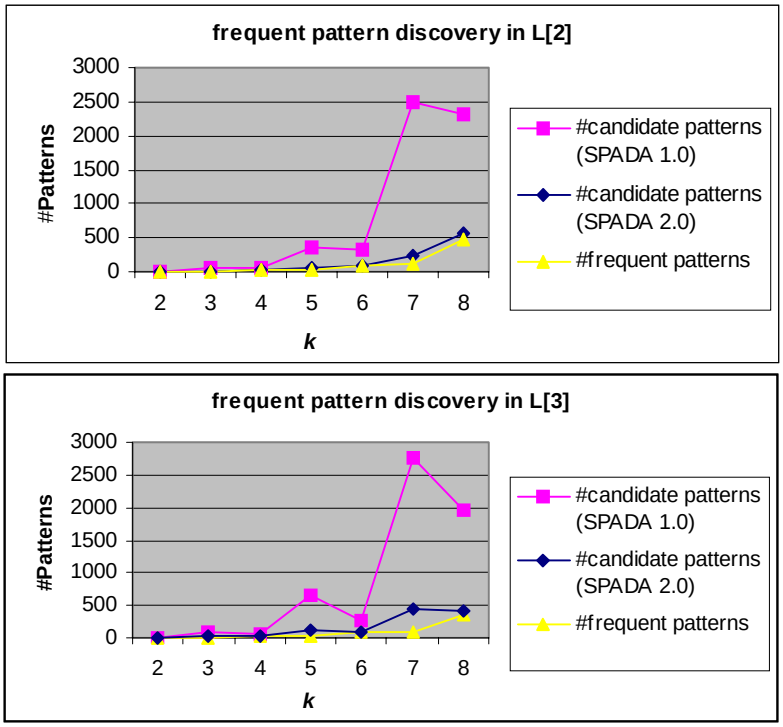

Fig. 3. SPADA 1.0 vs SPADA 2.0: frequent pattern discovery in $\mathcal{L}[2]$ and $\mathcal{L}[3]$.

discovering frequent patterns at multiple levels of description granularity. In particular, we have proposed an approach to candidate generation which is based on backward pointers to be updated while searching.

For the future we plan to investigate the properties of $\rho_{\mathcal{O}}$ and the impact of granularity assignment on experimental results. 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 given raise to new applications [8]. It would be interesting to extend our method in this direction.

References

1. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings of the 12th VLDB Conference*, 1994.
2. W. Buntine. Generalized subsumption and its application to induction and redundancy. *Artificial Intelligence*, 36(2):149–176, 1988.
3. S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer, 1990.

4. L. Dehaspe and H. Toivonen. Discovery of frequent DATALOG patterns. *Data Mining and Knowledge Discovery*, 3:7–36, 1999.
5. F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In Gerhard Brewka, editor, *Foundation of Knowledge Representation*, pages 191–236. CSLI-Publications, 1996.
6. F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. $\mathcal{AL}$-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.
7. U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: an overview. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press/The MIT Press, 1996.
8. F. Ferri, E. Pourabbas, M. Rafanelli, and F.L. Ricci. Linking geographic and multidimensional databases by functional attributes. In A. Celentano, L. Tanca, and P. Tiberio, editors, *Atti del IX Convegno Nazionale su Sistemi Evoluti per Basi di Dati*, pages 185–199, 2001.
9. J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In U. Dayal, P.M.D. Gray, and S. Nishio, editors, *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland*, pages 420–431. Morgan Kaufmann, 1995.
10. N. Helft. Inductive generalization: A logical framework. In I. Bratko and N. Lavrač, editors, *Progress in Machine Learning - Proceedings of EWSL87: 2nd European Working Session on Learning*, pages 149–157, Wilmslow, U.K., 1987. Sigma Press.
11. K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In M.J. Egenhofer and J.R. Herring, editors, *Advances in Spatial Databases*, volume 951 of *Lecture Notes in Computer Science*, pages 47–66. Springer, 1995.
12. A.Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104:165–209, 1998.
13. D. Malerba, F. Esposito, and F.A. Lisi. Mining Association Rules in Spatial Databases: A Logic-based Computational Method. In A. Celentano, L. Tanca, and P. Tiberio, editors, *Atti del IX Convegno Nazionale su Sistemi Evoluti per Basi di Dati*, pages 223–237, 2001.
14. D. Malerba and F.A. Lisi. Discovering Associations between Spatial Objects: An ILP Application. In C. Rouveirol and M. Sebag, editors, *Inductive Logic Programming*, volume 2157 of *Lecture Notes in Artificial Intelligence*, pages 156–163. Springer, 2001.
15. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
16. S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *LNAI*. Springer, 1997.
17. R. Reiter. Equality and domain closure in first order databases. *Journal of ACM*, 27:235–249, 1980.
18. C. Rouveirol and V. Ventos. Towards Learning in CARIN-$\mathcal{ALN}$. In J. Cussens and A. Frisch, editors, *Inductive Logic Programming*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 191–208. Springer, 2000.
19. M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
20. R. Srikant and R. Agrawal. Mining generalized association rules. In U. Dayal, P.M.D. Gray, and S. Nishio, editors, *Proceedings of 21th International Conference on Very Large Data Bases*, pages 407–419. Morgan Kaufmann, 1995.