Intelligent Techniques for the Extraction and Integration of Heterogeneous Information*

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

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 which take into account both structured and semistructured data sources. An object-oriented language called ODL_{I^3}, derived from the standard ODMG, with an underlying Description Logics, is introduced for information extraction. ODL_{I^3} descriptions of the information sources are exploited first to set a shared vocabulary for the sources. Information integration is performed in a semi-automatic way, by exploiting ODL_{I^3} descriptions of source schemas with a combination of Description Logics and clustering techniques. Techniques described in the paper have been implemented in the MOMIS system, based on a conventional mediator architecture.

Keywords - Heterogeneous information integration, Information extraction, Semistructured data, 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 of semantically related information. In fact, information sources available on-line in global information systems already exist and have been developed independently. Consequently, semantic heterogeneity for the aspects related to terminology, structure, and context of the information can arise, which have to be properly managed to effectively exploit the available information independent of the source it has been extracted from. The goal of information extraction and integration techniques is to construct synthesized, uniform descriptions of the information extracted from multiple heterogeneous sources, to provide the user with a uniform query interface against the sources. 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 propose intelligent, tool-supported techniques to information extraction and integration which take into account both structured and semistructured data sources. An object-oriented language, called ODL_{I^3}, derived from the standard ODMG, with the underlying Description Logic OLCD (Object Language with Complements allowing Descriptive cycles) [Beneventano *et al.*, 1998; Bergamaschi *Nebel*, 1994], derived from the KL-ONE family [Woods *et Schmolze*, 1989], is introduced. 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 first to set a shared ontology for the sources, in form of a Common Thesaurus, by exploiting the OLCD Description Logic inference capabilities. Furthermore, the Wordnet lexical system [Miller , 1995] is used to derive additional inter-sources terminological relationships for enriching the Common Thesaurus.

Information integration is performed in a semi-automatic way based on the knowledge in the Common Thesaurus, by means of clustering techniques and of the OLCD Description Logic internal representation

*This research has been partially funded by the *INTERDATA - Metodologie e Tecnologie per la Gestione di Dati e Processi su Reti Internet e Intranet - MURST ex-40%* project.

of ODL_{I^3} descriptions of source schemas. Mapping rules are defined at the global level to express the relationships holding between ODL_{I^3} integrated description 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 Modena 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 [Levy *et al.*, 1996; Carey *et al.*, 1994; Arens *et al.*, 1996], MOMIS follows a "semantic approach" to information integration based on the analysis of conceptual schemas of the information sources, and on a mediator and query-processing component. This latter component integrates two pre-existing tools, namely the ARTEMIS tool environment [Castano *De Antonellis*, 1997] and ODB-Tools [Beneventano *et al.*, 1997], to provide an I^3 architecture for the MOMIS system 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 management together with a running example used throughout the paper. In Subsection 2.2 and 2.3 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 enrich the semantic of the local sources by the construction of a Common Thesaurus of terminological relationship. In Section 4, we illustrate the integration process for building the mediator global schema, exploiting affinity-based technique clustering process for the formulation of groups of classes with affinity. In Section 5 we describe the MOMIS architecture, in Section 6, we discuss previous work on semistructured data modeling and heterogeneous information integration. Finally, in Section 7 we give our concluding remarks.

2 Information extraction from heterogeneous data schemas

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.

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. This task is generally performed by properly designed wrapper tools. 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 [Fahrner *Vossen*, 1995] for relational to ODMG schema conversion.

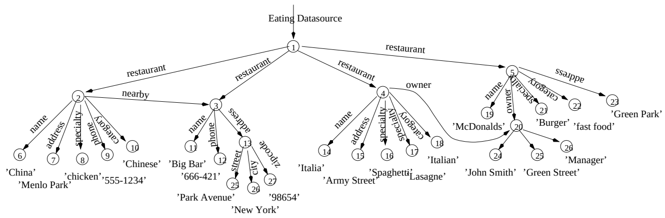

Figure 1: Eating Datasource (ED)

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 a schema is specified directly within the data. Models for semistructured information sources have been proposed in the literature [Buneman , 1997; Papakonstantinou *et al.*, 1995]. According to these models, 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. Fig. 1 shows an example of semistructured source, containing information related to the `Eating Datasource` that collects information on local restaurants (we use a notation similar to the one of the OEM model [Abiteboul *et al.*, 1996; Papakonstantinou *et al.*, 1995]).

When semistructured data sources are involved in the integration process, information extraction means also to provide the schema, or structure, of the source. In our approach, to reason about the schema of a semistructured source for integration purposes, we introduce the notion of object pattern as follows. A semistructured object (object, for short) can be viewed as a triple of the form $\langle \texttt{id}, \texttt{label}, \texttt{value} \rangle$, 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*). 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 the source. All complex 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 used to conveniently represent in a concise way all objects in each set_l. 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 $so \rightarrow so'$.*

From this definition, an object pattern is representative of all different objects that describe the same con-

Restaurant-pattern	=	(Restaurant,{name,address, 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

cept 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 Fig. 2. Three object patterns are defined: `Restaurant` containing information about restaurants; `Owner` containing information about people involved and `Address` object pattern.

Techniques for the extraction of object patterns from a semistructured data source should analyze the objects in the source, construct the sets set_l based on the analysis of the labels in the source, and define an object pattern out of each set_l. The extraction of object patterns presents strong analogies with the extraction of the structure from a semistructured source. For this reason, techniques recently proposed in the literature for structure extraction (see for example, [Buneman , 1997; Papakonstantinou *et al.*, 1995; Nestorov *et al.*, 1997]) can be used also for object pattern extraction. In the following, we will concentrate on the use of object patterns for integration rather than on their extraction.

It has to be noted that, object patterns in this way defined follow an *open world semantics* typical of the Description Logics approach [Borgida *et al.*, 1989; Woods *et Schmolze*, 1989]. Objects of 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

In order to illustrate the way our approach works, we will use the following example of integration in the Restaurant Guide domain. To be general in illustrating the techniques, we suppose to deal with different kinds of sources, such as structured and semistructured.

Consider two differents datasources that collect information about restaurants. The `Eating Datasource` guidebook (`ED`) contains semistructured objects about restaurants of the west coast and their menu, quality, ... Fig. 1 illustrates a portion of the data. There is one complex root object with four complex children objects that represent restaurants. Each `Restaurant` has an atomic `name`, `category` and `specialty`. Furthermore, some `Restaurant` have an atomic `address` and some other a complex `address`, a `phone`, a complex object `nearby`, that specifies the nearest restaurant, and `owner`, that indicates the `name`, the `address` and the `job` of the restaurant's owner.

Food Guide Database (FD)

```
Steakhouse(s_code, name, street, pers_id,
      special_dish)
Bistro(s_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 (newspaper reviews, regional guidebooks, etc.). There are four relations: `Steakhouse`, `Bistro`, `Person`, and `Brasserie` (see Fig. 3). Information related to restaurant is maintained into the `Steakhouse` relation. `Bistro` instance is a subset of `Steakhouse` instance and contains information about the small informal restaurants that serve wine. Each `Steakhouse` and `Bistro` is managed by a `Person`. Information about places where drinks and snacks are served on, are stored in `Brasserie` relation.

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

In order to support semantic integration of information sources, we introduce an object-oriented language, called ODL$_{I^3}$, for a semantically rich representation of conceptual schemas and object patterns associated with information sources.

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. expressing interschema knowledge for the extracted source schemas. They are example of intensional assertions for the sources [Catarci *Lenzerini*, 1993]. Terminological relationships can be defined for class and attributes, and are specified by considering their 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). An example of SYN relationship in our running example is ⟨`ED.Restaurant` SYN `FD.Steakhouse`⟩.[1]
- 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. An example of BT relationship in our example is ⟨`FD.Steakhouse` BT `FD.Bistro`⟩.
- 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. For example, we can have the following relationship ⟨`ED.Restaurant` RT `ED.Owner`⟩.

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.

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} is straightforward, on the basis of the ODL_{I^3} syntax (see Appendix A) and of the object pattern definition. In particular, given a pattern $\langle l, A\rangle$ or a relation of a relational source, translation is performed as follows: 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. As an example, `ED.Restaurant` and `FD.Steakhouse` will be described as follows:

```
interface Restaurant
( source semistructured Eating_Datasource )
{  attribute string      name;
   attribute string      address;
   attribute integer     phone*;
   attribute set<string> specialty;
   attribute string      category;
   attribute Restaurant  nearby*;
   attribute Person      owner*;
   };

interface Steakhouse
( source relational Food_Guide )
  key s_code
  foreign_key(pers_id) references Person )
{  attribute string  s_code;
   attribute integer name;
   attribute string  street;
   attribute string  pers_id;
   attribute string  special_dish; };
```

[1]We use dot notation for specifying the source where a given term is used.

Union and optional constructors are used for object patterns. In particular, the union constructor is used in presence of heterogeneous objects in the source for a given object pattern. With reference to our semistructured source ED of Fig. 1, consider the semistructured object restaurant 2:
$\langle$2, `restaurant`, {(6, `name`), (7, `address`), (8, `specialty`), (9, `phone`), (10, `category`)}$\rangle$.
Consider also the restaurant object 3, and the non-atomic `address` object 13 to which it refers:
$\langle$13, `address`, {(25, `street`), (26, `city`), (27, `zipcode`)}$\rangle$.

The `Address` object has a different structure in the source, that must be reflected in the extracted object patterns. In particular, an object pattern is extracted for address, `Address-pattern` = (`Address`, {`city`, `street`, `zipcode`}) to capture the complex structure of addressed for some restaurants. Moreover, to take into account the address heterogeneity, we use the constructor **union** in the in ODL_{I^3} representation of the `Address` object pattern. Resulting ODL_{I^3} specifications of `Restaurant` and `Address` object patterns are shown in Figure 4.

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

interface Restaurant
( source semistructured Eating_Datasource )
{  attribute string   name;
   attribute Address  address;
   ............................... };
```

Figure 4: An example of union constructor

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 Logic.

2.3 The OLCD Description Logic

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, 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 [Beneventano *et al.*, 1998]. OLCD is an extension of the *object description language* ODL (not to be confused with ODL-ODMG), introduced in [Bergamaschi *Nebel*, 1994] 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.

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* ($\sqcup$) constructor. 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 with 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 [Beneventano *et al.*, 1998], 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 normal attribute domain (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[2]:

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

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

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

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 [Beneventano *et al.*, 1997], based on OLCD and implementing the above reasoning techniques.

3 Reasoning about ODL_{I^3} schema descriptions using OLCD

To develop intelligent techniques for semantic integration, it is important to have at disposal a shared ontology for the information sources to be integrated. The ontology provides a reference vocabulary on which to base the identification of heterogeneity and the subsequent resolution for integration.

To provide a shared ontology for the sources, we exploit Description Logics capabilities to construct a *Common Thesaurus* of terminological relationships, describing common knowledge about ODL_{I^3} classes and attributes of source schemas. Terminological relationship represent synonymy (SYN), hypernymy (BT), hyponymy (NT) and positive associations (RT) between class and attribute names, as already described in the previous section. 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 following steps.

3.1 Extraction of relationships from ODL_{I^3} schema descriptions

By exploiting ODB-Tools capabilities and semantically rich schema descriptions, an initial set of BT, NT, and RT can be automatically extracted from source schemas.

[2] σ_P and σ_V introduces primitive types and virtual types respectively

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. 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 [Miller , 1995] lexical system. In this case, synonyms, hypernyms/hyponyms, and related terms for terms appearing in source schemas 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` datasources. The set of terminological relationships automatically extracted by ODB-Tools are the following:

⟨`ED.Restaurant` RT `ED.Owner`⟩,
⟨`ED.Restaurant` RT `ED.Address`⟩,
⟨`ED.Restaurant` RT `ED.Restaurant`⟩,
⟨`FD.Steakhouse` RT `FD.Person`⟩,
⟨`FD.Bistro` RT `FD.Person`⟩.
The relationships derived from WordNet are the following:

⟨`ED.Restaurant` BT `FD.Steakhouse`⟩,
⟨`ED.Restaurant` BT `FD.Bistro`⟩,
⟨`ED.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`⟩.

In addition, new relationships can be supplied directly by the designer, to capture specific domain knowledge about the source schemas (e.g., new synonyms).

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

3.2 Integration/Revision of relationships

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 schema. 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, suppose that a strong relationship (SYN) be defined between the two classes (having different structures) `ED.Restaurant` and `FD.Steakhouse` (see Appendix B). 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:

```
interface Restaurant
(...)
{  attribute string      name;
   attribute Address     address;
   attribute integer     phone*;
   attribute set<string> specialty;
   attribute string      category;
   attribute Restaurant  nearby*;
   attribute Person      owner*;
   attribute string      street;
   attribute string      pers_id;
   attribute string      special_dish;  };

interface Steakhouse
(...)
{  attribute string      s_code;
   attribute integer     name;
   attribute string      street;
   attribute string      pers_id;
   attribute string      special_dish;
   attribute Address     address;
   attribute integer     phone*;
   attribute set<string> specialty;
   attribute string      category;
   attribute Restaurant  nearby*;
   attribute Person      owner*; };
```

The introduction by the designer of this new relationship will lead to the discovery of a new relationship between `FD.Brasserie` and `FD.Steakhouse` as shown in subsection 3.4.

3.3 Relationship validation

In this step, ODB-Tools is employed to validate terminological relationships defined for 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's name in the Thesaurus using ODB-Tools:

- $\langle n_t$ 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 is defined using the `union` constructor, a *valid relationship* is recognized if at least one domain of d_t is compatible with d_q.

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.Restaurant.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.Restaurant.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 by ODB-Tools, by exploiting the "virtual schema" defined in the revision/integration step and by deriving new generalization and aggregation relationships.

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.Steakhouse` RT `ED.Address`⟩,
⟨`FD.Steakhouse` BT `FD.Brasserie`⟩,
⟨`ED.Restaurant` RT `ED.Address`⟩,
⟨`FD.Steakhouse` RT `ED.Restaurant`⟩,
⟨`FD.Steakhouse` RT `ED.Owner`⟩,
⟨`FD.Brasserie` RT `FD.Person`⟩,

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` datasources is reported in Fig. 5, where solid lines represent explicit relationships (i.e., extracted/supplied), dashed lines represent inferred relationships, and superscripts indicate their kind.[3]

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

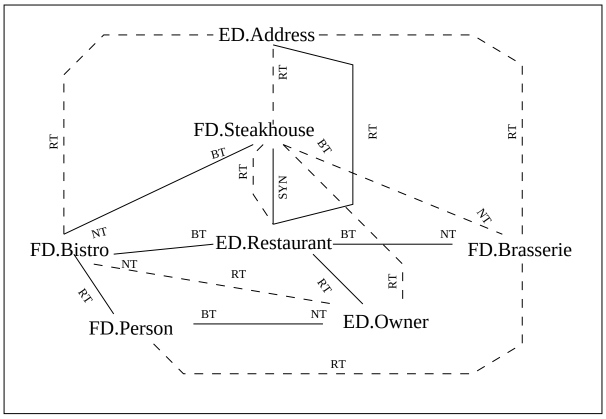

Figure 5: Common Thesaurus for `Eating` and `Food` Datasource

Note that, due the simplicity of the adopted example, many of the discovered relationships are trivial, except for the BT relationship between `FD.Brasserie` and `FD.Steakhouse`.

ODB-Tools performs validation and inference steps by exploiting subsumption (i.e., generalization) and equivalence computation. As we described in [Beneventano *et al.*, 1998; Bergamaschi *Nebel*, 1994], the computation of subsumption and equivalence between type of OLCD are PSPACE-hard problems. In the presence of general schema axioms (e.g., C=D where C and D are complex type descriptions) it may be argued that these problems are at least as difficult as the satisfiability problem of general ALC-schemata (with, possibly cyclic, inclusion statement and concept definition), which is EXPTIME-complete [Donini *et al.*, 1993; Buchheit *et al.*, 1998]. On the othe hand, as shown in [Bergamaschi *Nebel*, 1994], even if from a purely theoretical point of view this computation is intractable, 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 with ODL$_{I^3}$ and OLCD

In this section, we describe the information integration technique, to construct the 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

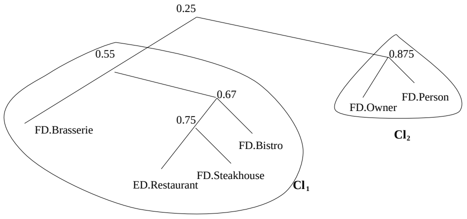

Figure 6: The Affinity tree

knowledge in the Common Thesaurus. Moreover, it supports semi-automatic synthesis of clusters of semantically related ODL$_{I^3}$ classes, by handling properly semantic heterogeneity.

4.1 Clustering ODL$_{I^3}$ classes

Providing an integrated representation of heterogeneous information requires to determine if the source schemas contain semantically related ODL$_{I^3}$ classes, that is, classes describing to the same or similar real-world concept(s).

We exploit the knowledge in the Common Thesaurus to assess the level of semantic relationship between ODL$_{I^3}$ classes. For this purpose, a set of *affinity coefficients* (i.e., numerical values in the range $[0, 1]$) are evaluated for all possible pairs of ODL$_{I^3}$ classes, based on the (valid) terminological relationships in the Common Thesaurus. Affinity coefficients determine the degree of semantic relationship of two classes based on their names (Name Affinity coefficient) and their attributes (Structural Affinity coefficient). A comprehensive value of affinity, called *Global Affinity* coefficient, is finally determined as the linear combination of the Name and Structural Affinity coefficients.

Global affinity coefficients are used by a hierarchical clustering algorithm, to classify ODL$_{I^3}$ classes according to their degree of affinity. The output of the clustering procedure is an affinity tree, where ODL$_{I^3}$ classes are the leaves and intermediate nodes have an associated affinity value, holding for the classes in the corresponding cluster. The affinity-based evaluation and clustering procedures are performed with the help of the ARTEMIS tool environment (for a detailed description see [Castano *De Antonellis*, 1999; 1997; Bergamaschi *et al.*, 1998]). The affinity tree obtained for our example is shown in Fig. 6.

Clusters for integration are interactively selected from the affinity tree using a threshold based mechanism. In Fig. 6, selected clusters, namely Cl_1 and Cl_2, are evidenced. For each selected cluster in the tree, a global class gc_i representative of the classes contained in the cluster is defined. gc_i is defined in way that it provides a synthezied, unified view of all the classes belonging to the corresponding cluster.

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 be treated properly to come up with an integrated and uniform representation at the global level.

The generation of gc_i is interactive with the designer. Let Cl_i be a selected cluster in the affinity tree. First, we associate with gc_i a set of global attributes, corresponding to the union of the attributes of the classes belonging to Cl_i. In particular, 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 Fig. 6 produces the following set of global attributes:
`name, address, phone*, specialty, category, nearby*, owner*, special_dish, street, type, s_code, pers_id`

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:

  And correspondence: this specifies that a global attribute corresponds to the concatenation of two or more attributes of a class $c_h \in \mathtt{Cl_i}$.
  For example, defining the mapping rule for the global attribute `name` of $\mathtt{Cl_1}$, the designer may specifies that the 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`, we state 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.

  Or correspondence: this specifies that a global attribute corresponds to at most one attribute of a class $c_h \in \mathtt{Cl_i}$. 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 that classes in the cluster Cl_1 have and additional `menu_price` attribute. Suppose

```
interface Food_Place
{ attribute name
    mapping_rule ED.Restaurant.name,
                 FD.Steakhouse.name,
                 FD.Brasserie.name;
    ...
  attribute category
    mapping_rule ED.Restaurant.category,
                 FD.Bistro.type;
  attribute zone
    mapping_rule ED.Restaurant = 'Pacific',
                 FD.Steakhouse = 'USA',
                 FD.Bistro = 'USA',
                 FD.Brasserie = 'USA';
}
```

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

also that values of this attribute are in US Dollars for `Restaurant` class and in Italian Lire for remaining classes. Here, `country` is the tag attribute. In this example, it is possible to define an *or* correspondence between the attributes `Italian_menu_price` and `US_menu_price` by declaring the following mapping rule:

```
...
attribute integer menu_price
 mapping rule(S.Brasserie.Italian_price union
              S.Restaurant.US_price on Rule1),
              ...
...
rule Rule1 { case of S.Restaurant.country:
     ''Italy'' : S.Brasserie.Italian_price;
     ''US''    : S.Restaurant.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_2, the mapping rule defined for the global attribute `zone` specifies that the objects of the class `ED.Restaurant` regarding the "Pacific Area", while objects of `FD.Steakhouse` and `FD.Bistro` whereever in the USA.

For each ODL_{I^3} global class gc_i a persistent mapping table is generated. As an example, the mapping table for the `Food_Place` class is shown in Fig. 8.

Food_Place	**code**	**name**	**...**	**zone**
ED.Restaurant	null	name	...	'Pacific'
FD.Steakhouse	s_code	name	...	'USA'
FD.Bistro	s_code	null	...	'USA'
FD.Brasserie	b_code	name	...	'USA'

Figure 8: `Food_Place` mapping table

5 Architecture of the support 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 [ARPA, 1997]: at the bottom layer are the schema of information sources and the above layers 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 [Beneventano *et al.*, 1998; Bergamaschi *Nebel*, 1994] which performs schema validation for the generation of the Common Thesaurus and query optimization [Beneventano *et al.*, 1997].
- The *ARTEMIS Tool Environment*, a tool based on affinity-based clustering techniques which performs ODL_{I^3} class analysis and clustering [Castano *De Antonellis*, 1997; 1999].

6 Related work and discussion

Works related to the issues discussed in this paper are in the area of semistructured data and of 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 [Buneman , 1997]. Two similar models for semistructured data have been proposed [Papakonstantinou *et al.*, 1995], based on rooted, labeled graph with the objects as nodes and labels on edges. According to the model presented in [Buneman *et al.*, 1996], information resides at labels only, while according to the "Object Exchange Model" (OEM) proposed by Papakonstantinou et. al. in [Papakonstantinou *et al.*, 1995], information also resides at nodes.

The issue of extracting structure from semistructured data, which is more directly concerned with our concept of object pattern, has also been investigated. In particular, in [Goldman *Widom*, 1997], the notion of *dataguide* has been proposed as a "loose description of the structure of the data" actually stored in an information source. In [Buneman , 1997], a new notion of schema appropriate for semistructured data has been proposed, represented as edge-labeled graph. Heuristic techniques to extract a type hierarchy for a semistructured data source have been presented in [Nestorov *et al.*, 1997], together with rules for classifying semistructured objects against the extracted type hierarchy. Our notion of object pattern has some analogies with the notion of type in this latter work, altough we do not introduce hierarchies. Object patterns represent in a unified way all possible heterogeneous objects with a ceratin label in the source. By considering all defined object patterns, we can represent the structure of the source as an edge-labeled graph, in analogy with proposals of the literature. In the paper we focused on the representation of object patterns by means of Description Logics rather than on algorithms for extracting object patterns. Techniques similar to the ones proposed in the literature for the extraction of the structure from semistructured sources can be used also for object pattern extraction. The main contribution of our paper is related to show the usage of object patterns for integration purposes. In the literature, extraction of the structure from a semistructured source has been studied for query optimization purposes on a single source. 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 used 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 [Arens *et al.*, 1996; Chawathe *et al.*, 1994; Carey *et al.*, 1994]. MOMIS is based on a mediator architecture and follows the 'semantic approach'.

Following the classification of integration system proposed by Hull [Hull , 1997], 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 [Levy *et al.*, 1996] such as CLASSIC [Borgida *et al.*, 1989]. All of the virtual approaches are based on a model of query decomposition, sending subqueries to source databases, and merging the answers that come back. Recent systems based on description logics are focused primarily on conjunctive queries (i.e., expressible using select, project and join), and have more the flavor of the *Open World Assumption* - the answer provided through an integrated view will hold a subset of the complete answer that is implied by the underlying databases. For the schema, a "top-down" approach is used: in essence a global schema encompassing all relevant information is created, and data held in the source databases is expressed as views over this global schema [J. D. Ullman , 1997].

With references to the same classification proposed by Hull, MOMIS is in the category of "read-only views", i.e. systems whose task is to support an integrated, read-only, view of data that resides in multiple databases. The most similar projects are: GARLIC, SIMS, Information Manifold and Infomaster.

The GARLIC project [Carey *et al.*, 1994] builds up on a complex wrapper architecture to describe the local sources with an OO language (GDL), and on the definition of Garlic Complex Objects to manually unify the local sources to define a global schema.
The SIMS project [Arens *et al.*, 1996] proposes to create a global schema definition exploiting the use of description logics (i.e. the LOOM language) for describing information sources. The use of a global schema allows both GARLIC and SIMS projects to support every possible user queries on the schema instead of a predefined subset of them.

Information Manifold Systems [Levy *et al.*, 1996], as the MOMIS project, provides a source independent, query independent mediator. The input schema of an Information Manifold System is a set of descriptions of the sources; so, given a query, the system will create a plan for answering the query using the source. The algorithms to decide the useful information sources and to generate the query plan are provided [Levy *et al.*, 1996]. With respect to the input schema generation, it is completely modeled by the user, while in our approach it is system-guided.

Infomaster System [Genesereth *et al.*, 1997] provides integrated access to multiple distributed heterogenuos information sources giving the illusion of a centralized, homogeneous information system. It is based on a global schema, completely modeled by the user, and a core system that dynamically determines an efficient plan to answer the user's queries by using translation rules that harmonizing heterogeneous sources.

An other proposal based on the Description Logic and Reasoning techniques is described in [Calvanese *et al.*, 1997], where a *declarative approach* (semantic approach) is used. The framework provide *intermodel assertions* to define inter-relationships between concepts in differents sources. The *intermodel assertions* may be defined both at intensional (similary to our terminological relationships) and extensional level. In the proposal, the definition of the global schema (called *Enterprise Model*) is a manual task.

On the other hand, other projects are based on a 'structural' approach. The TSIMMIS project [Chawathe *et al.*, 1994] 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 weather and how to use the new sources.

7 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 Logic-based ODL_{I^3} language is introduced for information extraction and integration, by taking into account also semistructured information sources.

Future research work will be devoted to the integration of XML data sources.

References

[Abiteboul *et al.*, 1996] 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.

[Arens *et al.*, 1996] 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.

[ARPA, 1997] ARPA, "ARPA I^3 Reference Architecture", Available at http://www.isse.gmu.edu/I3_Arch/index.html

[Beneventano *et al.*, 1997] 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)

[Beneventano *et al.*, 1998] 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.

[Bergamaschi *et al.*, 1998] 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.

[Bergamaschi *et al.*, 1999] 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.

[Bergamaschi *Nebel*, 1994] 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.

[Borgida *et al.*, 1989] 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.

[Buchheit *et al.*, 1998] M. Buchheit, F. M. Donini, W. Nutt and A. Schaerf, "A Refined Architecture for Terminological Systems: Terminology = Schemas + Views", em in Artificial Intelligence, Vol. 99, n. 2, pages 209-260, March 1998.

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

[Buneman *et al.*, 1996] 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.

[Calvanese *et al.*, 1997] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi and R. Rosati, "A Schema Analysis and Reconciliation Tool Environment for Heterogeneous Databases", in *In Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR-98)*, pages 2-13, 1998.

[Castano *De Antonellis*, 1997] S. Castano, V. De Antonellis, "A Schema Analysis and Reconciliation Tool Environment for Heterogeneous Databases", to appear in *IEEE Proc. of IDEAS'99, International Database Engineering and Applications Symposium*, Montreal, Canada, August 1999.

[Castano *De Antonellis*, 1999] 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.

[Catarci *Lenzerini*, 1993] 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.

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

[Chawathe *et al.*, 1994] 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.

[Donini *et al.*, 1993] F. M. Donini, A. Schaerf and M. Buchheit, "Decidable Reasoning in Terminological Knowledge Representation Systems", *in 13th International Joint Conference on Artificial Inteligence*, 1993.

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

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

[Carey *et al.*, 1994] 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.

[Goldman *Widom*, 1997] 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.

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

[Levy *et al.*, 1996] 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.

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

[Nestorov *et al.*, 1997] S. Nestorov, S. Abiteboul and R. Motwani, "Inferring Structure in Semistructured Data", SIGMOD Record 26(4): 39-43 (1997).

[Papakonstantinou *et al.*, 1995] 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.

[J. D. Ullman , 1997] J. D. Ullman. Information integration using logical views. *Intl. Conf on Database Theory*, pages 19-40, 1997.

[Woods *et Schmolze*, 1989] 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⟩
...		
⟨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**

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⟩ **;**

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

```
Eating_Datasource (ED):

interface Restaurant
( source semistructured Eating_Datasource )
{  attribute string          name;
   attribute Address         address;
   attribute integer         phone*;
   attribute set<string>     specialty;
   attribute string          category;
   attribute Restaurant      nearby*;
   attribute Owner           owner*; };

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

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

Food_Guide_Datasource (FD):

interface Steakhouse
( source relational Food_Guide
  key s_code
  foreign_key(pers_id) references Person )
```

```
{  attribute string         s_code;
   attribute string         name;
   attribute string         street;
   attribute integer        pers_id;
   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 s_code
  foreign_key(s_code) references Steakhouse,
  foreign_key(pers_id) references Person)
{  attribute string      s_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; };
```