# Towards Object-Relational Data Mining

Francesca A. Lisi and Donato Malerba

Dipartimento di Informatica, University of Bari, Italy,
{lisi, malerba}@di.uniba.it

**Abstract.** Mining data of complex data types (e.g. spatial, multimedial, etc.) is deemed an important research frontier in data mining. These data types are often currently modelled according to the object-relational data model. In this paper we face the problem of defining a general framework for mining object-relational databases instead of concentrating on specific forms of data mining tailored to specific data types (e.g. spatial data mining, multimedia data mining, etc.). Such a framework allows us to formulate data mining tasks in application domains characterized by objects, properties of objects, relations between objects and concept hierarchies. The hybrid language $\mathcal{AL}$-log and an ILP approach are the building blocks of the framework. Frequent pattern discovery at multiple levels of description granularity is taken as a showcase of ORDM.

## 1 Background and motivation

*Data models* play a relevant role in data mining [8]. Yet data mining techniques often make assumptions on the representation of input data which mismatch the data model adopted by the database to be mined. Indeed most techniques, here collectively referred to as Classical Data Mining (CDM), have been developed for data in the single-table form traditionally used in statistics. However, real-world data is seldom of this form. Rather, relational databases are widely used. Only recently a large body of research, named Relational Data Mining (RDM) and aimed at overcoming the limits of CDM in dealing with relational databases, has emerged [6]. We would like to emphasize that RDM is not simply data mining in relational databases. This definition would not be sufficient to distinguish it from CDM, which has been nonetheless extensively applied to relational databases. We define research on RDM as *the study of methods and techniques for discovering relational patterns in relational data* to emphasize that the relational data model is an invariant of the discovery process. RDM techniques have been mainly developed within the field of Inductive Logic Programming (ILP), a research area at the intersection of machine learning and logic programming [12]. Data in ILP is expected to be represented in Horn clausal logic. Therefore there is a natural fit between relational databases and ILP techniques as regards the data model. ILP was initially concerned with the synthesis of logic programs from examples and background knowledge. The recent developments, however, have broadened the range of learning problems of ILP from the traditional predictive tasks (classification rules) of machine learning to the descriptive ones

more peculiar to data mining. E.g., WARMR [4] is an ILP system for frequent pattern discovery where data and patterns are represented in DATALOG [3].

Mining data of complex *data types* (e.g. spatial, multimedial, etc.) is deemed an important research frontier in data mining [8]. These data types are often currently modelled according to the object-relational data model. In this paper we define Object-Relational Data Mining as *the study of methods and techniques for discovering object-relational patterns in object-relational data* and face the problem of defining a general framework for ORDM instead of concentrating on specific forms of data mining tailored to specific data types (e.g. spatial data mining, multimedia data mining, etc.). Such a framework allows us to formulate data mining tasks in application domains characterized by objects, properties of objects, relations between objects, and concept hierarchies (or taxonomies). An open question in ORDM research is: what approach? ILP seems a good candidate, except for the pure relational data model it adopts. Complex data types require appropriate representation and reasoning means. Description Logics (DLs) are particularly interesting because they have been invented for representing and reasoning with structural knowledge and concept hierarchies [1]. Unfortunately in exchange for the ability to model and reason about value restrictions in domains with a rich hierarchical structure, DLs offer a weaker than usual query language. This makes also pure DLs inadequate as a knowledge representation and reasoning means in ORDM problems. *Hybrid languages* such as $\mathcal{AL}$-log [5] appear more promising because they combine description logics and function-free Horn clausal logic. In this paper we show that $\mathcal{AL}$-log can be the starting point for the definition of a general framework for ORDM obtained by upgrading ILP solutions for RDM to ILP solutions for ORDM. Also we extend the work on spatial data mining presented in [10].

The paper is organized as follows. Section 2 recalls the link between description logics and databases. Section 3 defines the data mining task chosen as ORDM showcase. Section 4 presents the application of the $\mathcal{AL}$-log framework to the ORDM showcase. Section 5 concludes the paper with final remarks and directions for future work.

## 2 Description logics and databases

Description Logics (DLs) are fragments of first-order logic with equality [1]. From the beginning DLs have been considered general-purpose languages for knowledge representation and reasoning. They were considered especially effective for those domains where the knowledge could be easily organized along a hierarchical structure, based on the is-a relationship. This motivated the use of DLs as a modeling language in the design and maintenance of large, hierarchically structured bodies of knowledge. E.g. the description logic $\mathcal{ALC}$ [13] 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 by means of constructs, such as $\sqcap$

and $\sqcup$. An $\mathcal{ALC}$ knowledge base $\Sigma$ is a two-component system. The *intensional* component $\mathcal{T}$ consists of concept hierarchies spanned by is-a relations between concepts, namely *inclusion statements* of the form $C \sqsubseteq D$ (read "$C$ is included in $D$") where $C$ and $D$ are two arbitrary concepts. The *extensional* component $\mathcal{M}$ specifies instance-of relations, e.g. *concept assertions* of the form $a : C$ (read "$a$ belongs to $C$") where $a$ is an individual and $C$ is a concept. In $\mathcal{ALC}$ 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 $\mathcal{I}$). E.g., it maps concepts to subsets of $\Delta^{\mathcal{I}}$ and individuals to elements of $\Delta^{\mathcal{I}}$ such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$ (*unique names* assumption). We say that $\mathcal{I}$ is a *model* for $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and for $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$. The main reasoning services for $\mathcal{ALC}$ knowledge bases are checking whether $\Sigma$ logically implies an inclusion (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*. Both checks boil down to the more general problem of checking the satisfiability of an $\mathcal{ALC}$ knowledge base $\Sigma$.

The relationship between DLs and databases is also rather strong [1]. Several investigations have been carried out on the usage of DLs to formalize object-oriented data models and semantic data models. In these proposals concept descriptions in DLs are used to present the schema of a database. On the other hand, since a concept description provides necessary and sufficient conditions for objects to satisfy it, it is natural to treat it as a query, thus leading to a unification of two traditionally distinct languages: the data definition and data manipulation languages. Unfortunately in exchange for a more expressive description of the schema, DLs pay the price of a weaker than usual query language. Queries can only return subsets of existing concepts, rather than creating new concepts (as in standard SQL databases). Furthermore, the selection conditions are rather limited. Given the expressive limitations of DL concepts alone as queries, it has been reasonable to consider extending DATALOG queries with DLs. In one approach, exemplified by the $\mathcal{AL}$-log language [5], $\mathcal{ALC}$ concept assertions are used essentially as *type constraints* on variables appearing in function-free Horn clauses. E.g.,

`q(X)` $\leftarrow$ `item(X,Y)`, `item(X,Z)` & `X:Order`, `Y:DairyProduct`, `Z:GrainsCereals`

give a flavor of what unary conjunctive queries look like in $\mathcal{AL}$-log. Here the concept assertions `Y:DairyProduct` and `Z:GrainsCereals` restrict the range of the variables `Y` and `Z` to individuals of the concepts `DairyProduct` and `GrainsCereals` respectively. We claim that $\mathcal{AL}$-log can be used as a formal language for object-relational data models.

## 3 A case study for ORDM

A good representative of the class of data mining tasks which our framework can elegantly deal with is frequent pattern discovery at multiple levels of description granularity. This task and related issues have been already discussed in [10]. For the sake of brevity we only recall the formal problem statement.

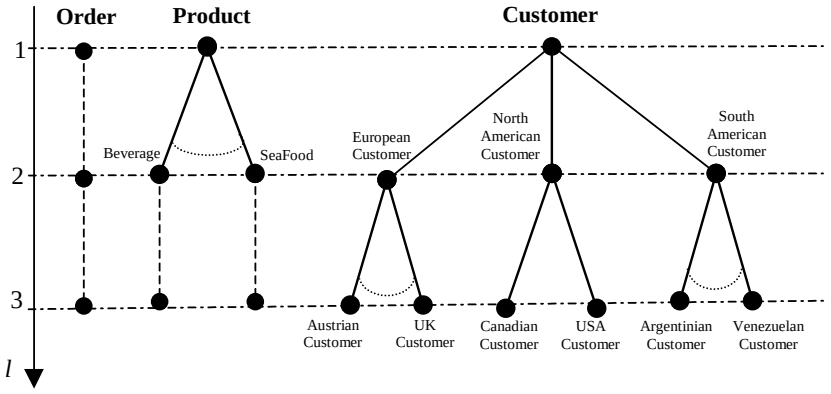


**Fig. 1.** Concept hierarchies for the N$\frac{\text{ORTHWIN}}{\text{TRADERS}}$D database.

**Definition 1.** *Given*

- *a data set* $\boldsymbol{r}$*,*
- *a taxonomy* $\mathcal{T}$ *where a reference concept and task-relevant concepts 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* $freq(\boldsymbol{r}, P)$*.*

This is a general formulation. Note that the usual formulation of frequent pattern discovery can be obtained for $\mathcal{T} = \emptyset$ and $maxG = 1$.

*Example 1.* Throughout this paper we shall refer to the N$\frac{\text{ORTHWIN}}{\text{TRADERS}}$D database which is distributed by Microsoft$^{TM}$ as a sample database for MS Access. An instantiation of Definition 1 for the case of N$\frac{\text{ORTHWIN}}{\text{TRADERS}}$D is sales analysis by finding associations between the category of ordered products and the geographic location of the customer within orders. Here the entity `Order` is the reference concept, and the entities `Product` and `Customer` are task-relevant concepts. In Figure 1 a three-layered taxonomy $\mathcal{T}$ is illustrated for the N$\frac{\text{ORTHWIN}}{\text{TRADERS}}$D database. All concept hierarchies in $\mathcal{T}$ have been rearranged according to the three problem-defined granularity levels. For instance, the concepts `Beverage`, ..., `SeaFood` in $\mathcal{H}_1 = \{\texttt{Beverage} \sqsubset \texttt{Product}, \ldots, \texttt{SeaFood} \sqsubset \texttt{Product}\}$ have been assigned to both $\mathcal{T}^2$ and $\mathcal{T}^3$ to make the hierarchies balanced.

Definition 1 does not make any assumption on either the data model adopted for $\mathbf{r}$ or the language $\mathcal{L}$ of patterns. Indeed the problem has been already faced both in CDM [7] and RDM [4]. In this paper we aim at investigating the issues raised by a simple yet meaningful object-relational representation.

*Example 2.* In WARMR **r** is a DATALOG knowledge base and $\mathcal{L}$ is a language of DATALOG queries generated starting from a WRMODE specification and ordered according to $\theta$-subsumption. With reference to the problem in Example 1 and assuming that $\mathcal{L}$ is defined with the following WRMODE specification[1]:

```
{orderID(-o), item(+o,-i), is_a(+i,beverage), ...,
purchaser(+o,-c), is_a(+c,europeanCustomer), ...}
```

WARMR can discover patterns such as the following DATALOG query $Q$

```
?- orderID(X), item(X,Y), is_a(Y,dairyProduct),
   item(X,W), is_a(W,grainsCereals), item(X,Z), is_a(Z,dairyProduct)
```

that are more expressive than patterns in CDM. In particular, variables allow for a direct representation of the links between the tables *Order Details*, *Products* and *Categories* and between tuples of *Order Details*. Though it overcomes the limits of the single-table assumption, WARMR suffers from some limits due to the pure relational data model. E.g., consider the DATALOG query $P$

```
?- orderID(A), item(A,B), is_a(B,dairyProduct),
   item(A,C), is_a(C,grainsCereals)
```

from the same $\mathcal{L}$ as above. It is more general than $Q$ because $P\theta \subseteq Q$ for $\theta = \{\mathtt{A/X}, \mathtt{B/Y}, \mathtt{C/Z}\}$. Also $Q$ is more general than $P$ because $Q\theta \subseteq P$ for $\theta = \{\mathtt{X/A}, \mathtt{Y/B}, \mathtt{W/C}, \mathtt{Z/B}\}$. We could expect that $P$ and $Q$ capture two distinct cases: the former the cases with exactly one dairy product, the latter the cases with exactly two dairy products (*object identity*).

Consider now the following DATALOG queries $Q_1$ and $Q_2$

```
?- orderID(A), item(A,B), is_a(B,dairyProduct),
   purchaser(A,C), is_a(C,europeanCustomer)
?- orderID(A), item(A,B), is_a(B,dairyProduct),
   purchaser(A,C), is_a(C,frenchCustomer)
```

Intuitively, looking at the concepts involved in the two queries, we can say that $Q_1$ is more general than $Q_2$ but $\theta$-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 *taxonomic reasoning*. E.g., it generates the following DATALOG queries

```
?- orderID(A), item(A,B), is_a(B,dairyProduct),
   purchaser(A,C), is_a(C,europeanCustomer)
?- orderID(A), item(A,B), is_a(B,dairyProduct),
   purchaser(A,C), is_a(C,europeanCustomer), is_a(C,frenchCustomer)
```

by simply adding `is_a` atoms from $\mathcal{L}$ without detecting semantic redundancies.

Object identity and taxonomic reasoning are central to object-relational representations. This induces us to investigate proper techniques for ORDM.

[1] These are directives for the candidate generation phase. Here the signs + and - stand for input and output variable, respectively. See [4] for further details.

## 4 The levelwise search in ORDM

Most algorithms of frequent pattern discovery follow the levelwise method for finding potentially interesting sentences of a given language [11]. Ingredients of this method are a data set $\mathbf{r}$, a language $\mathcal{L}$ of patterns, a generality relation $\succeq$ for $\mathcal{L}$, and a breadth-first search strategy for the space $(\mathcal{L}, \succeq)$. This section illustrates those ingredients in the $\mathcal{AL}$-log framework. The main feature of our framework is the extension of the unique names assumption from the semantic level to the syntactic one. In particular we resort to the bias of Object Identity [14]: In a formula, terms denoted with different symbols must be distinct, i.e. they represent different entities of the domain. This bias leads to a restricted form of substitution whose bindings avoid the identification of terms: A substitution $\sigma$ 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 $t_1\sigma \neq t_2\sigma$. In the following we assume substitutions to be OI-compliant.

### 4.1 The data set

Data is represented as an $\mathcal{AL}$-log knowledge base. An *$\mathcal{AL}$-log knowledge base* $\mathcal{B}$ is the pair $\langle \Sigma, \Pi \rangle$ where $\Sigma$ is an $\mathcal{ALC}$ knowledge base and $\Pi$ is a constrained DATALOG program. We remind that constraints are $\mathcal{ALC}$ concept assertions.

**Definition 2.** *A* constrained DATALOG clause *is an implication of the form* $E = \alpha_0 \leftarrow \alpha_1, \ldots, \alpha_m \& \gamma_1, \ldots, \gamma_n$ *where* $m \geq 0$, $n \geq 0$, $\alpha_i$ *are* DATALOG *atoms and* $\gamma_j$ *are constraints. We denote by* $head(E)$ *the head* $\alpha_0$*, and by* $body(E)$ *the body* $\leftarrow \alpha_1, \ldots, \alpha_m \& \gamma_1, \ldots, \gamma_n$ *of* $E$*.*

The interaction between the structural and the relational part of $\mathcal{B}$ allows for the extension of terminology and results related to the notion of *substitution* from DATALOG to $\mathcal{AL}$-log in a straightforward manner. This interaction is also at the basis of a model-theoretic semantics for $\mathcal{AL}$-log. We call $\Pi_D$ the set of DATALOG clauses obtained from the clauses of $\Pi$ 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 $\Sigma$ (i.e. an interpretation compliant with the unique names assumption) and an Herbrand interpretation $\mathcal{I}_\mathcal{H}$ for $\Pi_D$. An interpretation $\mathcal{J}$ is a *model* of $\mathcal{B}$ if $\mathcal{I}_\mathcal{O}$ is a model of $\Sigma$, and for each ground instance $\bar{\alpha}' \& \gamma'_1, \ldots, \gamma'_n$ of each clause $\bar{\alpha} \& \gamma_1, \ldots, \gamma_n$ in $\Pi$, either there exists one $\gamma'_i$, $i \in \{1, \ldots, n\}$, 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. A *query* $Q$ to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$ is a constrained DATALOG clause of the form $\leftarrow \alpha_1, \ldots, \alpha_m \& \gamma_1, \ldots, \gamma_n$ .

**Definition 3.** *Let* $\mathcal{B}$ *be a* $\mathcal{AL}$*-log knowledge base. An* answer *to the query* $Q$ *is a ground substitution* $\sigma$ *for the variables in* $Q$*. The answer* $\sigma$ *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 in $\mathcal{AL}$-log is based on constrained SLD-resolution [5]. Here hybridization requires SLD-resolution to perform some reasoning on the structural component of $\mathcal{B}$. This is done by applying the propagation rules of a tableau calculus to the outcome of SLD-resolution [5].

**Definition 4.** *Let $\mathcal{B}$ be a $\mathcal{AL}$-log knowledge base. An answer $\sigma$ to a query $Q$ is called a* computed answer *if there exists a constrained SLD-refutation for $Q\sigma$ in $\mathcal{B}$ ($\mathcal{B} \vdash Q\sigma$). The set of computed answers is called the* success set *of $Q$ in $\mathcal{B}$.*

*Example 3.* As a running example, we consider an $\mathcal{AL}$-log knowledge base $\mathcal{B}$ obtained from the $\mathrm{N}\frac{\text{ORTHWIN}}{\text{TRADERS}}\mathrm{D}$ database. The structural subsystem $\Sigma$ should reflect the E/R model underlying the $\mathrm{N}\frac{\text{ORTHWIN}}{\text{TRADERS}}\mathrm{D}$ database. To serve our illustrative purpose we focus on the concepts (entities) `Order`, `Product` and `Customer`. The intensional part of $\Sigma$ encompasses inclusion statements such as `DairyProduct` $\sqsubset$ `Product` and `FrenchCustomer` $\sqsubset$ `EuropeanCustomer` that represent the two taxonomies illustrated in Figure 1. The extensional part of $\Sigma$ contains 830 concept assertions for `Order` (e.g. `order10248:Order`), 77 assertions for the sub-concepts of `Product`, e.g. `product11:DairyProduct`, and 91 assertions for the sub-concepts of `Customer`, e.g. `'VINET':FrenchCustomer`. The relational subsystem $\Pi$ expresses the $\mathrm{N}\frac{\text{ORTHWIN}}{\text{TRADERS}}\mathrm{D}$ database as a constrained DATALOG program. We restrict ourselves to the relations Order and OrderDetail. The extensional part of $\Pi$ consists of 830 facts for `order`/14 and 2155 facts for `orderDetail`/5, e.g. `orderDetail(order10248,product11,'£14',12,0.00)`, represents the order detail concerning the order number 10248 and product code 11. The intensional part of $\Pi$ defines two views on `order` and `orderDetail`:

```
item(OrderID,ProductID)
    ← orderDetail(OrderID,ProductID,_,_,_)
        & OrderID:Order, ProductID:Product
purchaser(OrderID,CustomerID)
    ← order(OrderID,CustomerID,_,_,_,_,_,_,_,_,_,_,_,_)
        & OrderID:Order, CustomerID:Customer
```

When triggered on the EDB of $\Pi$ these rules can deduce implicit facts such as `item(order10248,product11)` and `purchaser(order10248,'VINET')`.

### 4.2 The language of patterns

Patterns are represented as $\mathcal{O}$-queries, a rule-based form of unary conjunctive queries whose answer set contains individuals of an $\mathcal{ALC}$ concept $\hat{C}$ of reference.

**Definition 5.** *Given a reference concept $\hat{C}$, 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. *We denote by $key(Q)$ the key constraint $X : \hat{C}$ of $Q$. A* trivial *$\mathcal{O}$-query is a clause of the form $q(X) \leftarrow \& X : \hat{C}$.*

We impose $\mathcal{O}$-queries to be linked and connected (or range-restricted) constrained DATALOG clauses. The language $\mathcal{L}$ of descriptions for a given mining problem is implicitly defined by a set $\mathcal{A}$ of atom templates, a key constraint $\hat{\gamma}$, and an additional set $\Gamma$ of constraint templates. An atom template $\alpha$ specify

name and arity of the predicate and mode of its arguments. An instantiation of $\alpha$ is a DATALOG atom with predicate and arguments that fulfill the requirements specified in $\alpha$. Constraint templates specify the concept name for concept assertions and determine the granularity level $l$ of descriptions.

*Example 4.* Following Example 3, let $\mathcal{A}$={`item(+,-)`, `purchaser(+,-)`} and $\hat{\gamma}$ be the key constraint built on the concept `Order`. Suppose that we are interested in descriptions at two different granularity levels. Thus $\mathcal{T}$ consists of the two layers shown in Figure 1 from which the sets $\Gamma^1$ and $\Gamma^2$ of constraints are derived. Examples of $\mathcal{O}$-queries belonging to this language are:

$Q_0$= `q(X)` ← & `X:Order`
$Q_1$= `q(X)` ← `item(X,Y)` & `X:Order`
$Q_2$= `q(X)` ← `purchaser(X,Y)` & `X:Order`
$Q_3$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Product`
$Q_4$= `q(X)` ← `purchaser(X,Y)` & `X:Order`, `Y:Customer`
$Q_5$= `q(X)` ← `item(X,Y)`, `item(X,Z)` & `X:Order`, `Y:Product`
$Q_6$= `q(X)` ← `item(X,Y)`, `purchaser(X,Z)` & `X:Order`, `Y:Product`
$Q_7$= `q(X)` ← `item(X,Y)`, `item(X,Z)` & `X:Order`, `Y:Product`, `Z:Product`
$Q_8$= `q(X)` ← `item(X,Y)`, `purchaser(X,Z)` & `X:Order`, `Y:Product`, `Z:Customer`
$Q_9$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:DairyProduct`
$Q_{10}$= `q(X)` ← `purchaser(X,Y)` & `X:Order`, `Y:EuropeanCustomer`
$Q_{11}$= `q(X)` ← `item(X,Y)`, `item(X,Z)` & `X:Order`, `Y:DairyProduct`
$Q_{12}$= `q(X)` ← `item(X,Y)`, `purchaser(X,Z)` & `X:Order`, `Y:DairyProduct`
$Q_{13}$= `q(X)` ← `item(X,Y)`, `item(X,Z)`
& `X:Order`, `Y:DairyProduct`, `Z:GrainsCereals`
$Q_{14}$= `q(X)` ← `item(X,Y)`, `purchaser(X,Z)`
& `X:Order`, `Y:DairyProduct`, `Z:EuropeanCustomer`

In particular, $Q_0$ and $Q_1$ are valid for both $\mathcal{L}^1$ and $\mathcal{L}^2$, $Q_3$ and $Q_5$ belong to $\mathcal{L}^1$, and $Q_9$ belongs to $\mathcal{L}^2$. Note that all of them are linked and connected.

An *answer* to an $\mathcal{O}$-query $Q$ is a ground substitution $\theta$ for the distinguished variable of $Q$. The aforementioned conditions of well-formedness guarantee that the evaluation of $\mathcal{O}$-queries is sound according to the following definitions of answer set and success set.

**Definition 6.** *Let $\mathcal{B}$ be a $\mathcal{AL}$-log knowledge base. An answer $\theta$ to an $\mathcal{O}$-query $Q$ is a* correct (resp. computed) answer *w.r.t. $\mathcal{B}$ if there exists at least one correct (resp. computed) answer to body$(Q)\theta$ w.r.t. $\mathcal{B}$.*

Query answering is necessary for computing the support of patterns during *candidate evaluation* phases. Support is defined as the ratio between the number of individuals in the reference concept $\hat{C}$ that satisfy the candidate pattern and the total number of individuals in $\hat{C}$.

**Definition 7.** *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 $\mathcal{L}^l$.*

*Example 5.* A correct answer to $Q_0$, $Q_3$ and $Q_9$ w.r.t. $\mathcal{B}$ is the substitution $\theta$ ={X/order10248}. In total we have that $answerset(Q_0, \mathcal{B})$ contains 830 answers (as many as the number of individuals for the concept Order), $answerset(Q_3, \mathcal{B})$ 830 answers as well (since the conditions in the body of $Q_3$ are not strong enough to filter the individuals of Order) and $answerset(Q_9, \mathcal{B})$ 303 answers. Therefore, $\sigma(Q_3, \mathcal{B}) = 100\%$ and $\sigma(Q_9, \mathcal{B}) = 36.5\%$.

The definition of a generality order 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 [2] has been introduced in ILP as a generality order for Horn clauses with respect to background knowledge. We propose to adapt it to our $\mathcal{AL}$-log framework as follows.

**Definition 8.** *Let $Q$ be an $\mathcal{O}$-query, $\alpha$ a ground atom, and $\mathcal{J}$ an interpretation. We say that $Q$* covers *$\alpha$ under $\mathcal{J}$ if there is a ground substitution $\theta$ for $Q$ ($Q\theta$ is ground) such that $body(Q)\theta$ is true under $\mathcal{J}$ and $head(Q)\theta = \alpha$.*

**Definition 9.** *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 $\alpha$ such that $Q$ covers $\alpha$ under $\mathcal{J}$, we have that $P$ covers $\alpha$ under $\mathcal{J}$.*

We have defined a quasi-order $\succeq_{\mathcal{B}}$ for $\mathcal{O}$-queries on the basis of $\mathcal{B}$-subsumption and provided a decidable procedure to check $\succeq_{\mathcal{B}}$ on the basis of constrained SLD-resolution [9]. Note that the underlying reasoning mechanism of $\mathcal{AL}$-log makes $\mathcal{B}$-subsumption more powerful than generalized subsumption.

**Theorem 1.** *Let $P$ and $Q$ be two $\mathcal{O}$-queries to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$ and $\sigma$ 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 $\theta$ 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.*

Furthermore $\succeq_{\mathcal{B}}$ is monotonic with respect to the evaluation function $\sigma$.

**Lemma 1.** *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 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 $\sigma(P, \mathcal{B}) \geq \sigma(Q, \mathcal{B})$.*

*Proof.* *Let $\theta \in answerset(Q, \mathcal{B})$. By Lemma 1, 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 1, there exists a substitution $\gamma$ for $P$ such that $head(P)\gamma = head(Q)$. By Definition 9, it holds that $\theta \in answerset(P\gamma, \mathcal{B})$. Since $answerset(Q, \mathcal{B}) \subseteq answerset(P\gamma, \mathcal{B})$ and $\gamma$ simply renames the distinguished variable of $P$, the thesis follows from Definition 7.*

Quasi-ordered sets can be searched by refinement operators [12]. From Proposition 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 operators for searching multi-level pattern spaces. To this aim, given two $\mathcal{ALC}$ constraints $\gamma_1 = t_1 : C$ and $\gamma_2 = t_2 : D$, we say that $\gamma_1$ is *at least as strong as* (resp. *stronger than*) $\gamma_2$, denoted as $\gamma_1 \succeq \gamma_2$ (resp. $\gamma_1 \succ \gamma_2$), iff $t_1 = t_2$ and $C \sqsubseteq D$ (resp. $C \sqsubset D$).

**Definition 10.** *Let maxD be the search depth bound, and $\mathcal{L} = \{\mathcal{L}^l\}_{1 \leq l \leq maxG}$ be a language of $\mathcal{O}$-queries. A (downward) refinement operator $\rho_{\mathcal{O}}$ for $(\mathcal{L}, \succeq_{\mathcal{B}})$ is defined such that, for a given $\mathcal{O}$-query*

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

*in $\mathcal{L}^l$, $l < maxG$, $m + n < maxD$, the set $\rho_{\mathcal{O}}(P)$ contains all $Q \in \mathcal{L}$ that can be obtained by applying one of the following refinement rules:*

$\langle Atom \rangle$ $Q = q(X) \leftarrow \alpha_1, \ldots, \alpha_m, \alpha_{m+1} \& X : \hat{C}, \gamma_2, \ldots, \gamma_n$ *where $\alpha_{m+1}$ is an instantiation of an atom template in $\mathcal{A}$ such that $\alpha_{m+1} \notin body(P)$.*

$\langle Constr \rangle$ $Q = q(X) \leftarrow \alpha_1, \ldots, \alpha_m \& X : \hat{C}, \gamma_2, \ldots, \gamma_n, \gamma_{n+1}$ *where $\gamma_{n+1}$ is an instantiation of a constraint template in $\Gamma^l$ such that $\gamma_{n+1}$ constrains an unconstrained variable in $body(P)$.*

$\langle \forall C \rangle$ $Q = q(X) \leftarrow \alpha_1, \ldots, \alpha_m \& X : \hat{C}, \gamma'_2, \ldots, \gamma'_n$ *where each $\gamma'_j$, $2 \leq j \leq n$, is an instantiation of a constraint template in $\Gamma^{l+1}$ such that $\gamma'_j \succeq \gamma_j$ and at least one $\gamma'_j \succ \gamma_j$.*

The rules $\langle Atom \rangle$ and $\langle Constr \rangle$ help 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 correct, i.e. the $Q$'s obtained by applying any of these refinement rules to $P \in \mathcal{L}^l$ are $\mathcal{O}$-queries such that $P \succeq_{\mathcal{B}} Q$ [9].

*Example 6.* Following Example 4, $\rho_{\mathcal{O}}(Q_1)$ is the set

$Q'_1$= `q(X)` ← `item(X,Y)`, `item(X,Z)` & `X:Order`
$Q'_2$= `q(X)` ← `item(X,Y)`, `purchaser(X,Z)` & `X:Order`
$Q'_3$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Product`
$Q'_4$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Customer`
$Q'_5$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Beverage`
$Q'_6$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Condiment`
$Q'_7$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Confection`
$Q'_8$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:DairyProduct`
$Q'_9$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:GrainsCereals`
$Q'_{10}$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:MeatPoultry`
$Q'_{11}$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Produce`
$Q'_{12}$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:SeaFood`
$Q'_{13}$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:EuropeanCustomer`
$Q'_{14}$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:NorthAmericanCustomer`
$Q'_{15}$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:SouthAmericanCustomer`

where the $\mathcal{O}$-queries $Q'_1$ and $Q'_2$ are generated by means of $\langle Atom \rangle$, the $\mathcal{O}$-queries $Q'_3$ and $Q'_4$ by means of $\langle Constr \rangle$, and the $\mathcal{O}$-queries from $Q'_5$ to $Q'_{15}$ also by means of $\langle Constr \rangle$ (but considering $Q_1$ as belonging to $\mathcal{L}^2$). Note that $Q'_4$, $Q'_{13}$, $Q'_{14}$, and $Q'_{15}$ will turn out to be infrequent. Yet they are generated. What matters while searching $(\mathcal{L}, \succeq_{\mathcal{B}})$ is to find patterns that are more specific than a given $P$ under $\mathcal{B}$-subsumption. Conversely, $\rho_{\mathcal{O}}(Q_3)$ is the set

$Q''_1$= `q(X)` ← `item(X,Y)`, `item(X,Z)` & `X:Order`, `Y:Product`
$Q''_2$= `q(X)` ← `item(X,Y)`, `purchaser(X,Z)` & `X:Order`, `Y:Product`
$Q''_3$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Beverage`
$Q''_4$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Condiment`
$Q''_5$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Confection`
$Q''_6$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:DairyProduct`
$Q''_7$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:GrainsCereals`
$Q''_8$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:MeatPoultry`
$Q''_9$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Produce`
$Q''_{10}$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:SeaFood`

where the $\mathcal{O}$-queries $Q''_1$ and $Q''_2$ are generated by means of $\langle Atom \rangle$, and the $\mathcal{O}$-queries from $Q''_3$ to $Q''_{10}$ by means of $\langle \forall C \rangle$. Note that the query $Q_9$ can be obtained by applying either $\langle Constr \rangle$ to $Q_1$ (here $Q_1$ is considered as belonging to $\mathcal{L}^2$) or $\langle \forall C \rangle$ to $Q_3$. Actually each node in $\mathcal{L}^2$ can be reached starting from either another node in $\mathcal{L}^2$ or a node in $\mathcal{L}^1$. This can be exploited to speed up the search at levels of finer granularity as shown in [10].

## 5 Conclusions

Hybrid languages such as $\mathcal{AL}$-log witness a strict relationship between DLs and DBs. In this paper we have shown the potential benefit of using hybrid languages in data mining. As representation languages they allow for the uniform treatment of both structural and relational knowledge. This ability is particularly appropriate for dealing with complex data types. Actually we have defined a framework that by-passes the level of data types and tackles with the level of data models. As a feasibility study for our framework, we have revised Mannila's levelwise method by extending previous work in RDM to a simple yet meaningful object-relational representation. Indeed we have motivated the need for ORDM by emphasizing the limits of WARMR in dealing with object identity and concept hierarchies. For the future we plan to evaluate empirically the framework by conducting experiments on real-world large object-relational databases. To this aim and following the work presented in [10] algorithms that can help mitigating the trade-off between efficiency and expressive power are under development. Furthermore this framework is general enough to serve as a starting point for defining other ORDM tasks. A natural evolution would be moving from mining at multiple levels of granularity to *mining at multiple levels of abstraction*. An extension of our work in this direction will require the investigation of properties (e.g. monotonicity) of user-defined aggregates in object-relational systems [15] and the definition of a refinement operator compliant with these properties.

**Acknowledgements**

This work has been supported by the annual Scientific Research Project "Scoperta di conoscenza in basi di dati: metodi e tecniche efficienti e robuste per dati complessi" - Year 2002 - funded by the University of Bari.

## References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2002.
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. 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.
6. S. Džeroski and N. Lavrač, editors. *Relational Data Mining*. Springer, 2001.
7. J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In U. Dayal, P. 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.
8. J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, CA, 2001.
9. F. Lisi. *An ILP Setting for Object-Relational Data Mining*. Ph.D. Thesis, Department of Computer Science, University of Bari, Italy, 2002.
10. F. Lisi and D. Malerba. Efficient Discovery of Multiple-Level Patterns. In P. Ciaccia, F. Rabitti, and G. Soda, editors, *Atti del Decimo Convegno Nazionale su Sistemi Evoluti per Basi di Dati*, pages 237–250. Centro Stampa 2P,Pontassieve, Italy, 2002.
11. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
12. S. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer, 1997.
13. M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
14. G. Semeraro, F. Esposito, D. Malerba, N. Fanizzi, and S. Ferilli. A logic framework for the incremental inductive synthesis of Datalog theories. In N. Fuchs, editor, *Proceedings of 7th International Workshop on Logic Program Synthesis and Transformation*, volume 1463 of *Lecture Notes in Computer Science*, pages 300–321. Springer, 1998.
15. H. Wang and C. Zaniolo. User Defined Aggregates in Object-Relational Systems. In *Proceedings of the 16th International Conference on Data Engineering*, pages 135–144. IEEE Computer Society, 2000.