Semantic Integration and Query of Heterogeneous Information Sources*

S. Bergamaschi[1] S. Castano[2] M. Vincini[1]
D. Beneventano[1]
[1] University of Modena and Reggio Emilia
DSI - Via Campi 213/B
41100 Modena
[2] University of Milano
DSI - Via Comelico, 39
20135 Milano
e-mail: {sonia,vincini,benevent}@dsi.unimo.it castano@dsi.unimi.it

Abstract

Developing intelligent tools for the integration of information extracted from multiple heterogeneous sources is a challenging issue to effectively exploit the numerous sources available on-line in global information systems. In this paper, we propose intelligent, tool-supported techniques to information extraction and integration from both structured and semistructured data sources. An object-oriented language, with an underlying Description Logics, called ODL$_{I^3}$, derived from the standard ODMG is introduced for information extraction. ODL$_{I^3}$ descriptions of the source schemas are exploited first to set a shared vocabulary for the sources. Information integration is then performed in a semi-automatic way, by exploiting ODL$_{I^3}$ descriptions of source schemas with a combination of Description Logics and clustering techniques and gives rise to a virtual integrated view of multiple sources. As the ultimate goal of providing an integrated view is querying the view, independently from the location/heterogeneity of the sources, a module for the reformulation of queries at the sources with semantic optimization capabilities is provided. Integration techniques described in the paper have been implemented in the MOMIS system, based on a conventional mediator architecture.

*This research has been partially funded by the *italian MURST ex-40% INTERDATA project - Metodologie e Tecnologie per la Gestione di Dati e Processi su Reti Internet e Intranet.* A preliminary version of the paper appears in the proceedings of IJCAI-99 Workshop on Intelligent Information Integration 31 July 1999, Stockholm.

Keywords - Information Extraction, Information Integration, Semistructured Data, Semantic Heterogeneity, Description Logics, Clustering Techniques.

1 Introduction

Developing intelligent tools for the integration of information extracted from multiple heterogeneous sources is a challenging issue to effectively exploit the numerous sources available on-line in global, Internet-based information systems. Main problems to be faced are related to the identification of semantically related information (that is, information related to the same real-world concept in different sources), and to semantic heterogeneity. In fact, information sources available on-line in global information systems already exist and have been developed independently. Consequently, semantic heterogeneity can arise for the aspects related to terminology, structure, and context of the information, and has to be properly dealt with in order to effectively use the information available at the sources.

Integration and reconciliation of data coming from heterogeneous sources is a hot research topic in databases. Several contributions appeared in the recent literature, including methods, techniques and tools for integrating and querying heterogeneous databases. The integration of semi-structured and unstructured data sources presents new problems and challenges: in this case, the heterogeneity concern not only the semantics of data, but also the degree by which the structure of data is explicitly represented in the sources. The significant growing of semi-structured data sources (document, texts, etc.) calls for the design of methods and techniques for this new type of data integration. Thus, the typical problems of integration should be addressed in the light of these new requirements.

The goal of information extraction and integration techniques is to construct synthesized, uniform descriptions (i.e. *a global virtual view*) of the information of multiple heterogeneous sources, to provide the user with a uniform query interface against the sources independent from the location and heterogeneity of the data at the sources. Any integration system that allows for a mechanisms of querying a global virtual view must contain a module for the reformulation of queries in terms of data stored in the sources and for the optimization of gloabal querying process. This problem is known in the literature as query rewriting and query answering using views, and has been studied very actively in the recent years. Moreover, to meet the requirements of global, Internet-based information systems, it is important to develop tool-based techniques, to make information extraction and integration activities semi-automatic and scalable as much as possible.

In this paper, we focus on capturing and reasoning about semantic aspects of schema descriptions of heterogeneous information sources for supporting integration and query optimization. Both semistructured and struc-

tured data sources are taken into account [10].

We propose intelligent, tool-supported techniques to information extraction and integration; an object-oriented language is introduced, called ODL_{I^3}, derived from the standard ODMG, with the underlying Description Logics OLCD (Object Language with Complements allowing Descriptive cycles) [5, 11], derived from the KL-ONE family [45]. Information extraction has the goal of representing source schemas in ODL_{I^3}. In case of semistructured information sources, information extraction produces also object patterns, to be used as schema information for the source to generate the corresponding ODL_{I^3} description.

ODL_{I^3} descriptions of the information sources are exploited to set a shared ontology for the sources, in form of a Common Thesaurus, by exploiting the OLCD Description Logics inference capabilities. Information integration has the goal of producing global, integrated ODL_{I^3} descriptions of the sources. It is performed in a semi-automatic way, by exploiting ODL_{I^3} descriptions of source schemas and by combining clustering techniques and the OLCD Description Logics on the Common Thesaurus. Furthermore, the WordNet lexical system [36] is used to automatically extract inter-sources terminological relationships. Mapping rules are defined at the global level to express the relationships holding between obtained ODL_{I^3} integrated descriptions and ODL_{I^3} sources descriptions, respectively.

Techniques described in the paper have been implemented in the MOMIS (Mediator envirOnment for Multiple Information Sources) system, conceived as a joint collaboration between University of Milano and University of Modena and Reggio Emilia in the framework of the INTERDATA national research project. This project aims at providing methods and tools for data management in Internet-based information systems. Like other integration projects [35, 18, 3], MOMIS follows a "semantic approach" to information integration based on the conceptual schemas of the information sources, and a mediator and query-processing component, based on two pre-existing tools, namely ARTEMIS [19] and ODB-Tools [6], to provide an I^3 architecture for integration and query optimization.

The architectural elements are the following: i) the data model and ODL_{I^3} have been defined in MOMIS as subset of the ODMG [24] ones, following the proposal for a standard mediator language developed by the I^3/POB working group. In addition, ODL_{I^3} introduces new constructors to support the semantic integration process [9, 10]; ii) one or more wrappers, to translate schema descriptions into the common ODL_{I^3} representation; iii) a mediator and a query-processing component, based on two pre-existing tools, namely ARTEMIS [22] and ODB-Tools (available on Internet at

http://sparc20.dsi.unimo.it/), to provide an I^3 architecture for integration and query optimization.

The paper is organized as follows. In Section 2, we give basic notations and assumptions regarding structured and semistructured data modelling and management and we introduce the ODL_{I^3} language, the OLCD Description Logics and its inference techniques. In Section 3, we describe the use of OLCD to provide a shared ontology and to build a Common Thesaurus of terminological relationships describing common knowledge about local sources. In Section 4, we describe the integration process for building the mediator global schema (i.e. the virtual global view), exploiting affinity-based clustering techniques for the formulation of groups of classes with affinity and ODL_{I^3} mapping rules. In Section 5 we describe the MOMIS architecture. In Section 6 we describe query processing and optimization. In Section 7, we discuss previous work on semistructured data modeling and heterogeneous information integration. Finally, in Section 8 we give our concluding remarks.

2 Information extraction with ODL_{I^3}

The first step in information extraction is the construction of a semantically rich representation of the information sources to be integrated by means of a common data model. In semantic approaches to integration, this task is performed by considering the conceptual schema of the source.

This operation is performed by wrappers that are developed for each kind of data representation. For conventional structured information sources (e.g., relational databases, object-oriented databases), schema description is always available and can be directly translated into the selected common data model. For example, for flat files and object-oriented databases wrappers perform a syntactic translation, while for the relational databases they are based on *transformation rule-sets*, as described in [28] for relational to ODMG schema conversion.

For semistructured information sources (e.g., Web sources), schema description is generally not directly available in the sources. In fact, a basic characteristic of semistructured data is that they are "self-describing". This means that the information generally associated with the schema is specified directly within data.

One of the goals of information extraction for integration when semistructured information sources are involved is to derive and explicity represent

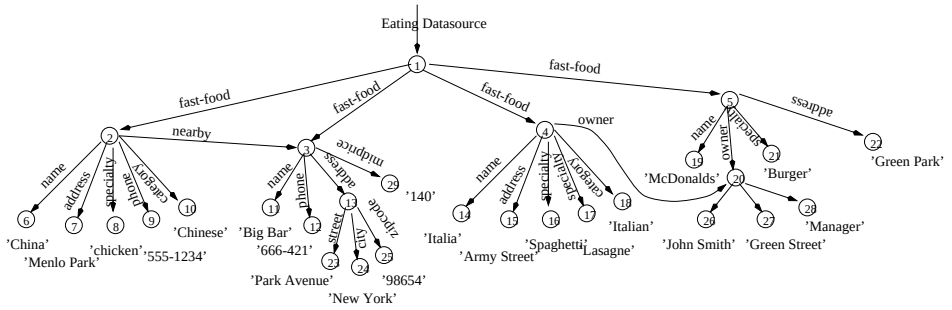

Figure 1: Eating Source (ED)

also the schema of the source. For this purpose, we proceed as follows. According to the models proposed in literature for semistructured information sources [13, 41], semistructured sources are represented as rooted, labeled graph with the semistructured data (e.g., an image or free-form text) as nodes and labels on edges. Figure 1 shows an example of semistructured source, containing information related to the `Eating Source` that collects information on local fast food. A semistructured object (object, for short) can be viewed as a triple of the form $\langle$`id`, `label`, `value`$\rangle$, where `id` is the object identifier, `label` is a string describing what the object represents, and `value` is the value, that can be atomic or complex. The atomic value can be integer, real, string, image, while the complex value is a set of semistructured objects, that is, a set of pairs (*id,label*). A complex object can be thought as the parent of all the objects that form its value (children objects). A given object can have one or more parents. We denote the fact that an object so' is a child object of another object so by $so \rightarrow so'$ and use notation $label(so)$ to denote the label of so. In semistructured data models, labels are descriptive as much as possible. Generally, the same label is assigned to all objects describing the same concept in a given source. To represent the schema of a semistructured source S, we introduce the notion of *object pattern*. All objects so of S are partitioned into disjoint sets, denoted set_l, such that all objects belonging to the same set have the same label l. An object pattern is then extracted from each set to represent all the objects in the set. Formally, an object pattern is defined as follows.

Definition 1 (Object pattern) *Let set_l be a set of objects in a semistructured source S having the same label l. The* object pattern *of set_l is a pair of the form $\langle l, A\rangle$, where l is the label of the objects belonging to set_l, and $A = \bigcup label(so')$ such that there exists at least one object $so \in set_l$ with*

Fast-Food-pattern = (Fast-Food,{name,address,midprice phone*,specialty,category,nearby*,owner*})
Owner-pattern = (Owner,{name,address,job})
Address-pattern = (Address,{street,city,zipcode})

Figure 2: The object patterns for the ED source

$so \rightarrow so'$.

From this definition, an object pattern is representative of all different objects that describe the same concept in a given semistructured source. In particular, l denotes the concept and set A the properties (or attributes) characterizing the concept in the source. Since semistructured objects can be heterogeneous, labels in A can be defined only for some of the objects in set_l, but not for all. We call such kind of labels "optional" and denote them with symbol "*".

Object patterns for all the objects in our semistructured source are shown in figure 2. Three object patterns are defined: `Fast-Food` containing information about eating places; `Owner` containing information about people involved and `Address`.

An object pattern description follows an *open world semantics* typical of the Description Logics approach [12, 45]. Objects conforming to a pattern share a common minimal structure represented by non optional properties, but can have further additional (i.e., optional) properties. In this way, objects in a semistructured data source can evolve and add new properties, but they will be retrieved as valid instances of the corresponding object pattern when processing a query.

2.1 Running example

Consider two sources in the Restaurant Guide domain that store information about restaurants. The `Eating Source` guidebook (`ED`) is semistructured and contains information about fast foods of the west coast, their menu, quality, and so on. Figure 1 illustrates a portion of the source. There is one complex root object with four complex children objects that represent fast-foods. Each `Fast-Food` has an atomic `name`, `category` and `specialty`. Furthermore, some `Fast-Food` objects have an atomic `address` and some other a complex `address`, a `phone`, a complex object `nearby`, that specifies the nearest fast-food, and `owner`, that indicates the `name`, the `address` and the `job` of the fast-food's owner.

Food Guide Database (FD)

```
Restaurant(r_code, name, street, zip_code, pers_id,
     special_dish, category, tourist_menu_price)
Bistro(r_code, type, pers_id)
Person(pers_id, first_name, last_name,
     qualification)
Brasserie(b_code, name, address)
```

Figure 3: Food Guide Database (FD)

The `Food Guide` Database (`FD`) is a relational database containing information about USA restaurants from a wide variety of publications (e..g newspaper reviews, regional guidebooks). There are four relations: `Restaurant`, `Bistro`, `Person`, and `Brasserie` (see figure 3). Information related to restaurants is maintained into the `Restaurant` relation. `Bistro` instances are a subset of `Restaurant` instances and give information about the small informal restaurants that serve wine. Each `Restaurant` and `Bistro` is managed by a `Person`. Information about places where drinks and snacks are served on are stored in the `Brasserie` relation.

2.2 The ODL$_{I^3}$ language

For a semantically rich representation of conceptual schemas and object patterns associated with information sources, we introduce an object-oriented language, called ODL$_{I^3}$. According to recommendations of ODMG and to the diffusion of I^3/POB, the object data model ODL$_{I^3}$ is very close to the ODL language. ODL$_{I^3}$ is a source independent language used for information extraction to describe heterogeneous information in a common way. ODL$_{I^3}$ introduces the following main extensions with respect to ODL:

Union constructor. The union constructor, denoted by `union`, is introduced to express alternative data structures in the definition of ODL$_{I^3}$ class, thus capturing requirements of semistructured data. An example of its use will be shown in the following.

Optional constructor. The optional constructor, denoted by (*), is introduced for class attributes to specify that an attribute is optional for an instance (i.e., it could be not specified in the instance). This constructor too has been introduced to capture requirements of semistructured data. An example of its use will be shown in the following.

Terminological relationships. They express inter-schema knowledge for the extracted source schemas. They are example of intensional assertions for the sources [23]. Terminological relationships are defined between classes and attributes, and are specified by considering class/attribute names, called terms. The following relationships can be specified in ODL$_{I^3}$:

- SYN (Synonym-of), defined between two terms t_i and t_j, with $t_i \neq t_j$, that are considered synonyms in every considered source (i.e., t_i and t_j can be indifferently used in every source to denote a certain concept).
- BT (Broader Terms), or hypernymy, defined between two terms t_i and t_j such as t_i has a broader, more general meaning than t_j.
- RT (Related Terms), or positive association, defined between two terms t_i and t_j that are generally used together in the same context in the considered sources.

Rules. Two kinds of rules are introduced in ODL$_{I^3}$: *if then* rules, to express in a declarative way integrity constraints intra and inter sources, and *mapping* rules, to express relationships holding between the integrated schema description of the information sources and the schema description of the original sources. These rules will be illustrated in detail in Section 4, together with examples of use.

As the result of the extraction process, object patterns and source schemas are translated into ODL$_{I^3}$ descriptions. Translation is performed by a wrapper. Moreover, the wrapper is also responsible for adding the source name and type (e.g., relational, semistructured). The translation into ODL$_{I^3}$, on the basis of the ODL$_{I^3}$ syntax (see Appendix A) and of the schema definition is performed by the wrapper as follows. Given a pattern $\langle l, A \rangle$ or a relation of a relational source, translation involves the following steps: i) an ODL$_{I^3}$ class name corresponds to l or to the relation name, respectively, and ii) for each label $l' \in A$ or relation attribute, an attribute is defined in the corresponding ODL$_{I^3}$ class. Furthermore, attribute domains have to be extracted. Structure extraction can be performed as proposed in [37, 14]

As an example, schemas of the `ED.Fast-Food` and `FD.Restaurant` sources are represented in ODL$_{I^3}$ as follows:

```
interface Fast-Food
 ( source semistructured Eating_Source )
```

```
{  attribute string      name;
   attribute string      address;
   attribute integer     phone*;
   attribute set<string> specialty;
   attribute string      category;
   attribute Restaurant  nearby*;
   attribute integer     midprice*;
   attribute Person      owner*;
   };

interface Restaurant
( source relational Food_Guide )
key r_code
foreign_key(pers_id) references Person )
{  attribute string         r_code;
   attribute string         name;
   attribute string         category;
   attribute string         street;
   attribute string         zip_code;
   attribute integer        pers_id;
   attribute integer        tourist_menu_price;
   attribute string         special_dish; };
```

Union and optional constructors are used with semistructured sources schemas to represent object patterns. In particular, the union constructor is used when in presence of heterogeneous object patterns. With reference to our semistructured source ED of figure 1, consider the semistructured object 2:
$so = \langle 2, \texttt{fast} - \texttt{food}, \{(6, \texttt{name}), (7, \texttt{address}), (8, \texttt{specialty}), (9, \texttt{phone}), (10, \texttt{category})\}\rangle$.
Consider the restaurant object 3, and in particular the non-atomic address object 13:
$so = \langle 13, \texttt{address}, \{(25, \texttt{street}), (26, \texttt{city}), (27, \texttt{zipcode})\}\rangle$.

In this case, because of the different structure of the Address object (i.e., in one case it is atomic while in the other case it is complex), pattern extraction produces a pattern also for address.

$Address - pattern = (\texttt{Address}, \{\texttt{city}, \texttt{street}, \texttt{zipcode}\})$.
To take into account this kind of heterogeneity, in ODL_{I^3} we define the special constructor **union** for a class. The specification in ODL_{I^3} of Address pattern is shown in figure 4. The semantics of the union constructor and of

optional attributes in ODL_{I^3} will be discussed in the next section, using the OLCD Description Logics.

```
interface Address
 ( source semistructured
      Eating_Source )
{  attribute string city;
   attribute string street;
   attribute string zipcode; };
 union
{  string; };
```

Figure 4: An example of union constructor

2.3 The OLCD Description Logics

ODL_{I^3} descriptions are translated into OLCD descriptions in order to perform inference tasks typical of Description Logics that will be useful for semantic integration and query processing, as will be illustrated in the remaining part of the paper.

In this section, we give an informal description of OLCD. Readers interested in a formal account can refer to [5]. OLCD is an extension of the *object description language* ODL (not to be confused with ODL-ODMG), introduced in [11] and holds usual type constructors of complex object data models. OLCD, as its ancestor ODL, provides a *system of base types*: string, boolean, integer, real; the type constructors *tuple*, *set* and *class* allow the construction of complex value types and class types. Class types (also briefly called *classes*) denote sets of *objects with an identity and a value*, while *value types* denote sets of *complex, finitely nested values without object identity.* In addition, an intersection operator can be used to create intersections of previously introduced types allowing simple and multiple inheritance specialization and an union operator ($\sqcup$) can be used to create unions of previously introduced types allowing to express generalization.

Finally, types can be given names. Named types come in two flavors: a named type may be *primitive* that means the user has to specify an *element*'s membership in the interpretation of the name or *virtual* and in such a case its interpretation is computed.

The extensions to ODL introduced in OLCD are: *quantified path types*, *integrity constraint rules* and *union* ($\sqcup$) operator. The first extension has been introduced to deal easily and powerfully with nested structures. Paths,

which are essentially sequences of attributes, represent the central ingredient of object-oriented query languages to navigate through the aggregation hierarchies of classes and types of a schema. In particular we provide *quantified* paths to navigate through set types. The allowed quantifications are existential and universal and they can appear more than once in the same path. A path type is a type associating a path to a type of the formalism. Therefore, by means of path types, we can express a class of integrity constraints.

The second extension allows the declarative expression of integrity constraints represented as *if then rule* universally quantified over the elements of the domain with an antecedent and a consequent which are types of the formalism.

The *union* ($\sqcup$) operator can be used to represent the semantics of the `union` constructor of ODL_{I^3}. It has been formalized in [5], with the meaning of the union of all possible union attribute instances.

For example, the `Address` pattern of figure 4 is translated in OLCD as follows:

$$\sigma_V(\texttt{Address}) = [\ \texttt{city} : \texttt{String}, \texttt{street} : \texttt{String}, \texttt{zipcode} : \texttt{String}\ ] \sqcup \texttt{String}$$

The *union* ($\sqcup$) operator is also useful to translate *optional* attributes into OLCD. In fact, an optional attribute `att` specifies that a value may exist or not for a given instance. This fact is described in OLCD as the union between the attribute specification with its domain and *attribute undefinedness*, denoted by $\uparrow$ operator: $([\texttt{att1} : \texttt{domain1}] \sqcup \texttt{att1}\uparrow)$

For our example, the `Restaurant` pattern can be represented as follows[1]:

$$\begin{aligned}\sigma_P(\texttt{Restaurant}) = \triangle\ \Big(\ & [\ \texttt{name} : \texttt{String}] \sqcap \\ & [\ \texttt{address} : \texttt{Address}\ ] \sqcap \\ & ([\ \texttt{phone} : \texttt{Integer}\ ] \sqcup \texttt{phone}\uparrow) \sqcap \\ & [\ \texttt{specialty} : \{\texttt{String}\}\ ] \sqcap \\ & [\ \texttt{category} : \texttt{String}\ ] \sqcap \\ & ([\ \texttt{nearby} : \texttt{Bar}\ ] \sqcup \texttt{nearby}\uparrow) \sqcap \\ & ([\ \texttt{owner} : \{\ \texttt{Owner}\ \}\ ] \sqcup \texttt{owner}\uparrow)\ \Big)\end{aligned}$$

Description Logics, and thus OLCD, permits, by exploiting virtual type semantics, and, given a *type as set* semantics to type descriptions, one to

[1] σ_P and σ_V represent the mapping between primitive type names and virtual type names and their structures respectively.

provide relevant reasoning techniques: computing *subsumption* relations between types (i.e. "isa" relationships implied by type descriptions), deciding *equivalence* between types and detecting *incoherent* (i.e., always empty) types.

As a subsumption example in the context of optional attributes, let us suppose to consider the value types `A` and `B`,

$\sigma_V(\texttt{A}) \quad = \quad [\ \texttt{att1} : \texttt{String}, \texttt{att2} : \texttt{String}\ ]$

$\sigma_V(\texttt{B}) \quad = \quad [\ \texttt{att1} : \texttt{String}] \sqcap ([\texttt{att2} : \texttt{String}] \sqcup \texttt{att2}\uparrow)$

By computing subsumption between types `A` and `B` we obtain that `B` subsumes `A` (A isa B) even if it has not been explicitely declared.

We developed a system, called ODB-Tools, based on OLCD and implementing the above reasoning techniques available on internet [6].

3 Reasoning about ODL$_{I^3}$ schema descriptions using OLCD and ODB-Tools

To develop intelligent techniques for semantic integration, a shared ontology for the information sources to be integrated is necessary. The ontology provides a reference vocabulary on which to base the identification of heterogeneities and the subsequent resolution for integration.

To provide a shared ontology for the sources, we exploit both the WordNet lexical system [36] and the Description Logics capabilities. We construct a *Common Thesaurus* of terminological relationships, describing common knowledge about ODL$_{I^3}$ classes and attributes of source schemas. ODL$_{I^3}$ descriptions and their internal representation into OLCD allow to discover terminological relationships from ODL$_{I^3}$ schema descriptions and reason about them, using inference techniques typical of Description Logics. The activity proceeds in the steps described below.

3.1 Automatic extraction of terminological relationships from ODL$_{I^3}$ schema descriptions

Terminological relationships represent synonymy (SYN), hypernymy (BT), hyponymy (NT) and positive associations (RT) between class and attribute names [36]. By exploiting ODB-Tools capabilities and semantically rich schema descriptions, an initial set of BT, NT, and RT can be automatically extracted. In particular, by translating ODL$_{I^3}$ into OLCD descriptions, ODB-Tools extracts BT/NT relationships among classes directly from generalization hierarchies, and RT relationships from aggregation hierarchies,

respectively. Other RT relationships are extracted from the specification of foreign keys in relational source schemas. When a foreign key is also a primary key both in the original and in the referenced relation, a BT/NT relationship is extracted; for example, see `Bistro` and `Restaurant` referring to the ODL_{I^3} descriptions of appendix B.

In case of semistructured sources, ODB-Tools extracts RT relationships, due to the nature of relationships defined in the semistructured data model.

Another set of relationships can be automatically extracted exploiting the WordNet [36] lexical system. In this case, synonyms, hypernyms/hyponyms, and related terms can be automatically proposed to the designer, by selecting them according to relationships predefined in the lexical system.

Example 1 Consider the `ED` and `FD` sources. The set of terminological relationships automatically extracted by ODB-Tools are the following:

⟨`ED.Fast-Food` RT `ED.Owner`⟩,
⟨`ED.Fast-Food` RT `ED.Address`⟩,
⟨`ED.Fast-Food` RT `ED.Fast-Food`⟩,
⟨`FD.Restaurant` RT `FD.Person`⟩,
⟨`FD.Restaurant` BT `FD.Bistro`⟩,
⟨`FD.Bistro` RT `FD.Person`⟩.

The relationships derived from WordNet are the following:

⟨`FD.Restaurant` BT `FD.Brasserie`⟩,
⟨`FD.Person` BT `ED.Owner`⟩,
⟨`ED.Owner.name` BT `FD.Person.first_name`⟩,
⟨`ED.Owner.name` BT `FD.Person.last_name`⟩.

3.2 Integration/Revision of relationships

New relationships can be supplied directly by the designer, to capture specific domain knowledge about the source schemas (e.g., new synonyms).
This is a very crucial operation, because the new relationships are forced to belong to the *Common Thesaurus* and thus used to generate the global integrated schema. This means that, if a nonsense or wrong relationship is inserted, the subsequent integration process may provide a wrong global schema.

Example 2 In our domain, the designer supplies the following terminological relationships for classes and attributes:
⟨`ED.Fast-Food` SYN `FD.Restaurant`⟩,
⟨`ED.Fast-Food.category` BT `FD.Bistro.type`⟩,
⟨`ED.Fast-Food.specialty` BT `FD.Bistro.special_dish`⟩.

Since terminological relationships are established for names, they can correlate ODL_{I^3} classes whose types present structural conflicts with respect to the semantics of *generalization* and *equivalence* relationships.
To exploit inference capabilities of Description Logics, we "promote" terminological relationships to the rank of semantic relationships, that is, SYN to equivalence, BT to generalization, and RT to aggregation. For this purpose, we need to solve structural conflicts producing an "ODL_{I^3} virtual schema" containing a restructured description of the extracted source schemas. In this way, the virtual schema can be used to enrich the Thesaurus with new relationships, by exploiting ODB-Tools inference techniques.

To promote a SYN relationship into a *valid equivalence relationship* it is necessary to "uniform" the types of both classes, that is, to give the same structure to both classes. The same problem arises for the BT relationship, whose transformation implies the addition of the attributes of the generalization class to the ones of the specialization class. Finally, when an RT relationship holds, a new aggregation attribute is defined between the two classes.

For example, consider the SYN relationship defined between the two classes (having different structures) `ED.Fast-Food` and `FD.Restaurant` (see Appendix B for their original structures). In order to translate this terminological relationship into a *valid equivalence relationship* for ODB-Tools it is necessary to "uniform" the types of both classes, i.e., to give the same structure to both classes. The resulting modified ODL_{I^3} classes are (the `Restaurant` description is the same):

```
interface Fast-Food
 (...)
{  attribute string      name;
   attribute Address     address;
   attribute integer     phone*;
   attribute set<string> specialty;
   attribute string      category;
   attribute Restaurant  nearby*;
```

```
attribute integer     midprice*;
attribute Person      owner*;
attribute string      r_code;
attribute string      street;
attribute string      zip_code;
attribute string      pers_id;
attribute integer     tourist_menu_price;
attribute string      special_dish;  };
```

The introduction by the designer of this new relationship, will lead to the discovery of a new relationship between `FD.Brasserie` and `ED.Fast-Food` and between `FD.Bistro` and `ED.Fast-Food` as will be shown in subsection 3.4.

3.3 Terminological Relationships validation

In this step, ODB-Tools is employed to validate terminological relationships between attribute names in the Thesaurus, by exploiting the virtual schema. Validation is based on the compatibility of domains associated with attributes. This way, *valid* and *invalid* terminological relationships are distinguished. In particular, let $a_t = \langle n_t, d_t \rangle$ and $a_q = \langle n_q, d_q \rangle$ be two attributes, with a name and a domain, respectively. The following checks are executed on terminological relationships defined for attribute names in the Thesaurus:

- $\langle n_t$ SYN $n_q \rangle$: the relationship is marked as valid if d_t and d_q are equivalent, or if one is a specialization of the other;
- $\langle n_t$ BT $n_q \rangle$: the relationship is marked as valid if d_t contains or is equivalent to d_q;
- $\langle n_t$ NT $n_q \rangle$: the relationship is marked as valid if d_t is contained in or is equivalent to d_q.

When an attribute domain d_t (d_q) is defined using the `union` constructor, as in the `Address` example, a *valid relationship* is recognized if at least one domain d_t (d_q) is compatible with d_q (d_t).

Example 3 Referring to our Thesaurus resulting from Examples 1 and 2, the output of the validation phase is the following (for each relationship, control flag [1] denotes a valid relationship while [0] an invalid one):

⟨ED.Fast-Food.category BT FD.Bistro.type⟩ [0]
⟨ED.Owner.name BT FD.Person.first_name⟩ [1]
⟨ED.Owner.name BT FD.Person.last_name⟩ [1]
⟨ED.Fast-Food.specialty BT FD.Bistro.special_dish⟩ [1]

3.4 Inferring new relationships

In this step, inference capabilities of ODB-Tools are exploited: a new set of terminological relationships is inferred on the "virtual schema".

Example 4 Terminological relationships inferred in this step are the following:
⟨FD.Bistro RT ED.Owner⟩,
⟨FD.Bistro RT ED.Address⟩,
⟨FD.Brasserie RT ED.Address⟩,
⟨FD.Brasserie RT FD.Person⟩,
⟨FD.Restaurant RT ED.Address⟩,
⟨ED.Fast-Food BT FD.Brasserie⟩,
⟨ED.Fast-Food BT FD.Bistro⟩,
⟨FD.Restaurant RT ED.Fast-Food⟩,
⟨FD.Restaurant RT ED.Owner.⟩

Inferred semantic relationships are represented as new terminological relationships enriching the Thesaurus. The result of the overall process is the so-called Common Thesaurus. A graphical representation of the Common Thesaurus for ED and FD sources is reported in figure 5, where solid lines represent explicit relationships (i.e., extracted/supplied), dashed lines represent inferred relationships, and superscripts indicate their kind.[2]

Note that, due the simplicity of the adopted example, many of the discovered relationships are a direct consequence of the relationships obtained by WordNet and supplied by the designer, except for the BT relationships between FD.Brasserie and ED.Fast-Food and FD.Bistro and ED.Fast-Food. All the discovered relationships are very useful in order to identify semantically similar concepts in different sources, as will be shown in next section.

ODB-Tools performs validation and inference steps by exploiting subsumption (i.e. generalization) and equivalence computation. As we showed in [5, 11], the computation of subsumption and equivalence in OLCD is decidable. Furthermore, even if from a purely theoretical point of view this

[2]For the sake of simplicity, only relationships between class names are reported.

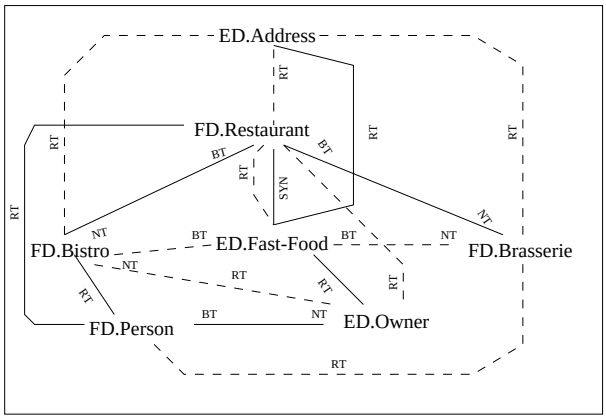

Figure 5: Common Thesaurus for `Eating` and `Food` Source

computation is PSPACE-hard (as proved in [11]), these problems can be efficiently solved by transforming a schema in a canonical form. These results imply that computing the canonical extension of a schema is difficult or that the canonical extension of a schema has a worst-case size that is exponential in the size of the original schema. However, the intractability previously mentioned rarely occurs in practice as a schema is generally formulated in such a way as to be "almost" canonical. Hence, we can conclude that transforming a schema to its canonical extension is feasible in polynomial time for most cases that appear in practice.

4 Semantic information integration

In this section, we describe the information integration process, to construct the global integrated view of ODL_{I^3} source schemas, based on the knowledge in the Common Thesaurus. The proposed technique allows semi-automatic identification of semantically similar ODL_{I^3} classes by means of clustering procedures based on the knowledge in the Common Thesaurus. Moreover, it supports semi-automatic synthesis of clusters of semantically related ODL_{I^3} classes, by handling semantic heterogeneity properly.

4.1 Affinity-based clustering of ODL_{I^3} classes

To integrate the ODL_{I^3} classes of the different sources into global ODL_{I^3} classes, we employ hierarchical clustering techniques based on the concept

of *affinity*. This way, we identify ODL_{I^3} classes that describe the same or semantically related information in different source schemas.

This activity is performed with the support of the ARTEMIS tool environment. ARTEMIS has been conceived for a semi-automatic integration of heterogeneous structured databases [21, 9]. In the context of MOMIS, the ARTEMIS affinity framework has been extended to be applied to the analysis of ODL_{I^3} schema descriptions. In the following, we describe the extensions to the affinity-based clustering introduced in MOMIS to cope with object pattern and semistructured data integration.

ODL_{I^3} classes are analyzed and compared by means of *affinity coefficients* which allow us to determine the level of similarity between classes in different source schemas. In particular, ARTEMIS evaluates a *Global Affinity* coefficient as the linear combination of a *Name Affinity* coefficient and a *Structural Affinity* coefficient, respectively.

Affinity coefficients for ODL_{I^3} classes are evaluated by exploiting terminological relationships in the Common Thesaurus. To this end, a strength $\sigma_{\Re}$ is assigned to each type of terminological relationship $\Re$ in the Common Thesaurus, with $\sigma_{\text{SYN}} \geq \sigma_{\text{BT/NT}} \geq \sigma_{\text{RT}}$. In the following, when necessary, we use notation $\sigma_{ij_{\Re}}$ to denote the strength of the terminological relationship $\Re$ for terms t_i and t_j in the Thesaurus; furthermore, we use $\sigma_{\text{SYN}} = 1$, $\sigma_{\text{BT}} = \sigma_{\text{NT}} = 0.8$ and $\sigma_{\text{RT}} = 0.5$.

Two terms have affinity if they are connected through a path in the Common Thesaurus. Their level of affinity depends on the length of the path, on the type of relationships involved in this path, and on their strengths. The affinity of two terms in the Thesaurus is 0 if a path does not exist between them in the Common Thesaurus and it is 1 if they coincide or are synonyms. In remaining cases, the longer the path defined between two terms, the lower their affinity. For a given path length, the higher the strength of the involved relationships, the greater the affinity of the involved terms.

Let c and c' be two classes belonging to sources S and S' respectively. Let us now define how the affinity coefficients are computed.

4.1.1 Name Affinity coefficient

The Name Affinity coefficient of two classes c and c', denoted $NA(c, c')$, is the measure of the affinity of their names n_c and $n_{c'}$, represented as terms in the Common Thesaurus (see Table 1).

For any pairs of classes, $NA(c, c') \in [0, 1]$. We introduce a threshold α to select classes having high values of Name Affinity (e.g., $\alpha \in [0.4, 0.6]$).

Coefficient	Value	Condition
$NA(c, c')$	$0 < \sigma_{12_{\Re}} \cdot \sigma_{23_{\Re}} \cdot \ldots \cdot \sigma_{(m-1)m} \leq 1$	if $n_c \rightarrow^m n_{c'}$ AND $NA(c, c') \geq \alpha$
	0	in all other cases

Legend:
$n_c \rightarrow^m n_{c'}$ denotes the path of length m, with $m \geq 1$, between n_c and $n_{c'}$ in the Common Thesaurus for which $NA(c, c')$ is the highest value.
α is a threshold used to select name having high affinity.

Table 1: Name Affinity coefficient

Example 5 Consider the Common Thesaurus illustrated in figure 5. A path exists between `ED.Address` and `FD.Restaurant` in the Thesaurus.

$$\texttt{ED.Address} \xrightarrow{\text{RT}} \texttt{ED.Fast} - \texttt{Food} \xrightarrow{\text{SYN}} \texttt{FD.Restaurant}$$

Therefore, we have that $NA(\texttt{ED.Address}, \texttt{FD.Restaurant}) = 0.5 \cdot 1 = 0.5$. Furthermore, between `FD.Person` and `FD.Brasserie` several paths exist. In particular, if we consider the inferred relationship $\texttt{FD.Brasserie} \xrightarrow{\text{RT}} \texttt{FD.Person}$, the highest value path is $NA(\texttt{FD.Brasserie}, \texttt{FD.Person}) = 0.5$, otherwise the highest value is due to the path $\texttt{FD.Brasserie} \xrightarrow{\text{NT}} \texttt{FD.Restaurant} \xrightarrow{\text{RT}} \texttt{FD.Person}$ path, giving $NA(\texttt{FD.Brasserie}, \texttt{FD.Person}) = 0.8 \cdot 0.5 = 0.4$. Therefore, in general, inferring new terminological relationships lead to evaluate more concepts with "affinity" and a greater value of "affinity" among concepts.

In the remainder of the paper symbol $\sim$ will be used to denote affinity between names.

4.1.2 Structural Affinity coefficient

The Structural Affinity coefficient of two classes c and c', denoted $SA(c, c')$, is the measure of the affinity of their attributes (see Table 2).

The Structural Affinity coefficient (refined by a control factor F_c), returns a value in the range $[0, 1]$ proportional to the number of attributes whose names have affinity in the Common Thesaurus. The value 0 indicates the absence of attributes with affinity in the considered classes, while the value 1 indicates that all attributes defined in the two classes have affinity. The greater the number of attributes with affinity in the considered classes, and the greater the number of positive control results, the higher the $SA()$ value for the classes.

Coefficient	Value	Condition
$SA(c, c')$	$\frac{\lvert\{a_t \mid a_t \in A(c), a_q \in A(c'), n_t \sim n_q\}\rvert + \lvert\{a_q \mid a_t \in A(c), a_q \in A(c'), n_t \sim n_q\}\rvert}{\lvert A(c)\rvert + \lvert A(c')\rvert} \cdot F_c$	if $\lvert C\rvert \neq 0$
	0	if $\lvert C\rvert = 0$

Legend:
$C = \{(a_t, a_q) \mid a_t \in A(c), a_q \in A(c'), n_t \sim n_q\}$, where $A(c)$ and $A(c')$ are the sets of attributes in c and c', respectively
$F_c = \frac{\lvert\{x \in C \mid \ flag(x)=1 \ \}\rvert}{\lvert C\rvert}$, where $flag(x) = 1$ stands for a positive result

Table 2: Structural Affinity coefficient

In the $SA()$ formula, the control factor F_c evaluates the percentage of valid relationships between attributes obtained in the step Validation of relationships illustrated in Section 3.3.
In general, given two classes, an attribute of one class may have affinity with more than one attribute of the other class. In the evaluation of the $SA()$ coefficient, we consider these multiple affinities as a single affinity between one attribute and a set of attributes.

For the evaluation of Structural Affinity, *optional* attributes of ODL_{I^3} classes representing object patterns must be considered properly. Depending on which attributes are taken into account, the following options are possible for the computation of the $SA()$ coefficient:

1. *All attribute-based.* With this option, *optional* attributes are treated as the other ones and are always taken into account when evaluating the affinity of ODL_{I^3} classes describing object patterns.

2. *Common attribute-based.* With this option, *optional* attributes are not taken into account when evaluating the affinity.

3. *Threshold-based.* With this option, *optional* attributes are taken into account for affinity evaluation only if they are common to at least a certain number of objects (i.e., a threshold) of the considered object pattern.

The third option is difficult to apply, since it requires setting the value of a threshold, which can be dependent on the specific object pattern or on the source. As for the other two options, they have different implications on the affinity values produced. Given a pattern to be compared, the second option gives affinity values higher than the first one, in presence of the same number of attribute pairs with affinity. In fact, less attributes are considered

at the denominator of the $SA()$ formula choosing option 2). On the other side, if most attributes of an object pattern are optional, then the option 1) is better for Structural Affinity evaluation. The choice between the first two options depends on the specific application under analysis. In our example, we applied both options and we discuss obtained values in the following, when presenting results of clustering.

Example 6 Consider classes `ED.Owner` and `FD.Person`. By applying the all attribute-based option, we have that

$$SA(\texttt{ED.Owner}, \texttt{FD.Person}) = \frac{1+2}{3+4} \cdot 1 = 0.43$$

due to the following affinities:
`ED.Owner.name`$\sim\{$`FD.Person.first_name`,`FD.Person.last_name`$\}$.

4.1.3 Global Affinity coefficient

The Global Affinity coefficient of two classes c and c', denoted $GA(c, c')$, is the measure of their affinity computed as the weighted sum of the Name and Structural Affinity coefficients (see Table 3).

Coefficient	Value	Condition
$GA(c, c')$	$w_{NA} \cdot NA(c, c') + w_{SA} \cdot SA(c, c')$	in all cases

Legend:
w_{NA} and w_{SA}, with $w_{NA}, w_{SA} \in [0, 1]$ and $w_{NA} + w_{SA} = 1$, are introduced to assess the relevance of each coefficient in computing the global affinity value.

Table 3: Global Affinity coefficient

Weights in $GA(c, c')$ allow the analyst to differently stress the impact of each coefficient in the evaluation of the global affinity value.

Example 7 The Global Affinity coefficient of `ED.Owner` and `FD.Person` is computed as follows:

$$GA(\texttt{ED.Owner}, \texttt{FD.Person}) = 0.5 \cdot 0.8 + 0.5 \cdot 0.43 = 0.61$$

using $w_{NA} = w_{SA} = 0.5$, since we consider both affinity coefficients equally relevant.

4.1.4 Considerations on the affinity evaluation process

The goodness of affinity evaluation relies on both the linguistic correctness of the terminological relationships and on the parameters (i.e., strengths, weights, thresholds) intervening in the calculation of the coefficients and their relative values. As it is intuitive, affinity has an element of subjectivity, due to the fact that the knowledge and experience of the designer can be required to integrate terminological relationships typical of the domain under analysis and to properly set parameter values. To make affinity even more objective, interactive functionalities are provided in MOMIS. In particular, the construction and validation of the Common Thesaurus in ODB-Tools is interactive, and the designer can supply additional terminological relationships typical of application domain under analysis. The affinity evaluation process is also interactive and weight-based in the ARTEMIS tool environment, to enable the designer to properly set the involved parameters and to validate the choices performed by the tool in all steps of the process. The ARTEMIS affinity approach has been experimented on different sets of conceptual database schemas to select a set of default values (i.e., the ones working satisfactorily in most cases) for setting the tool. The values of terminological relationship strengths in the Thesaurus, and of affinity weights and thresholds used in the examples of this paper correspond to these selected default values. Default values can however be dynamically varied by the designer when necessary, to tailor the affinity calculation to the specific application. For a deeper discussion of the experimentation of the use of a dictionary of strengthened terminological relationships for affinity analysis of data schemas in the Italian Public Administration domain, the reader can refer to [21, 22]. More general considerations on usage and settings of semi-automatic, weight-based techniques for large-scale conceptual schema analysis and comparison can be found in [20].

4.1.5 Clustering of ODL$_{I^3}$ classes

To identify all the ODL$_{I^3}$ classes having affinity in the considered source schemas, we employ a hierarchical clustering technique, which classifies classes into groups at different levels of affinity, forming a tree [27].

The hierarchical clustering procedure uses a matrix M of rank K where k the total number of ODL$_{I^3}$ classes to be analyzed. An entry $M[h,k]$ of the matrix represents the affinity coefficient $GA(c_h, c_k)$ between classes c_h and c_k.

Clustering is iterative and starts by placing each class in a cluster by

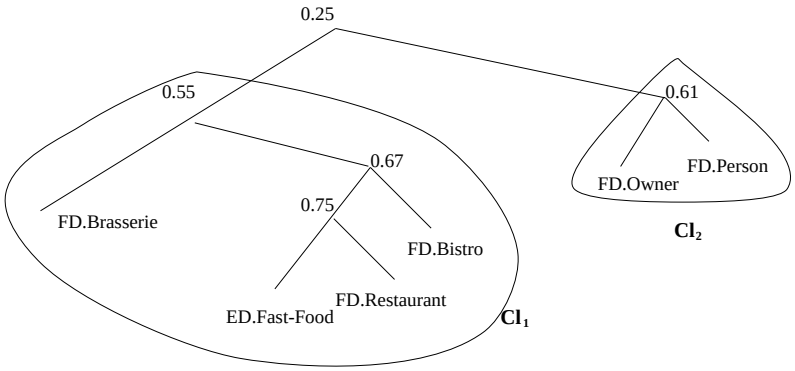

Figure 6: Example of affinity tree

itself. Then, at each iteration, the two clusters having the greatest $GA()$ value in M are merged. M is updated at each merging operation by deleting the rows and the columns corresponding to the merged clusters, and by inserting a new row and a new column for the newly defined cluster c_{hk}. The $GA()$ value between c_{hk} and each remaining cluster $\bar{c}$ in M is computed. The new value between c_{hk} and a remaining cluster is set to the maximum $GA()$ value between the $GA()$ values c_h and c_k had with $\bar{c}$ in the matrix. The procedure terminates when only one cluster is left and produces as the output a tree of ODL_{I^3} classes, where intermediate nodes have an associated $GA()$ value and leaves are ODL_{I^3} classes.

Figure 6 shows the affinity trees resulting from clustering our set of ODL_{I^3} classes by using the *all attribute-based* method.

Once the affinity tree has been constructed, the question is the selection of clusters to be integrated for the definition of the global schema. Cluster selection is interactive, based on the numerical affinity values in the affinity tree. In particular, ARTEMIS provides a threshold-based mechanism for cluster selection. The designer specifies a value for a threshold T and clusters characterized by a $GA()$ value greater than or equal to T are selected and proposed. High values of T return small, highly homogeneous clusters. By decreasing T's value, clusters with more classes can be selected. In the tool, the default value of T is set to 0.5. This default value can be refined dynamically, on the basis of the specific application under analysis.

4.2 Synthesis into an integrated schema description

Synthesis of clusters of ODL$_{I^3}$ classes requires to take into account semantic heterogeneity, which has to be treated properly to come up with an integrated and uniform representation at the global level.

The generation of global ODL$_{I^3}$ classes out of selected clusters is interactive with the designer. Let Cl_i be a selected cluster in the affinity tree and gc_i the global ODL$_{I^3}$ class to be defined for Cl_i. First, we associate with the gc_i a set of global attributes, corresponding to the union of the attributes of the classes belonging to Cl_i. The attributes having a valid terminological relationship are unified into a unique global attribute in gc_i. The attribute unification process is performed automatically for what concerns names according to the following rules:

- for attributes that have a SYN relationship, only one term is selected as the name for the corresponding global attribute in gc_i;
- for attributes that have a BT/NT relationship, a name which is a broader term for all of them is selected and assigned to the corresponding global attribute in gc_i.

For example, the attribute unification process for cluster Cl_1 of figure 6 automatically produces the following set of global attributes:
`name, address, phone*, specialty, category, nearby*, midprice*, owner*, special_dish, street, zip_code, type, r_code, b_code, pers_id, tourist_menu_price`

The designer can add mapping rules to properly set the global class. A global class includes also mapping rules for global attributes. A mapping rule is defined for each global attribute a of gc_i and specifies:

- *Attribute correspondences in the cluster*: values of a depends on the attributes that have been unified into a during the construction of gc_i. Mapping rules are defined to state for a which attributes of the ODL$_{I^3}$ classes in the cluster under analysis correspond to a. In specifying mapping rules for global attributes, the following correspondences can be specified:

  1. *And correspondence*: this specifies that a global attribute corresponds to the concatenation of two or more attributes of a class $c_h \in$ `Cl`$_i$.
     For example, by defining a mapping rule for the global attribute

name of $\texttt{Cl}_2$, the designer specifies that a global attribute name corresponds to both first_name and last_name attributes of FD.Person class. By specifying the *and* correspondence between first_name and last_name for the global attribute name, the designer states that the values of both first_name and last_name attributes have to be considered as values of name when class FD.Person is considered.

2. *Or correspondence*: this specifies that a global attribute corresponds to at most one among the attributes of a class $c_h \in \texttt{Cl}_\texttt{i}$. An *or* correspondence is useful when a global attribute is suitable for two or more local attributes of a source, depending on the value of another local attribute, called "tag attribute". For example, let us suppose to have a cluster describing an automobile class and that classes in the cluster have price values for cars in Italian Lire and US Dollars. Here, country is the tag attribute. In this example, it is possible to define an *or* correspondence between the attributes Italian_price and US_price by declaring the following mapping rule:

```
...
attribute integer price
 mapping rule(S.car.Italian_price union
              S.car.US_price on Rule1),
              ...
...
rule Rule1 { case of S.car.country:
      ''Italy'' : S.car.Italian_price;
      ''US''    : S.car.US_price; }
```

- *Default/null values*: they are possibly defined for local attributes corresponding to a, based on the knowledge of the single local source, if a is not applicable in the considered source. For example, with reference to Cl_1, the mapping rule defined for the global attribute zone specifies that the objects of the class ED.Fast-Food regarding the "Pacific Area", while objects of FD.Restaurant and FD.Bistro wherever in the USA.

For each global ODL_{I^3} class gc_i, a persistent mapping table storing all the mapping information is generated. As an example, the mapping table for the Food_Place class, set by the mapping rules of figure 7, is shown in figure 8.

```
interface Food_Place
{ attribute name
    mapping_rule ED.Fast-Food.name,
                 FD.Restaurant.name,
                 FD.Brasserie.name;
     ...
  attribute category
    mapping_rule ED.Fast-Food.category,
                 FD.Restaurant.category,
                 FD.Bistro.type;
  attribute specialty
    mapping_rule ED.Fast-Food.specialty,
                 FD.Restaurant.special_dish;
  attribute address
    mapping_rule ED.Fast-Food.address,
                 (FD.Restaurant.street and
                 FD.Restaurant.zip_code and
                 FD.Brasserie.address);
  attribute price
    mapping_rule ED.Fast-Food.midprice,
                 FD.Restaurant.tourist_menu_price;
  attribute zone
    mapping_rule ED.Fast-Food = 'Pacific Coast'',
                 FD.Restaurant = 'Atlantic Coast',
                 FD.Bistro = 'Atlantic Coast',
                 FD.Brasserie = 'Atlantic Coast';
}
```

Figure 7: Example of global class specification in ODL_{I^3}

Food_Place	code	name	...	zone
ED.Fast-Food	null	name	...	'Pacific Coast'
FD.Restaurant	r_code	name	...	'Atlantic Coast'
FD.Bistro	r_code	null	...	'Atlantic Coast'
FD.Brasserie	b_code	name	...	'Atlantic Coast'

Figure 8: `Food_Place` mapping table

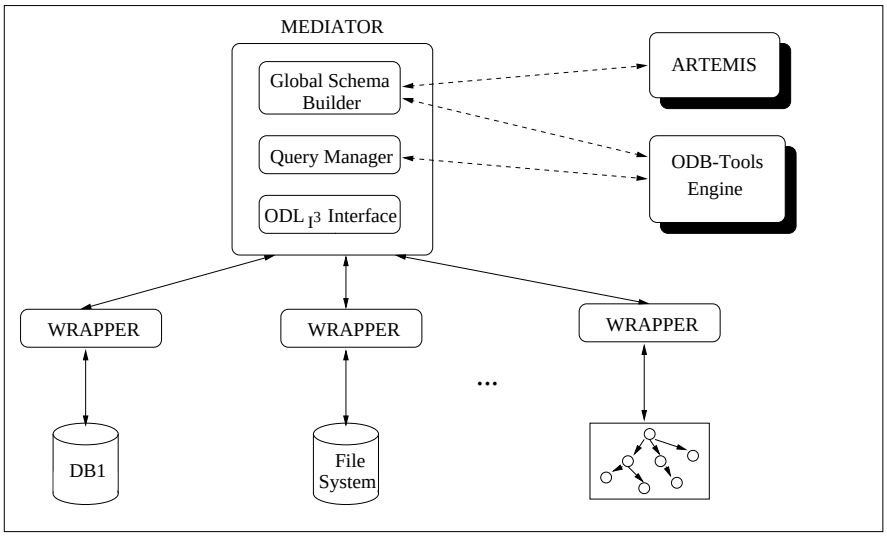

Figure 9: Architecture of MOMIS

Integrity constraint rules can also be specified for global ODL$_{I^3}$ classes to express semantic relationships holding among the different sources. Suppose that in our domain, a relationship exists between the category and the price of a food place. For example, the fact that all the food places with a 'High' category have a price higher then $ 100 can be expressed by the following integrity constraint rule in the global schema:

```
rule Rule2 forall X in Food_Place :
  (X.category ='High') then X.price > 100;
```

5 Architecture of the MOMIS system

In this section we describe the architecture of the MOMIS system. The MOMIS system has been conceived to provide an integrated access to heterogeneous information stored in traditional databases (e.g., relational, object-oriented) or file systems, as well as in semistructured sources. MOMIS is based on the I^3 architecture [4] (see figure 9): at the bottom layer we have the schema of information sources, while the layers above provide the semantic integration and the coordination management support. The integration of structured and semistructured data sources is performed in a semi-automatic way in MOMIS, by exploiting schema ODL_{I^3} descriptions of the sources, using the Description Logics and clustering techniques previously illustrated. Main components of MOMIS are the following:

- *Wrappers*. They are placed on top of the information sources and are responsible for translating the schema of the source into the ODL_{I^3} language. A wrapper performs also the translation of a query expressed in the ODL_{I^3} language into a local request executable by the query processor of the corresponding source.

- *Mediator*. It is composed of two modules: the *Global Schema Builder* (GSB) and the *Query Manager* (QM). The GSB module processes and integrates ODL_{I^3} descriptions received from wrappers to derive the integrated representation of the information sources. The QM module performs query processing and optimization. In particular, it generates the OQL_{I^3} queries for wrappers, starting from a global OQL_{I^3} query formulated by the user on the global schema. Using Description Logics techniques, the QM component can generate in an automatic way the translation of the global OQL_{I^3} query into different sub-queries, one for each involved local source.

- The *ODB-Tools Engine*, a tool based on the OLCD Description Logics [5, 11, 8] which performs schema validation for the generation of the Common Thesaurus and query optimization [6].

- The *ARTEMIS Tool Environment*, a tool based on affinity-based clustering techniques which performs ODL_{I^3} class analysis and clustering [19, 21].

MOMIS provides the following extraction and integration functionalities:

1. *Extraction of terminological relationships.* Terminological relationships are derived in a semi-automatic way from ODL_{I^3} schema descriptions, by analyzing structure and context of classes, by using ODB-Tools and the Description Logics techniques illustrated in Section 3.

2. *Affinity-based clustering of ODL_{I^3} classes*, with the support of the ARTEMIS-Tool environment. Terminological relationships in the Thesaurus are used by ARTEMIS to assess the level of *affinity* between ODL_{I^3} classes by interactively computing the affinity coefficients. ODL_{I^3} classes with affinity are automatically classified using hierarchical clustering techniques [21].

3. *Construction of the mediator global schema*, with the support of ODB-Tools. Affinity clusters of ODL_{I^3} classes are interactively selected in ARTEMIS and passed to ODB-Tools to construct the global schema of the Mediator. An integrated global ODL_{I^3} class is interactively defined for each selected cluster, together with its corresponding mapping table. OLCD and ODB-Tools are exploited for a semi-automatic generation of the global ODL_{I^3} classes. The set of global ODL_{I^3} classes defined constitutes the global schema of the Mediator to be used for posing queries against the sources.

At present the GSB software component has been implemented and is under testing where the QM component is under development. In particular, the present version of the QM component supports global queries with respect to a "Global Virtual View" of the integrated sources in a way transparent to the user and decompose a global query in terms of source related sub-queries for relational/object database sources and performs semantic optimization. One of the most innovative aspects of the QM in fact, consists in employing Description Logics based components (i.e. ODB-Tools) which can perform, both on the global and local queries, semantic optimization steps which minimize both the number of accessed sources and the volume of data to be integrated, as will be sketched in section 6.

6 Query Processing and Optimization

In this section, we briefly describe how the global ODL_{I^3} schema is exploited for global query processing. When the user submits a query to the MOMIS system, the MOMIS Query Manager (QM) produces a set of subqueries that will be sent to each involved information source.

According to other semantic approaches [3], this process consists of two main phases:

- semantic optimization
- query plan formulation

6.1 Semantic Optimization

In this phase, the QM operates on the query by exploiting the semantic optimization techniques [44] supported by ODB-Tools [8, 6, 7], in order to reduce the query access plan cost. The query is replaced by a new one that incorporates any possible restriction which is not present in the original query but is logically implied by the global schema (classes descriptions and integrity rules). The transformation is based on logical inference from content knowledge (in particular on the integrity constraints rules) of the mediator global schema, shared among the classes belonging to the cluster. Let us consider the request: "Retrieve the places with 'High' category and in the New York area with zip code 98654", corresponding to the following query:

```
select name from Food_Place
where category = 'High' and address like '%98654%'
```

The QM, using the query optimizer of ODB-Tools, executes the semantic expansion of the query by applying rule `Rule2`.

```
rule Rule2 forall X in Food_Place :
  (X.category ='High') then X.price > 100;
```

The resulting query is the following:

```
select name from Food_Place
where category = 'High' and price > 100
and address like '%98654%'
```

Semantic expansion is performed in order to add predicates in the "where clause": this process makes query plan formulation more expensive (because a heavier query has to be translated for each interesting source) but single sources query processing overhead can be lighter in case secondary indexes on added predicates exist in the involved sources (i.e. an index on the price attribute with reference to the example). The query optimization algorithm

included in ODB-Tools and presented in [8] is polynomial. Our experiments show that the overhead of optimization is very small compared to the overall query processing cost. On a set of 8 queries performed on 10 different database instances, the query optimization gave an average reduction in execution time of 47%.

6.2 Query Plan Formulation

Once the mediator has produced the global optimized query, a set of subqueries for the local information sources will be formulated. For each information source, by using the mapping table associated with each global class, the QM has to express the optimized query in terms of local schemas. In order to obtain each local query, the mediator checks and translates every predicate in the where clause. In particular, a local query is generated only when all the attributes of the where clause have a not-null correspondence in the local source.
Referring to our example, the algorithm will exclude the `FD.Brasserie` and the `FD.Bistro` class which do not have a category attribute[3], so that we derive the following queries:

```
FD: select name, address from Restaurant R
    where R.category = 'High' and R.tourist_menu_price > 100
    and concat(R.zipcode, R.street) like '%98654%'

ED: select name, address from Fast-Food F
    where F.category = 'High' and F.midprice > 100
    and (F.address like '%98654%' or
         concat(F.address.zipcode,
                F.address.street,
                F.address.city) like '%98654%')
```

where the `concat` function is automatically generated in the presence of an AND correspondence in the mapping table. For the semistructured `ED` source, the global clause on the `address` attribute is translated into the local attributes having different structures in the data source. The resulting local query is thus obtained by inserting the alternative structures in the where clause joined by the `or` operator.

[3] The user is informed that Brasserie and Bistro do not admit a category attribute and, therefore, he may properly reformulate the query.

An other example to illustrate the QM functionality is the following "Retrieve the places with a specialty called 'Lasagne' ", corresponding to the query:

```
select name from Food_Place
where specialty = 'Lasagne'
```

After the global semantic optimization, that do not modify the global query, the QM generates the local query for the wrappers. For the semistructured source `ED` the `specialty` attribute maps to a set of value, so the simple query rewriting from the global class to the local object pattern would give a type error (comparison between a value and a set of values). However, the user intention is clear, he requests the restaurants which specialty including 'Lasagne', that is:

```
exists X in R: X.specialty = 'Lasagne'
```

Predicate transformation is automatically performed by the QM as suggested also by Abiteboul in [2].

Thus the resulting queries are the following (in the `FD` sources only the `Restaurant` class has the mapping to the specialty attribute):

```
ED: select name from Fast-Food F
    where exists X in F: X.specialty = 'Lasagne'

FD: select name from Restaurant R
    where R.special_dish = 'Lasagne'
```

An other example to illustrate the optimization process is the following "Retrieve the places located in the 'Atlantic Coast' ", corresponding to the query:

```
select name from Food_Place
where zone = 'Atlantic Coast'
```

The optimization process performs a pattern matching with the default value of the attribute `zone` in the mapping table, and detects an inconsistency for the `ED.fast-Food` class (whose object refer to `'Pacific Coast'`); thus the `ED` source is not quering and the resulting queries are the following:

```
FD: select name from Restaurant R
    where R.zone = 'Atlantic Coast'

    select name from Bistro B
    where B.zone = 'Atlantic Coast'

    select name from Brasserie B
    where B.zone = 'Atlantic Coast'
```

7 Related work and discussion

Works related to the issues discussed in this paper are in the area of semistructured data and heterogeneous information integration.

Semistructured data. The issue of modeling semistructured data has been investigated in the literature. In particular, a survey of problems concerning semistructured data modeling and querying is presented in [13]. Two similar models for semistructured data have been proposed [41], based on rooted, labeled graph with the objects as nodes and labels on edges. According to the model presented in [15], information resides at labels only, while according to the "Object Exchange Model" (OEM) proposed by Papakonstantinou et. al. in [41], information also resides at nodes.

Very similar proposal for modelling semi-structured data come from the Artificial Intelligence area [16], where in analogy with our approach, a Description Logics is adopted.

The issue of adding structure to semistructured data, which is more directly concerned with our concept of object pattern, has also been investigated. In particular, in [31], the notion of *dataguide* has been proposed as a "loose description of the structure of the data" actually stored in an information source. A proposal to infer structure in semistructured data has been presented in [37], where the authors use a graph-based data model derived from [15, 41]. In [13], a new notion of a graph schema appropriate for rooted, labeled graph databases has been proposed. The main usages of the structure extracted from a semistructured source have been presented for query optimization. In fact, the existence of a path in the structure simplifies query evaluation by limiting the query only to data that are relevant. In this paper, we are more concerned with usage of the structure in form of object patterns to support the integration of semistructured sources with structured databases.

Heterogeneous information integration. In this area, many projects based on a mediator architecture have been developed [3, 25, 18]. MOMIS is based on a mediator architecture and follows the 'semantic approach'. Following the classification of integration system proposed by Hull [32], MOMIS is in the line of the "virtual approach". Virtual approach was first proposed in multidatabase models in the early 80s. More recently, systems have been developed based on the use of Description Logics [35] such as CLASSIC [12]. With references to the same classification proposed by Hull, MOMIS is in the category of "read-only views", that is, systems whose task is to support an integrated, read-only, view of data that resides in multiple databases. The most similar projects are: GARLIC [18], SIMS [3], Information Manifold [35] and Infomaster [30].

New techniques for the analysis and the integration of conceptual schemas of distributed databases are presented in [20]. The discovery, analysis and representation of inter-schema properties (such as synonyms, inclusions, type mismatch, etc.) is another critical aspect of the integration process. In [38] semi-automatic techniques for extracting synonymies, homonyms and object inclusions from database schemes are described. In [39] a graph-based approach to detect type conflicts in database schemes is proposed. Finally, in [40] a semi-automatic algorithm for integrating and abstracting database schemes is presented. It is worth noticing that the design of systems for information gathering from multiple sources is also addressed in Artificial Intelligence through multi-agent systems [34], concentrating mainly on high level tasks (co-operation, planning, belief revision, etc.) related to the extraction process [26].

On the other hand, many projects are based on a 'structural' approach [25]. The TSIMMIS project [25] follows a 'structural' approach and uses a self-describing model (OEM) to represent the data objects and pattern matching techniques to perform a predefined set of queries based on a query template. The semantic knowledge is effectively encoded in the MSL (Mediator Specification Language) rules enforcing source integration at the mediator level. Although the generality and conciseness of OEM and MSL make this approach a good candidate for the integration of widely heterogeneous and semistructured information sources, a major drawback in such an approach is that dynamically adding sources is an expensive task. In fact, new TSIMMIS sources not only must be wrapped, but the mediators that uses them have to be redefined and their MSL definitions recompiled. The administrator of the system must figure out whether and how to use the new sources.

8 Conclusions and future work

In this paper, we have presented an intelligent approach to information extraction and integration for heterogeneous information sources. It is a semantic approach based on a Description Logics component (ODB-Tools engine) and a cluster generator module, ARTEMIS, together with a minimal ODL_{I^3} interface module. Generation of the global schema for the mediator is a semi-automated process. The Description Logics-based ODL_{I^3} language is introduced for information extraction and integration, by taking into account also semistructured information sources.

Future research will be devoted to the "definition of extensional axioms" and provide "answer composition".

Extensional axioms can be used to define set relationships among the source extensions. The presence of a set of axioms which is both complete and correct is a precondition for achieving an effective integration. On the other hand, the generation of the axioms is left to the designer's experience and knowledge and consequently, it can be only partly carried out mechanically. It will be necessary, henceforth, to design and develop instruments which help the designer in the axiom specification phase and which "reason about the given axiom" increasing the knowledge for the integration task. The methodology that is meant to be used in order to exploit extensional knowledge consists in the individuation of the "base extensions", as recently proposed by [42, 43] and in reasoning activities performed by the Description Logics component. The use of base extension and Description Logics will allow to obtain significant results in semantic query optimization. The problem of "answer composition" consists in the following : the data provided by the sub-queries execution at the sources have to be reconciled giving rise the answer for the global query. In particular, very few proposals in the literature deal with the problem of the "fusion of objects" coming from different sources [33, 29]

Finally, the approach will be extended to take into account XML data sources.

References

[1] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener, "The Lorel Query Language for Semistructured Data", *Journal of Digital Libraries*, 1(1), November 1996.

[2] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel Query Language for Semistructured Data. International Journal on Digital Libraries, 1(1):68-88, April 1997.

[3] Y. Arens, C. A. Knoblock and C. Hsu, "Query Processing in the SIMS Information Mediator," in *Advanced Planning Technology, editor, Austin Tate, AAAI Press, Menlo Park, CA*, 1996.

[4] ARPA, "ARPA I^3 Reference Architecture", Available at http://www.isse.gmu.edu/I3_Arch/index.html

[5] D. Beneventano, S. Bergamaschi, S. Lodi and C. Sartori, "Consistency Checking in Complex Object Database Schemata with Integrity Constraints", in *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, pag. 576-598, 1998.

[6] D. Beneventano, S. Bergamaschi, C. Sartori, M. Vincini, "ODB-Tools: A Description Logics Based Tool for Schema Validation and Semantic Query Optimization in Object Oriented Databases", in *Proc. of Int. Conf. on Data Engineering, ICDE'97*, Birmingham, UK, April 1997. (http://sparc20.dsi.unimo.it/index.html)

[7] D. Beneventano, S. Bergamaschi, C. Sartori, M. Vincini, "ODB-QOptimizer: a Tool for Semantic Query Optimization in OODB", in *Proc. of Fifth Conference of the Italian Association for Artificial Intelligence (AI*IA97)*, Rome 1997, LNAI 1321.

[8] D. Beneventano, S. Bergamaschi and C. Sartori "Semantic Query Optimization by Subsumption in OODB," in *Proc. of the Int. Workshop on Flexible Query-Answering Systems (FQAS96)*, Roskilde, Denmark, may 1996, ISSN 0109-9779 N0. 62,1996.

[9] S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, M. Vincini, "An Intelligent Approach to Information Integration," in *International Conference on Formal Ontology in Information Systems (FOIS'98)*, Trento, Italy, June 1998.

[10] S. Bergamaschi, S. Castano e M. Vincini "Semantic Integration of Semistructured and Structured Data Sources", SIGMOD Record Special Issue on Semantic Interoperability in Global Information, Vol. 28, No. 1, March 1999.

[11] S. Bergamaschi and B. Nebel, "Acquisition and Validation of Complex Object Database Schemata Supporting Multiple Inheritance," *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks and Complex Problem Solving Technologies*, 4:185–203, 1994.

[12] A. Borgida, R.J. Brachman, D.L. McGuinness and L.A. Resnik, "CLASSIC: A Structural Data Model for Objects," in *SIGMOD*, pages 58-67, Portland, Oregon, 1989.

[13] P. Buneman, "Semistructured Data," in *Proc. of 1997 Symposium on Principles of Database Systems (PODS97)*, Tucson, Arizona, May 1997, pp. 117-121.

[14] P. Buneman, S. B. Davidson, M. F. Fernandez, D. Suciu: "Adding Structure to Unstructured Data", in *Proc. of ICDT 1997*, Delphi, Greece, January 8-10, 1997. Lecture Notes in Computer Science, Vol. 1186, Springer, 1997, ISBN 3-540-62222-5.

[15] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu "A Query Language and Optimization Techniques for Unstructured Data", in *Proc. of the ACM SIGMOD International Conference*, Montreal, Canada, June 1996, pp. 505-516.

[16] D. Calvanese, G. De Giacomo and M. Lenzerini, "What can knowledge representation do for semi-structured data?", in *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI-98)*, 1998, (205-210).

[17] D. Calvanese, G. De Giacomo, M. Lenzerini and M.Y. Vardi, "Rewriting of regular expressions and regular path queries", in *Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Syst. (PODS-99)*.

[18] M.J. Carey, L.M. Haas, P.M. Schwarz, M. Arya, W.F. Cody, R. Fagin, M. Flickner, A.W. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J.H. Williams and E.L. Wimmers, "Towards Multimedia Information System: The Garlic Approach", *IBM Almaden Research Center*, San Jose, 1994.

[19] S. Castano, V. De Antonellis, "Deriving Global Conceptual Views from Multiple Information Sources," invited paper, in *preProc. of ER'97 Preconference Symposium on Conceptual Modeling, Historical Perspectives and Future Directions*, Los Angeles, November 1997.

[20] S. Castano, V. De Antonellis, M.G. Fugini, B. Pernici, "Conceptual Schema Analysis: Techniques and Applications", accepted for publication on *ACM Transactions on Database Systems*, 1997.

[21] S. Castano, V. De Antonellis, "A Discovery-Based Approach to Database Ontology Design", *Distributed and Parallel Databases - Special Issue on Ontologies and Databases*, Vol.7, N.1, 1999.

[22] S. Castano, V. De Antonellis, "Semantic Dictionary Design for Database Interoperability", in *Proc. IEEE ICDE'97*, 1997.

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

[24] R. Cattel (ed.), *The Object Data Standard: ODMG 2.0*, Morgan Kaufmann, 1997.

[25] S. Chawathe, H. Garcia Molina, J. Hammer, K. Ireland, Y. Papakostantinou, J.Ullman, and J.Widom, "The TSIMMIS project: Integration of Heterogeneous Information Sources," *in IPSJ Conference, Tokyo, Japan*, 1994. ftp://db.stanford.edu/pub/chawathe/1994/tsimmis-overview.ps.

[26] A. F. Dragoni, "Belief Revision: from Theory to Practice", The Knowledge Engineering Review, Vol. 12, n. 2, 1997, (147-179).

[27] B. Everitt, *Cluster Analysis*, Heinemann Educational Books Ltd, Social Science Research Council, 1974.

[28] C. Fahrner and G. Vossen, "Transforming Relational Database Schemas into Object Oriented Schemas according to ODMG-93", *ICDOOD95*, LNCS 1013,1995.

[29] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos and J. Widom. "The TSIMMIS approach to mediation: Data models and Languages". In Journal of Intelligent Information Systems, 1997.

[30] M. R. Genesereth, A. M. Keller and O. Duschka, "Infomaster: An Information Integration System", in *proceedings of 1997 ACM SIGMOD Conference*, May 1997.

[31] R. Goldman and J. Widom, "DataGuides: Enabling Query Formulation and Optimization in Semistructured Data", in *Proc. of the Twenty-Third International Conference on very Large Data Bases*, 1997.

[32] R. Hull. Managing Semantic Heterogeneity in Databases: A Theoretical Perspective. *ACM Symp. on Principles of Database Systems*, pages 51-61, 1997.

[33] R.Hull and M. Yoshikawa, "Ilog:Declarative creation and manipulation of object identifiers", in *Proc. of the 16th VLDB Conference*, Australia, 1990.

[34] V. Lesser, B. Horling, F. Klassner, A. Raja, T. Wagner and S. XQ. Zhang, "BIG: A Resource-Bounded Information Gathering Agent" in *Proc. AAAI-98*.

[35] A. Y. Levy, A. Rajaraman, and J. J. Ordihe, " Querying heterogeneous information sources using source descriptions", in *Proc. of VLDB'96*, pages 251-262, 1996.

[36] A.G. Miller, "WordNet: A Lexical Database for English", *Communications of the ACM*, Vol. 38, No.11, November 1995, pp. 39 - 41.

[37] S. Nestorov, S. Abiteboul and R. Motwani, "Extracting Schema from Semistructured Data", in *Proceedings ACM SIGMOD International Conference on Management of Data*, June 2-4, 1998, Seattle, Washington, USA. ACM Press 1998, ISBN 0-89791-955-5.

[38] L. Palopoli, D. Saccà and D. Ursino. "Automatic derivation of terminological properties from database schemes", in *Proc. DEXA'98*, 90-99, Wien, Austria, 1998.

[39] L. Palopoli, D. Saccà and D. Ursino. "An automatic technique for detecting type conflicts in database schemes", in *Proc. ACM-CIKM'98*, 306-313, Bethesda , MD, 1998.

[40] L. Palopoli, D. Saccà and D. Ursino. "Semi-automatic semantic discovery of properties from database schemes", in *Proc. IDEAS'98*, 244-253, Cardiff, UK, 1998.

[41] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom, "Object Exchange Across Heterogeneous Information Sources", in *Proc. of ICDE*, Taipei, Taiwan, March 1995, pp. 251-260.

[42] I. Schmitt and C. Turker, "An Incremental Approach to Schema Integration by Refining Extensional Relationships", in *Proc. of the 7th ACM CIKM Int. Conf. on Information and Knowledge Management*, November 3-7, 1998, Bethesda, Maryland, USA, ACM Press, pp. 322-330, 1998.

[43] I. Schmitt and G. Saake, "Merging Inheritance Hierarchies for Database Integration", in *Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems*, New York, pg. 322-331, 1998.

[44] S. Shenoy and M. Ozsoyoglu, "Design and Implementation of a Semantic Query Optimizer," *IEEE trans. Knowl. and Data Engineering*, 1(3):344-361, September 1989.

[45] W.A. Woods and J.G. Schmolze, "The kl-one family", in F.W. Lehman, (ed), Special Issue of *Computers & Mathematics with Applications*, Vol. 23, No. 2-9, 1989.

A The ODL$_{I^3}$ description language

The following is a BNF description for the ODL$_{I^3}$ description language. We included the syntax fragments which differ from the original ODL grammar, referring to this one for the remainder.

⟨interface_dcl⟩ ::= ⟨interface_header⟩
{[⟨ interface_body⟩]
[**union** ⟨interface_body⟩]**};**
⟨interface_header⟩ ::= **interface** ⟨identifier⟩
[⟨inheritance_spec⟩]
[⟨type_property_list⟩]
⟨inheritance_spec⟩ ::= : ⟨scoped_name⟩
[,⟨inheritance_spec⟩]

Local schema pattern definition: the wrapper must indicate the kind and the name of the source of each pattern.

⟨type_property_list⟩ ::= **(** [⟨source_spec⟩]
[⟨extent_spec⟩]
[⟨key_spec⟩] [⟨f_key_spec⟩] **)**
⟨source_spec⟩ ::= **source** ⟨source_type⟩
⟨source_name⟩
⟨source_type⟩ ::= **relational** | **nfrelational**
| **object** | **file**
| **semistructured**
⟨source_name⟩ ::= ⟨identifier⟩
⟨extent_spec⟩ ::= **extent** ⟨extent_list⟩
⟨extent_list⟩ ::= ⟨string⟩ | ⟨string⟩,⟨extent_list⟩
⟨key_spec⟩ ::= **key**[**s**] ⟨key_list⟩
⟨f_key_spec⟩ ::= **foreign_key** (⟨f_key_list⟩)
references ⟨identifier
⟩[,⟨f_key_spec⟩]

...

Global pattern definition rule, used to map the attributes between the global definition and the corrisponding ones in the local sources.

⟨attr_dcl⟩ ::= **[readonly] attribute**
[⟨domain_type⟩]
⟨attribute_name⟩ **[*]**
[⟨fixed_array_size⟩]
[⟨mapping_rule_dcl⟩]

⟨mapping_rule_dcl⟩ ::= **mapping_rule** ⟨rule_list⟩

⟨rule_list⟩ ::= ⟨rule⟩ | ⟨rule⟩,⟨rule_list⟩

⟨rule⟩ ::= ⟨local_attr_name⟩ |
'⟨identifier⟩'
⟨and_expression⟩ |
⟨union_expression⟩

⟨and_expression⟩ ::= **(** ⟨local_attr_name⟩ **and**
⟨and_list⟩ **)**

⟨and_list⟩ ::= ⟨local_attr_name⟩
| ⟨local_attr_name⟩ **and**
⟨and_list⟩

⟨union_expression⟩ ::= **(** ⟨local_attr_name⟩ **union**
⟨union_list⟩ **on** ⟨identifier⟩ **)**

⟨union_list⟩ ::= ⟨local_attr_name⟩
| ⟨local_attr_name⟩ **union**
⟨union_list⟩

⟨local_attr_name⟩ ::= ⟨source_name⟩.⟨class_name⟩.
⟨attribute_name⟩

...

Terminological relationships used to define the Common Thesaurus.

⟨relationships_list⟩ ::= ⟨relationship_dcl⟩; |
⟨relationship_dcl⟩;
⟨relationships_list⟩

⟨relationships_dcl⟩ ::= ⟨local_name⟩
⟨relationship_type⟩
⟨local_name⟩

⟨local_name⟩ ::= ⟨source_name⟩.
⟨local_class_name⟩
[.⟨local_attr_name⟩]

⟨relationship_type⟩ ::= **SYN** | **BT** | **NT** | **RT**

...

OLCD integrity constraint definition: declaration of rule (using *if then* definition) valid for each instance of the data; mapping rule specification (*or* and *union* specification rule).

⟨rule_list⟩ ::= ⟨rule_dcl⟩; | ⟨rule_dcl⟩; ⟨rule_list⟩
⟨rule_dcl⟩ ::= **rule** ⟨identifier⟩ ⟨rule_spec⟩
⟨rule_spec⟩ ::= ⟨rule_pre⟩ **then** ⟨rule_post⟩ |
{ ⟨case_dcl⟩ **}**
⟨rule_pre⟩ ::= ⟨forall⟩ ⟨identifier⟩ **in** ⟨identifier⟩ **:**
⟨rule_body_list⟩
⟨rule_post⟩ ::= ⟨rule_body_list⟩
⟨case_dcl⟩ ::= **case of** ⟨identifier⟩ **:** ⟨case_list⟩
⟨case_list⟩ ::= ⟨case_spec⟩ | ⟨case_spec⟩ ⟨case_list⟩
⟨case_spec⟩ ::= ⟨identifier⟩ **:** ⟨identifier⟩ **;**
⟨rule_body_list⟩ ::= **(** ⟨rule_body_list⟩ **)** |
⟨rule_body⟩ |
⟨rule_body_list⟩ **and**
⟨rule_body⟩ |
⟨rule_body_list⟩ **and**
(⟨rule_body_list⟩ **)**
⟨rule_body⟩ ::= ⟨dotted_name⟩
⟨rule_const_op⟩
⟨literal_value⟩ |
⟨dotted_name⟩
⟨rule_const_op⟩
⟨rule_cast⟩ ⟨literal_value⟩ |
⟨dotted_name⟩ **in**
⟨dotted_name⟩ |
⟨forall⟩ ⟨identifier⟩ **in**
⟨dotted_name⟩ **:**
⟨rule_body_list⟩ |
exists ⟨identifier⟩ **in**
⟨dotted_name⟩ **:**
⟨rule_body_list⟩
⟨rule_const_op⟩ ::= = | ≥ | ≤ | > | <
⟨rule_cast⟩ ::= **(**⟨simple_type_spec⟩**)**
⟨dotted_name⟩ ::= ⟨identifier⟩ | ⟨identifier⟩.
⟨dotted_name⟩
⟨forall⟩ ::= **for all** | **forall**

B ODL$_{I^3}$ sources descriptions

`Eating_Source (ED):`

```
interface Fast-Food
( source semistructured Eating_Source )
{  attribute string         name;
   attribute Address        address;
   attribute integer        phone*;
   attribute set<string>    specialty;
   attribute string         category;
   attribute Restaurant     nearby*;
   attribute integer        midprice*;
   attribute Owner          owner*; };

interface Address
( source semistructured Eating_Source )
{  attribute string city;
   attribute string street;
   attribute string zipcode; };
  union
{  string; };

interface Owner
( source semistructured Eating_Source )
{  attribute string   name;
   attribute Address  address;
   attribute string   job; };

Food_Guide_Source (FD):

interface Restaurant
( source relational Food_Guide
  key r_code
  foreign_key(pers_id) references Person )
{  attribute string         r_code;
   attribute string         name;
   attribute string         category;
   attribute string         street;
   attribute string         zip_code;
   attribute integer        pers_id;
   attribute integer        tourist_menu_price;
   attribute string         special_dish; };
```

```
interface Person
( source relational Food_Guide
  key pers_id)
{  attribute integer        pers_id;
   attribute string         first_name;
   attribute string         last_name;
   attribute integer        qualification;};

interface Bistro
( source relational Food_Guide
  key r_code
  foreign_key(r_code) references Restaurant,
  foreign_key(pers_id) references Person)
{  attribute string      r_code;
   attribute set<string> type;
   attribute integer     pers_id;};

interface Brasserie
( source relational Food_Guide
  key b_code )
{  attribute string       b_code;
   attribute string       name;
   attribute string       address; };
```