Integrated Design and Process Technology, **IDPT-Vol. 2**
Printed in the United States of America, July, 1998

AUTOMATIC EXTRACTION OF DATABASE SCHEME SEMANTIC PROPERTIES USING KNOWLEDGE DISCOVERY TECHNIQUES

A. Bonifati, L. Palopoli, D. Saccá and D. Ursino
DEIS, Universitá della Calabria
Rende (CS), Italy

ABSTRACT

This paper presents a number of techniques by which useful interscheme properties can be automatically extracted from pre-existing database schemes. The presented approach uses a variant of description logics enriched with plausibility factors to represent and reason about scheme properties. Our method comprises algorithms for terminological normalization of schemes, for extraction of basic inclusion properties between simple scheme objects, for selection of relevant subschemes and, finally, for the derivation of semantic scheme substructures, defined as complex expressions of our descriptive language. It is shown that the availability of such derived properties can be advantageous in various application contexts, including federated database integration, query optimization and view maintenance.

1 Introduction

The problem of understanding the intensional semantics of databases is an essential task for dealing with data management and utilization in the most appropriate way. Recent papers point out such need by discussing various aspects of using knowledge about database schemes in various application contexts, including accessing integrated and cooperative data resource systems (Ullman (1987), Levy et al. (1996), Catarci - Lenzerini (1993), Abiteboul (1997)), query optimization and view maintenance (Chaudhuri et al. (1995)), structuring of data warehouses and data constraints (Gupta et al. (1995)).

Some of these papers put into evidence the need for the adoption of formal languages to describe and manipulate intensional knowledge about data. (Catarci - Lenzerini (1993)) proposes a logic formalism largely based on Description Logics (DL) to express interscheme properties in cooperative database systems. Ullman (1997) discusses mediators for obtaining and filtering information from heterogeneous pre-existing data resources in integrated information systems. Mediator capabilities are obtained using logic-based formalisms. For instance, the Information Manifold uses an extension of DL with Datalog-like rules (Levy et al. (1996)).

A difficulty that is encountered while reasoning about intensional semantics of pre-existing databases is that, most often, the only available intensional knowledge consists of properties explicitly encoded in database schemes whereas many other useful properties are implicit, hidden within scheme structures and, as such, cannot be immediately exploited.

The aim of this paper is to show how such useful properties can be extracted from database schemes applying a process which can be looked at as a special knowledge discovery process (Fayyad et al. (1996)). Contrary to traditional knowledge discovery techniques, our methods work mainly on schemes rather than on extensional data[1].

In order to represent and manipulate intensional knowledge, we use a logic language, called PDL, which is obtained by extending the language in (Catarci - Lenzerini (1993)) along two directions: first, besides expressing structural properties of object classes, most notably, class inclusions, our formalism allows also the expression of synonymies and homonymies between class names (we will call such properties *nominal* properties); second, each logic assertion is associated with a real number between 0 and 1, used to measure assertion's strength, in a way that will be explained shortly.

The adoption of such DL based formalism is moti-

[1]Even if in several application contexts, when multiple and heterogeneous data sources are involved, the sharp distinction traditionally made in databases between schemes and data looses much of its importance (Abiteboul (1997)).

vated by two reasons. First, Description Logic is well suited for representing complex semantic properties of represented domains, and indeed we focus on complex semantic properties of database schemes. Second, DL features a precise formal inference system, which we take advantage of as the basis for deriving scheme properties.

The approach we are presenting here consists in two main phases:

Phase 1, or preprocessing, where database schemes are analyzed for extracting an initial set of assertions. In this phase, database experts are required to provide some basic background knowledge about involved databases which includes (1) a set of (so called, basic) synonymy or homonymy properties and (2) a set of further assertions describing inclusions between classes of objects. This latter knowledge will typically relate classes of objects belonging to the *same* scheme (whereas our algorithm will typically extract properties relating classes of objects belonging to *different* schemes).

Phase 2, which is devoted to discovering more complex properties and consists in (1) focusing on relevant database objects and (2) discovery of new properties they are involved in. Some of the techniques used in Phase 1 have been already presented as part of another paper (La Camera et al. (1997)), and are reported here for the sake of the presentation.

Summarizing, the main contributions of this paper are the following: *(i)* the presentation of a formalism based on DL for the representation of scheme properties; *(ii)* the definition of a set of techniques for the semi automatic discovering of hidden database properties; *(iii)* the illustration of the applicability of extracted properties in various database contexts including database scheme integration, query optimization and view maintenance.

Before closing this section, we briefly overview some literature related to what is presented in this paper. In the literature other methodologies exist for deriving inclusion, synonymy and other properties among scheme objects (Batini et al., (1996), Castano - De Antonellis (1997)). Our methodology derives this type of properties but, in addition, it derives more complex properties and associates to each property a coefficient which expresses its strength. In (Castano - De Antonellis (1997)) a method for the construction of a semantic dictionary is described; this method retrieves only similarity and dissimilarity information. Our method yields a far more richer derivation of properties. Finally, in (Catarci - Lenzerini (1993)) a formalism for representing interscheme knowledge based on DL is described, which forms the basis of our PDL formalism. However, (Catarci - Lenzerini (1993)) aims at *representing* interscheme knowledge while our aim is to *extract* new knowledge and to use PDL for formally representing extracted properties.

The rest of this paper is organized as follows. In the next section we overview some preliminary concepts, namely, the description language of (Catarci - Lenzerini (1993)) and knowledge discovery background. Then, in Section 3 we illustrate the preprocessing phase. Section 4 presents main methods for extracting properties. Some application cases are presented in Section 5.

2 Intentional Knowledge Discovery in Databases

2.1 Knowledge Discovery in Databases

During the last decades there has been an enormous growth in the amount of data stored in database systems. To indicate the systematic organization of such data collections within structured information depository, the term Data Warehousing was coined. Because of their overwhelming size, information stored in data warehouses cannot be properly queried using traditional techniques, as very often they are not able to produce data reporting of reasonably small size. Therefore, the necessity arises to employ special techniques capable to assist, in an intelligent and semi-automatic way, operators in retrieving useful information from large databases. The cultural area focusing on devising such data extraction techniques is known as Knowledge Discovery in Databases (KDD) and is attracting a growing interest in the database community (Fayyad et al. (1996)).

A KDD process consists of several steps: creation of target data sets, noise elimination and preprocessing, data reduction, data mining algorithm execution, interpretation of extracted data patterns, strengthening of discovered knowledge (Fayyad et al. (1996)). KDD systems usually interact with users, who are requested to make several choices by which data processing is influenced and supervised. The computational core of KDD processes consists in data mining algorithms, which perform the actual extraction of interesting pattern from data.

In this paper, we concentrate on such core phase and present some techniques by which interesting patterns can be extracted from database schemes rather than from extensional data. We have coined the term *Intentional Data Mining* to refer to this process of extracting new information from database schemes. The associated Knowledge Discovery process is similarly called *Intentional Knowledge Discovery* (IKDD).

2.2 PDL: A Description Logic for IKDD

To reason about database intensional properties, we use a variant of DL, which we call PDL, which extends the logics presented in (Catarci - Lenzerini (1993)). We implicitly assume that our input database schemes are represented in PDL. This is not an actual limitation, since translations exist from all data models (Catarci - Lenzerini (1993)). However, for the sake of the presentation, we shall often refer to Entity Relationship database schemes.

Description logics is a specialized logics developed to model complex data domains. As in (Catarci - Lenzerini (1993)) we use DL to reason about class properties. The language is based on an alphabet B of symbols including class names and the special symbols $\top$, $\bot$, $\sqcap$, $\sqcup$, $\exists$, $\forall$ and usual parentheses. A class expression is either an entity expression or a relationship expression. An entity expression over the alphabet B is constructed according to the following rules:

$$\begin{aligned} C, F \longrightarrow\ & E|\ \ C \sqcup F|\ \ C \sqcap F|\ \ \neg C| \\ & \forall R[U].T_1 : C_1, ..., T_n : C_n| \\ & \exists R[U].T_1 : C_1, ..., T_n : C_n| \\ & \forall A.D|\ \ \exists A.D \end{aligned}$$

where C, F and E are entity expressions, R is a relationship symbol from B, and $T_1, \ldots, T_n, U$ are role symbols.

A relationship expression is an expression of the form $R[U_1, U_2, ..., U_n]$ where R is a relationship symbol over the alphabet B and $\{U_1, U_2, ..., U_n\} = rol(R)$ are the roles associated to R.

Knowledge about scheme properties is expressed in such a logic in the form of assertions. An assertion is a statement of the form: $L_1 \leq L_2$ where L_1 and L_2 are class expressions of the same type.

Language semantics is based on interpretations. An interpretation $I = (\Delta^I, .^I)$ consists of: (1) a non empty set Δ^I, called *universe of I* which comprises all the objects, (2) a mapping $.^I$, called *interpretation function* of I. For each I, the interpretation function of I assigns to each entity expression a subset of Δ^I, according to the following rules:

$$\top^I = \Delta^I \qquad \bot^I = \emptyset \qquad (C \sqcap F)^I = C^I \cap F^I$$
$$(C \sqcup F)^I = C^I \cup F^I \qquad (\neg C)^I = \{a \in \Delta^I \mid a \notin C^I\}$$
$$(\forall R[U].T_1 : C_1, ..., T_n : C_n)^I = \{a \mid \forall r \in R^I.(r[U] = a) \Rightarrow (r[T_1] \in C_1^I \wedge ... \wedge r[T_n] \in C_n^I)\}$$
$$(\exists R[U].T_1 : C_1, ..., T_n : C_n)^I = \{a \mid \exists r \in R^I.(r[U] = a) \wedge (r[T_1] \in C_1^I \wedge ... \wedge r[T_n] \in C_n^I)\}$$
$$(\forall A.D)^I = \{a \mid \forall (a, b) \in A^I.b \in D^I\}$$
$$(\exists A.D)^I = \{a \mid \exists (a, b) \in A^I.b \in D^I\}$$

The interpretation function of I assigns a set of labeled tuples to each relationship expression as:

$$(R[U_1, U_2, ..., U_n])^I = R^I$$

where, if R is a relationship with roles $\{U_1, U_2, ..., U_m\}$, R^I is a set of labeled tuples of the form $\langle U_1 : u_1, ..., U_m : u_m \rangle$ and $u_1, ..., u_m \in \Delta^I$. In the following, if r is an instance of R we shall use $r[U_i]$ to denote the object associated to U_i by R. An *assertion* $L_1 \leq L_2$ is satisfied by an interpretation I if $L_1^I \subseteq L_2^I$. An interpretation I is called a *model* of a set of assertions Σ if each assertion in Σ is satisfied by I.

In the following, we shall use the language PDL, which extends the descriptive language presented above as follows: (1) each assertion is associated to a coefficient, which is a real number between 0 and 1 which measures the plausibility of the assertion. An assertion $L_1 \leq L_2$ with inclusion coefficient $W_{L_1L_2}$ will be represented as a triplet $\langle L_1, L_2, W_{L_1L_2} \rangle_{Inc}$, where the suffix *Inc* tells us the triplet is relative to an inclusion property. (2) besides inclusion properties, PDL allows to represent also what we call nominal properties, which express synonymy and homonymy characterizations of class names. Nominal properties are associated with synonymy/homonymy coefficients. $\langle L_1, L_2, V_{L_1L_2} \rangle_{Syn}$, tells that a synonymy is believed to exist between the name L_1 and the name L_2 with a confidence equal to $V_{L_1L_2} \times 100$ percent. Homonymies are represented similarly, their plausibility coefficient being represented by the letter Z. Note that both kinds of nominal properties make sense only between classes belonging to different input database schemes.

3 Phase 1: Preprocessing

Assume we are given a set of database schemes S. The preprocessing phase has two steps: Step 1, consisting in the extraction of nominal properties about classes belonging to schemes in S, and Step 2, which consists in deriving *simple* inclusion properties, i.e., properties of the form $\langle A, C, f \rangle_{Inc}$ where both A and C are class symbols.

3.1 Extracting Basic Nominal Properties

The process of synonymy extraction is based on the idea that nominally similar classes are most often used in a structurally similar manner into schemes. In turn, the structural similarity analysis is recursively mediated through nominal similarity. The process starts with a small set of synonymy properties, called basic synonymy properties, provided by experts. Then, given a pair of candidate classes for synonymy, the extraction algorithm analyzes the class neighborhood to determine the similarity degree of the two classes. The algorithm proceeds incrementally, by storing discovered synonymies in the dictionary, and possibly uses this new knowledge to derive further synonymies. It terminates when no further *interesting* property is derived. A new property is classified as *interesting* by the algorithm if it is either already discovered but with a coefficient strictly smaller than the new one, or it is not repeated and its coefficient is greater than a fixed threshold value.

Algorithm for extracting synonymy properties

```
Input:  database schemes, basic synonymy properties
Output:  derived synonymy properties.
Begin
    Extract_Candidates()
    For each pair of candidates do
        Extract_direct_identical();
        Extract_direct_basic_synonyms()
    repeat
        Extract_identical();
        Extract_basic_synonyms();
        Extract_simple_synonyms()
    until at least a valid property is extracted
end
```

The synonymy coefficient associated to derived properties, V, is defined as follows:

$$V = Round\left(\frac{n_u \times th_u + \left(\sum_{i=1}^{ngn} n_{ni} \times V_{ni}\right)}{n_u + \sum_{i=1}^{ngn} n_{ni} + \sum_{i=1}^{ngs} n_{si} + n_d} + \frac{\left(\sum_{i=1}^{ngs} n_{si} \times V_{si}\right) + n_d \times th_d}{n_u + \sum_{i=1}^{ngn} n_{ni} + \sum_{i=1}^{ngs} n_{si} + n_d}\right)$$

where th_u and th_d are thresholds identifying conditions of equality and distinctness, respectively; n_u and n_d are the number of equal and distinct synonymy properties between classes in the considered neighborhood, whereas n_{si} is the number of synonymy properties whose coefficient is V_{si} and, finally, n_{ni} is the number of basic synonymy properties whose coefficient is V_{ni}.

More details about the synonymy algorithm and the corresponding one for discovering homonymies can be found in (La Camera et al. (1997)).

Example 3.1 Consider the following schemes representing the Production (denoted SP) and Administration (denoted SO) Departments of an organization, respectively:

Suppose that experts provided in input the following synonymy properties:

$\langle \texttt{Chief}_{[SP]}, \texttt{Manager}_{[SO]}, 0.8\rangle_{Syn}$

$\langle \texttt{Department}_{[SP]}, \texttt{Division}_{[SO]}, 0.9\rangle_{Syn}$

Our synonymy algorithm will discover the following synonymy properties and associated factors:

$\langle \texttt{Town}_{[SP]}, \texttt{BirthPlace}_{[SO]}, 1\rangle_{Syn}$

$\langle \texttt{Subordinate}_{[SP]}, \texttt{Employee}_{[SO]}, 0.67\rangle_{Syn}$

$\langle \texttt{Operates}_{[SP]}, \texttt{Works}_{[SO]}, 0.79\rangle_{Syn}$

$\langle \texttt{Born}_{[SP]}, \texttt{Born}_{[SO]}, 1\rangle_{Syn}$

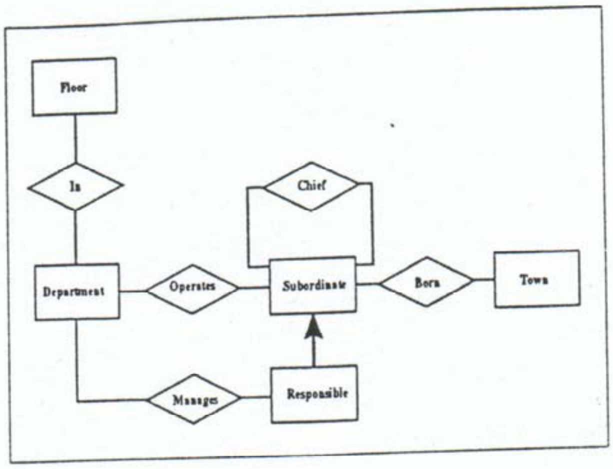

Fig. 1 Scheme SP: the Production Department database

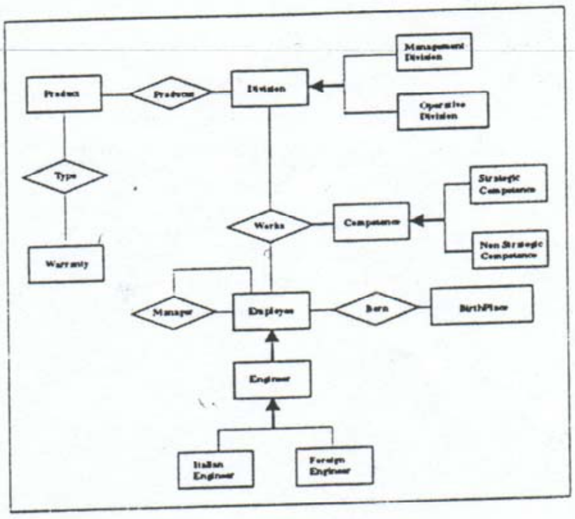

Fig. 2 Scheme SO: the Administration Department database

3.2 Deriving further interscheme knowledge

Step 2 of the preprocessing phase consists in deriving further interscheme knowledge. In this phase, an algorithm is applied which enriches the content of three dictionaries: synonymy and homonymy dictionaries, which have been populated using the algorithms discussed above, and the so called Inclusion Dictionary, which stores PDL assertions stating inclusion properties between database classes.

This latter dictionary is initially filled using a (possibly empty) set of basic inclusion properties provided by database experts. This initial set of inclusion properties can be also derived by running appropriate aggregate queries over input databases. New inclusion and new synonymy properties can be derived. Derived properties are either classified into exact or conditional. *Exact properties* are those which are always valid while *conditional properties* are those whose validity depends on the values of the involved coefficients. We will explain the extraction of exact properties and of conditional ones using two examples.

Suppose that $\langle C, A, W_{CA}\rangle_{Inc}$ and $\langle A, B, W_{AB}\rangle_{Inc}$ are stored in the inclusion dictionary. Then we deduce $\langle C, B, \tau(W_{CA}, W_{AB})\rangle_{Inc}$ where $\tau(\alpha, \beta)$ is a t_norm (Fankhauser et al. (1991)) which is instantiated depending on the specific application; t_norms are two place functions from [0,1] $\times$ [0,1] to [0,1]; they are monotonic, commutative and associative and have been used to define fuzzy set intersection.

As for the second case suppose that there are two schemes S_1 and S_2 and suppose that, in the scheme S_1, B is a subset of A with inclusion coefficient W_{BA} while, in the scheme S_2, D is a subset of C with coefficient W_{DC}. Suppose that the inclusion dictionary stores that A is a subset of C with inclusion coefficient W_{AC} and synonymy coefficient V_{AC}; then, **if** $W_{DC} \gg \tau(W_{BA}, W_{AC})$ **then** it is possible to deduce that almost all the instances of C are also instances of D; so we can deduce that B is a subset of D with inclusion coefficient $W_{BD} = \tau(\tau(W_{BA}, W_{AC}), W_{DC})$ and synonymy coefficient $V_{BD} = V_{AC}$. This property is conditional because it is true only if $W_{DC} \gg \tau(W_{BA}, W_{AC})$.

Example 3.1[continued...] Suppose that the following inclusion properties are provided by the experts (or extracted from the databases using suitable aggregate queries):

$\langle \texttt{Responsible}_{[SP]}, \texttt{Subordinate}_{[SP]}, 0.3\rangle_{Inc}$
$\langle \texttt{OperativeDivision}_{[SO]}, \texttt{Subordinate}_{[SO]}, 0.7\rangle_{Inc}$
$\langle \texttt{Employee}_{[SO]}, \texttt{Dependent}_{[SP]}, 0.7\rangle_{Inc}$
$\langle \texttt{Engineer}_{[SO]}, \texttt{Employee}_{[SO]}, 0.5\rangle_{Inc}$
$\langle \texttt{Manager}_{[SO]}, \texttt{Chief}_{[SP]}, 0.95\rangle_{Inc}$
$\langle \texttt{NonStrategicCompe}_{[SO]}, \texttt{Competence}_{[SO]}, 0.6\rangle_{Inc}$
$\langle \texttt{BirthPlace}_{[SO]}, \texttt{Town}_{[SP]}, 0.95\rangle_{Inc}$
$\langle \texttt{ManagementDivision}_{[SO]}, \texttt{Division}_{[SO]}, 0.3\rangle_{Inc}$
$\langle \texttt{StrategicCompe}_{[SO]}, \texttt{Competence}_{[SO]}, 0.4\rangle_{Inc}$
$\langle \texttt{Division}_{[SO]}, \texttt{Department}_{[SP]}, 0.95\rangle_{Inc}$
$\langle \texttt{ItalianEngineer}_{[SO]}, \texttt{Engineer}_{[SO]}, 0.9\rangle_{Inc}$
$\langle \texttt{ForeignEngineer}_{[SO]}, \texttt{Engineer}_{[SO]}, 0.1\rangle_{Inc}$

By using the set of already derived nominal properties, the algorithm discovers the following new synonymy and inclusion properties:

$\langle \texttt{Engineer}_{[SO]}, \texttt{Subordinate}_{[SP]}, 0.67\rangle_{Syn}$
$\langle \texttt{Responsible}_{[SP]}, \texttt{Employee}_{[SO]}, 0.67\rangle_{Syn}$
$\langle \texttt{Engineer}_{[SO]}, \texttt{Subordinate}_{[SP]}, 0.35\rangle_{Inc}$
$\langle \texttt{Responsible}_{[SP]}, \texttt{Employee}_{[SO]}, 0.21\rangle_{Inc}$

Inferred properties must be checked for possible conflicts, so a validation phase takes place. This is conducted in part automatically but may also require the intervention of experts.

In particular, some derived properties may result in contradiction with some belief of some expert, in which case a negotiation phase takes place between the system and the expert. During this phase the system provides inference tracing justifying the deduction of the contradictory piece of information and the expert supplies further information either validating or modifying or rejecting the assertions included in the inference tracing.

4 Phase 2: Discovering Complex Structural Properties

The second phase extracts properties involving, in general, complex class expressions. The extraction of such complex properties consists of two steps: *Step 1*, where most interesting classes are singled out on the basis of a weight assigned by an algorithm which uses dictionaries constructed in the preprocessing phase; *Step 2*, in which new properties involving interesting classes identified in Step 1 are derived.

4.1 Singling out most interesting classes

This step uses the information stored in the synonymy and inclusion dictionaries. The process consists in associating an interest weight, denoted by the function $\sigma(.)$, to entities and relationships. The interest coefficient associated to relationships is defined as $\sigma_{R_i} = \sum_{i=1}^{n} Val(E_j)$, where the E_j's represent entities directly connected to R_i. The function $Val(E_j)$ returns a value encoding the "local" interest of the class E_j and is defined as $Val(E_j) = \sum_{k=1}^{l} Max(W_{E_j E_k}, V_{E_j E_k})$, where l is the number of objects related in the dictionaries to entity E_j by a synonymy property, an inclusion property or both. The underlying assumption here is that the more a class is involved in properties appearing in dictionaries with high factors, the more probable is it will be used to extract new interesting properties from schemes.

In order to single out interesting relationships, an interest threshold value relative to relationships is used. This is defined as $Th_{\sigma_R} = \frac{Min(\sigma_R)+Max(\sigma_R)}{D_R}$, where $Min(\sigma_R)$ and $Max(\sigma_R)$ denote the minimum and the maximum interest coefficient we have computed, and D_R is a normalization factor used to tune up the threshold: the smaller its value is the more selective the threshold will be. All relationships with interest coefficient greater then Th_{σ_R} are considered interesting. Interesting entities are singled out in an analogous way.

Now, let R_i be an interesting relationship. Let C_1 ,..., C_n be the entities connected to R_i. Our method proceeds by tentatively populate the inclusion dictionary with the following set of assertions:

$$\exists R_i[T_1].T_2 : C_2, ..., T_n : C_n \leq C_1$$
$$\forall R_i[T_1].T_2 : C_2, ..., T_n : C_n \leq C_1$$
$$\dots$$

$$\exists R_i[T_n].T_1 : C_1, ..., T_{n-1} : C_{n-1} \dot{\leq} C_n$$
$$\forall R_i[T_n].T_1 : C_1, ..., T_{n-1} : C_{n-1} \leq C_n$$

Note that these are assertions involving objects belonging to a single scheme. In order to associate them with proper inclusion coefficients, we proceed by first asking the database expert to state which of these assertions are meaningful and then by submitting a suitable aggregate query to get the coefficient associated to meaningful assertions (note that both the left-hand and the right-hand sides of the assertions above correspond to simple queries). Resulting PDL assertions are stored in the inclusion dictionary. An example follows.

Example 3.1[continued...] From entities and relationships of the previous example the algorithm singles out as interesting the objects $\texttt{Works}_{[SO]}$, $\texttt{Born}_{[SP]}$, $\texttt{Born}_{[SO]}$, $\texttt{Operates}_{[SP]}$, $\texttt{Produces}_{[SO]}$ e $\texttt{Manages}_{[SP]}$ and entities $\texttt{ForeignEngineer}_{[SO]}$, $\texttt{ItalianEngineer}_{[SO]}$, $\texttt{Engineer}_{[SO]}$, $\texttt{OperativeDivision}_{[SO]}$, $\texttt{Competence}_{[SO]}$, $\texttt{Division}_{[SO]}$, $\texttt{Responsible}_{[SP]}$, $\texttt{Subordinate}_{[SP]}$, $\texttt{Employee}_{[SO]}$, as interesting.

Two of the assertions tentatively selected to populate the dictionaries are:

$$\exists\texttt{Born[NL].IN} : \texttt{Employee}_{[SO]} \leq \texttt{BirthPlace}_{[SO]}$$
$$\exists\texttt{Born[IN].NL} : \texttt{BirthPlace}_{[SO]} \dot{\leq} \texttt{Employee}_{[SO]}$$

Note that, while the former is semantically meaningful, the latter is not. As a consequence, the latter is discarded.

4.2 Extraction of complex interscheme properties

In this section, Step 2 of the intensional knowledge discovery process is illustrated. As already stated, the general form of PDL assertions extracted by our method correspond to formulae $L_1 \leq L_2$ (see Section 2), to which an inclusion factor f is associated to form a triplet $\langle L_1, L_2, f\rangle_{Inc}$. Here, both L_1 and L_2 are class expressions. The preprocessing phase extracts properties where both L_1 and L_2 are simple entity symbols. The method we are presenting next derives properties involving more complex expressions. The method works by case analysis: each of the following subsections will be devoted to the illustration of one of such inference cases.

4.2.1 Expressions containing $\sqcap$ and $\sqcup$

Here, we consider the case in which L_1 is an intersection or a union between two entities. In order to illustrate the foregoing, assume the following properties were extracted in the preprocessing phase:

$$\langle A,\ C,\ W_{AC}\rangle_{Inc} \quad \langle B,\ C,\ W_{BC}\rangle_{Inc}$$

To establish the inclusion coefficient of the assertions:

$$(A \sqcap B) \dot{\leq} C \quad (A \sqcup B) \dot{\leq} C$$

we reason as follows; first of all, note that there are two extreme situations to be considered:

- A and B are included either ways into one another, in which case the derived coefficients are:
$$W_{(A\sqcap B,C)} = min(W_{AC}, W_{BC})$$
$$W_{(A\sqcup B,C)} = max(W_{AC}, W_{BC})$$
- A and B have minimal intersection, i.e. their symmetric difference has the greatest value, in which case we derive the following coefficients:
$$W_{(A\sqcap B,C)} = max(0, W_{AC} + W_{BC} - 1)$$
$$W_{(A\sqcup B,C)} = min(1, W_{AC} + W_{BC})$$

To derive the general coefficient form, we choose to compute the mean value between those extremal ones; thus, derived coefficients will be:

$$W_{(A\sqcap B,C)} = \frac{min(W_{AC},W_{BC})+max(0,W_{AC}+W_{BC}-1)}{2}$$
$$W_{(A\sqcup B,C)} = \frac{max(W_{AC},W_{BC})+min(1,W_{AC}+W_{BC})}{2}$$

Example 3.1[continued...] As shown above, the preprocessing phase has yielded the following properties:

$\langle\texttt{Engineer}_{[SO]},\ \texttt{Subordinate}_{[SP]}\ 0.35\rangle_{Inc}$
$\langle\texttt{Responsible}_{[SP]},\ \texttt{Subordinate}_{[SP]},\ 0.3\rangle_{Inc}$

By applying the rule described above, the following properties are derived:

$\langle\texttt{Engineer}_{[SO]} \sqcap \texttt{Respon.}_{[SP]}, \texttt{Subordinate}_{[SP]},\ 0.15\rangle_{Inc}$
$\langle\texttt{Engineer}_{[SO]} \sqcup \texttt{Respon.}_{[SP]}, \texttt{Subordinate}_{[SP]},\ 0.5\rangle_{Inc}$

4.2.2 Expressions containing $\forall$ and $\exists$

Let R be an interesting relationship. Assume R is connected to entity C_1 through role T_1 and to the entity E through role U. Assume that the assertion $\forall R[U].T_1 : C_1 \leq E$ is meaningful. Assume, moreover, the inclusion dictionary stores the assertion $\langle\forall R[U].T_1 : C_1, E, W_{C_1\forall E}\rangle_{Inc}$. The algorithm considers several cases, which are illustrated next:

- **There exists a subset property between an expression E' and the entity C_1 (Case A)**

  Suppose that the property $\langle E', C_1, W_{E'C_1}\rangle_{Inc}$ is stored in the inclusion dictionary, where E' is any expression denoting a subset of C_1. From this, we infer that $\langle\forall R[U].T_1 : E', E, W_{E'\forall E}\rangle_{Inc}$ where $W_{E'\forall E} = W_{C_1\forall E} \times (W_{E'C_1})^{\mu}$ and μ is the average number of fillers of the role T_1 in R.

  As usual the difficult part of the derivation process is associating an appropriate coefficient to the assertion. Such computation is based on the following observations:

(i) $\langle \forall R[U].T_1 : C_1, E, W_{C_1 \forall E}\rangle_{Inc}$ corresponds to the set of instances of E connected to R and associated, through role T_1, only to instances of C_1; *(ii)* $\langle \forall R[U].T_1 : E', E, W_{E' \forall E}\rangle_{Inc}$ denotes all instances of E connected to the relation R and associated, through role T_1, only to instances of E'[2]; *(iii)* say μ is the average number of instances relative to the role T_1 in R. Then, all these μ instances must belong to E'. The probability for this to happen is $(W_{E'C_1})^{\mu}$.

The derived coefficient is then $W_{E' \forall E} = W_{C_1 \forall E} \times (W_{E'C_1})^{\mu}$. We note that the factor μ can be easily provided by experts.

- **The entity F includes the entity E (Case B)**

  Assume that the inclusion dictionary stores the assertion $\langle E, F, W_{EF}\rangle_{Inc}$. Then, the algorithm infers $\langle \forall R[U].T_1 : C_1, F, W_{C_1 \forall E} \times W_{EF}\rangle_{Inc}$.

- **R has more than one role (case C)**

  Assume we have already derived the following assertions:

  $$\langle \forall R[U].T_1 : C_1, E, W_{C_1 \forall E}\rangle_{Inc}$$
  $$\langle \forall R[U].T_2 : C_2, E, W_{C_2 \forall E}\rangle_{Inc}$$

  where T_1 and T_2 are roles of the same relation R. Suppose we want to determine the inclusion coefficient associated to the assertion $\langle \forall R[U].T_1 : C_1, T_2 : C_2, E, W_{expr}\rangle_{Inc}$. Note that PDL expressions such as the one above, where two roles occur in the selection part, are equivalent to intersection expressions, as those treated is subsection 4.2.1 and, therefore, we can use the reasoning shown there to derive that: $W_{expr} = \frac{min(W_{C_1 \exists E}, W_{C_2 \exists E})}{2} + \frac{max(0, W_{C_1 \exists E} + W_{C_2 \exists E} - 1)}{2}$

- **E is a complex expression (Case D)**

  Suppose E is a complex expression (i.e., not an entity symbol) and A and C are simple or complex expressions. Suppose the following properties have been already derived:

  $$\langle A,\ C,\ W_{AC}\rangle_{Inc} \quad \langle E,\ C,\ W_{EC}\rangle_{Inc}$$

  If $W_{AC} < W_{EC}$, the plausibility coefficient corresponding to the expression $\langle A, E, W_{AE}\rangle$ can be evaluated. For determining W_{AE}, we again analyze two extreme situations:

  - the subsets A and E have minimal intersection, in which case W_{AE} is:

    $$W_{AE} = \frac{max(0, W_{AC} + W_{EC} - 1)}{max(W_{AC}, W_{EC})}$$

  - the subsets A and E are included into one another in either way, in which case we infer:

    $$W_{AE} = \frac{min(W_{AC}, W_{EC})}{max(W_{AC}, W_{EC})}$$

  We proceed as we have done before and take the average between the two values above as the value for W_{AE}:

  $$W_{AE} = \frac{max(0, W_{AC} + W_{EC} - 1) + min(W_{AC}, W_{EC})}{2 \times max(W_{AC}, W_{EC})}$$

Example 3.1[continued...] Suppose we have the following properties:

$\langle \forall \mathtt{Works[DL].IL} : \mathtt{Engineer, Division, 0.4}\rangle_{\mathtt{Inc}}$
$\langle \forall \mathtt{Works[DL].IL} : \mathtt{Subordinate, Division, 0.5}\rangle_{\mathtt{Inc}}$
$\langle \forall \mathtt{Works[DL].IL} : (\mathtt{Engineer} \sqcap \mathtt{Employee}), \mathtt{Division, 0.4}\rangle_{\mathtt{Inc}}$
$\langle \forall \mathtt{Works[DL].LC} : (\mathtt{StratCompe} \sqcap \mathtt{Competence}), \mathtt{Division, 0.7}\rangle_{\mathtt{Inc}}$
$\langle \exists \mathtt{Chief[DC].CD} : \mathtt{Subordinate, Subordinate, 0.4}\rangle_{\mathtt{Inc}}$

then, we can infer:

$\langle \forall \mathtt{Works[DL].IL} : (\mathtt{Engineer} \sqcap \mathtt{ItalianEngineer}), \mathtt{Division, 0.2916}\rangle_{\mathtt{Inc}}$
$\langle \forall \mathtt{Works[DL].IL} : (\mathtt{Engineer} \sqcap \mathtt{ItalianEngineer}), \mathtt{Department, 0.277}\rangle_{\mathtt{Inc}}$
$\langle \forall \mathtt{Works[DL].LC} : (\mathtt{StratCompe} \sqcap \mathtt{Competence}), \mathtt{IL} : (\mathtt{Engineer} \sqcap \mathtt{Employee}), \mathtt{Division, 0.25}\rangle_{\mathtt{Inc}}$
$\langle \forall \mathtt{Works[DL].IL} : (\mathtt{Engineer} \sqcap \mathtt{Employee}), \forall \mathtt{Works[DL].LC} : (\mathtt{StratCompe} \sqcap \mathtt{Competence}), \mathtt{0.357}\rangle_{\mathtt{Inc}}$

The corresponding expressions containing $\exists$ in the place of $\forall$ lead to analogous result.

4.2.3 Negation In DL, negation of a class represents all the instances of the domain which are not instances of that class. In order to preserve evaluation safety, we shall avoid computing complements w.r.t. the entire domain and, therefore, we shall evaluate negation by intersection with one of the superset classes. Thus, it is possible to derive properties for negation of a class only if there exists an inclusion property relative to that class. Assume, then, that the inclusion dictionary includes the triplet $\langle B, A, W_{BA}\rangle_{Inc}$. Then, our algorithm infers $\langle \neg B, A, (1 - W_{BA})\rangle_{Inc}$. Once derived, the coefficient for $\neg B \leq A$, $\neg B$ can be handled as any other subset of A for the purpose of deriving other complex properties.
This closes the illustration of our intensional knowledge discovery approach.

[2] Differently from case (i) above, here we do not consider instances of E connected to R associated, through role T_1, to an instance of C_1 which is not an instance of E'.

5 Applications

PDL assertions describing intensional knowledge concerning sets of database schemes have many applications, and all of them benefit from methods for automatic derivation of such assertions, as the ones we have presented in this paper. Applications include (1) the design of information integration layers on top of existing database systems, such as mediators (Ullman (1997), Levy et al. (1996)), and more in general the development of tools for supporting integrated access to cooperative information systems (Papazoglou et al. (1992), Catarci - Lenzerini (1993)), (2) query optimization and view maintenance (Chaudhuri et al. (1995)), (3) structuring and maintenance of warehouses and constraints (Gupta et al. (1995)).

Next, we describe simple examples in the area of pre-existing database systems integration, query optimization and constraints.

Example 3.1[continued...] PDL assertions, deduced so far, can be used by a specific integration algorithm. In (La Camera et al. (1997)) is shown how the integration of schemes SP and SO of the example can be performed to obtain a global scheme.

Example 5.1 Suppose we have a generic scheme S. Suppose the process of IKDD extracted the following assertions:

$\langle \exists R[U].T_1 : C_1, A, W_{Expr1} \rangle_{Inc}$
$\langle \exists R[U].T_2 : C_2, B, W_{Expr2} \rangle_{Inc}$
$\langle A, C, W_{AC} \rangle_{Inc}$
$\langle B, \neg C, W_{A \neg C} \rangle_{Inc}$

Now, assume that we want to evaluate the query $Q = \exists R[U].T_1 : C_1, T_2 : C_2$ (i.e., we want all the fillers of role U in R such that roles T_1 and T_2 are instantiated with instances of C_1 and C_2 only, resp.). In this case, the empty answer can be returned without actually executing Q over the underlying database. Using similar ideas we can also reduce some queries to simpler and equivalent forms.

A second application domain for Intentional Knowledge Discovery in Databases processes is materialized view structuring, i.e., given a database, we want to decide which views to materialize for the purpose of optimized query answering. We propose an algorithm that decide which views to materialize. This algorithm is based on the following observations. PDL formulae naturally correspond to queries, and then to views, as already pointed out. Consider the assertions stored in the inclusion dictionary. Note that the greater the number of inclusion properties in which a given expression E appears as a superset is, the higher the probability of answering queries exploiting the view corresponding to E is. For a fixed given number of inclusion properties in which E appears, the greater inclusion coefficients associated to E are, the more convenient to materialize the view corresponding to E is. Therefore, we can define a view coefficient M for each formula that appears as superset in the inclusions dictionary, as follows: $M(E) = N_{Incl} \times \beta + \sum_{i=1}^{N_{Incl}} W_{Expr_i}$ where N_{Incl} is the number of inclusions where E appears as the superset, W_{Expr_i} represent the associated inclusion coefficients and β is a factor used for tuning up the coefficient.

After determining the view coefficient of each expression, we set a view threshold: $ThM = \frac{M_{MAX}+M_{MIN}}{D_M}$ where M_{MAX} is the greatest view coefficient we have computed, whereas M_{MIN} is the smallest one and, as usual, D_M is a tuning-up factor. Then, views corresponding to the expressions whose view coefficient is greater than the threshold will be materialized.

REFERENCES

S. Abiteboul, Querying Semi-Structured Data, Proc. ICDT, 1997, 1-18.

C. Batini, S. Castano, V. De Antonellis, M.G.Fugini, B. Pernici, Analysis of an Inventory of Information Systems in the Public Administration, Requirement Engineering Journal, 1(1), 47-62, 1996.

S. Castano, V. De Antonellis, Semantic Dictionary Design for Database Interoperability, Proceedings of ICDE'97, Birmingham, 1997.

T. Catarci, M. Lenzerini, Representing and using interschema knowledge in cooperative information systems, Journal of Intelligent and Cooperative Information Systems, 2(4), 375-398, 1993.

S. Chaudhuri, R. Krishnamurthy, S. Potamianos, K. Shim, Optimizing queries with materialized views, Proc. ICDE, 1995.

U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, Advances in Knowledge Discovery and Data Mining, The AAAI - The MIT press 1996

P. Fankhauser, M. Kracker, E.J.Neuhold, Semantic vs. Structural Resemblance of Classes, SIGMOD RECORD, 20(4), 1991, 59-63

A. Gupta, I.S. Mumick, K.A. Ross, Adapting materialized views after redefinitions. Proc. ACM SIGMOD, 1995

M. La Camera, L. Palopoli, S. Rotundo, D. Saccá and D. Ursino, Discovering properties for database schemes to synthesize an integrated, abstract dictionary, 1997. Manuscript. Submitted for publication.

A. Levy, A. Rajaraman, J. Ordille, Querying heterogeneous information sources using source descriptions, Proc. VLDB, 1996.

M.P. Papazoglou, S.C. Laufmann, T.K. Sellis, An organizational framework for cooperative information systems, Journal of Intelligent and Cooperative Information Systems, 1(1), 1992.

J.D. Ullman, Information integration using logical views, Proc. ICDT, 1997, 19-40.