Semantic Integration of Heterogeneous Information Sources ⋆

S. Bergamaschi [a,b], S. Castano [c], M. Vincini [a] and D. Beneventano [a,b]

[a] *University of Modena and Reggio Emilia, Modena*
e-mail: {sonia.bergamaschi,domenico.beneventano,maurizio.vincini}@unimo.it

[b] *CSITE-CNR, Bologna*

[c] *University of Milano, Milano*
e-mail: 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 Logic, 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 Common Thesaurus for the sources. Information integration is then performed in a semi-automatic way, by exploiting the knowledge in the Common Thesaurus and ODL$_{I^3}$ descriptions of source schemas with a combination of clustering techniques and Description Logics. This integration process gives rise to a virtual integrated view of the underlying sources for which mapping rules and integrity constraints are specified to handle heterogeneity. Integration techniques described in the paper are provided in the framework of the MOMIS system, based on a conventional wrapper/mediator architecture.

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

⋆ 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.

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 describing the same real-world concept in different sources, and to semantic heterogeneity. In fact, information sources available in global information systems are pre-existing 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 during integration in order to effectively and correctly exploit the information available at the sources.

Integration and reconciliation of data coming from heterogeneous sources is a hot research topic in databases [28]. Several contributions appeared in the recent literature, including methods, techniques and tools for integrating and querying heterogeneous databases [15,26,30,36]. The integration of structured and semistructured data sources presents new problems and challenges: in this case, the heterogeneity concerns 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 (e.g., Web documents) calls for the development of methods, techniques, and languages for this new type of data [10,12,14]. 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, integrated descriptions (i.e., *a global virtual view*) of the information coming from multiple heterogeneous sources, to provide the user with a uniform query interface against the sources independent from their location and the level of heterogeneity of their data. Moreover, to meet the requirements of global, Internet-based information systems with a possibly high number of sources to be integrated, it is important to develop tool-assisted techniques, to make information extraction and integration activities semi-automatic as much as possible.

In this paper, we propose intelligent, semi-automated techniques for heterogeneous information extraction and integration, by taking into account both semistructured and structured data sources. Information extraction techniques have the goal of producing a semantically rich representation of source schemas in ODL_{I^3}, an object-oriented language derived from the standard ODMG, with the underlying Description Logic OLCD (Object Language with Complements allowing Descriptive cycles) [5,8], derived from the KL-ONE family [39]. In case of semistructured information sources, information extraction produces also *object patterns*, that are used as schema information for the source to generate the corresponding ODL_{I^3}

description.

Information integration techniques have the goal of producing global, integrated ODL_{I^3} descriptions of the sources in a semi-automated way. Starting from extracted ODL_{I^3} descriptions of the information sources, first a shared Common Thesaurus of validated inter-source terminological relationships is set up, by exploiting the WordNet lexical system [32] and the OLCD Description Logic inference capabilities. Based on the relationships in the Common Thesaurus, affinity coefficients are evaluated which give a measure of the level of matching of ODL_{I^3} classes of different sources for integration. Candidates to integration are identified by applying affinity-based clustering to ODL_{I^3} schemas. Global ODL_{I^3} classes are derived from selected clusters to provide an integrated description of the whole cluster. Moreover, mapping rules are defined for global ODL_{I^3} classes to express the relationships holding between them and ODL_{I^3} source descriptions, respectively.

Techniques described in the paper are provided in the framework of the MOMIS (Mediator envirOnment for Multiple Information Sources) [7] project [1] . Like other integration projects [1,15,30], MOMIS follows a "semantic approach" to information integration based on the conceptual schemas of the information sources, and on a mediator component. To provide an I^3 [2] architecture for integration and query optimization, the mediator component relies on two tools, namely ARTEMIS [17], developed by University of Milano and University of Brescia, and ODB-Tools [6], developed by University of Modena e Reggio Emilia.

The paper is organized as follows. In Section 2, we describe the information extraction techniques with the ODL_{I^3} language, and we introduce the OLCD Description Logic and its inference capabilities. In Section 3, we describe the use of OLCD to provide a Common Thesaurus of terminological relationships describing inter-source knowledge. In Section 4, we describe the integration techniques for building the mediator global schema of considered sources. In Section 5, we describe the architecture of the MOMIS system and experimentation issues. In Section 6, we make comparison with related work. Finally, in Section 7, we give our concluding remarks.

2 Information extraction with ODL_{I^3}

An important goal of information extraction is the construction of a semantically rich representation of the information sources to be integrated by means of a com-

[1] MOMIS is a joint project among the Università di Modena e Reggio Emilia, the Università di Milano, and the Università di Brescia within the national research project INTERDATA, theme n.3 "Integration of Information over the Web", coordinated by V. De Antonellis, Università di Brescia.

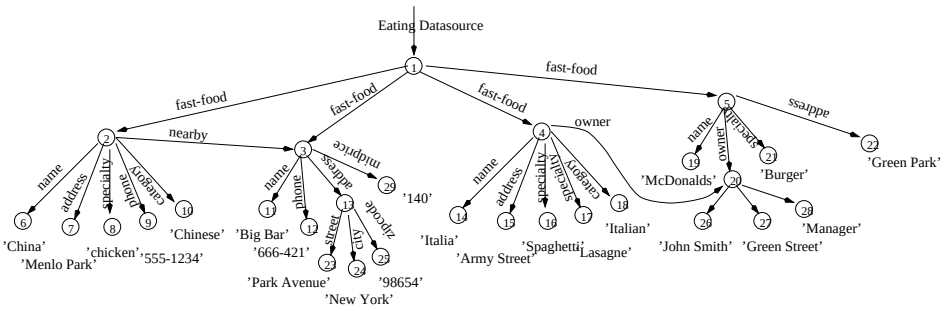

Fig. 1. Eating Source (ED)

mon data model. In semantic approaches to integration, this task is performed by wrapper tools developed for each kind of data representation, by considering the conceptual schema of a given 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. 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 [24] for relational to ODMG schema conversion. For semistructured information sources (e.g., Web data 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 [10].

One of the goals of information extraction for integration when semistructured information sources are involved is to derive and explicitly represent also the schema of the source. For this purpose, we proceed as follows. According to the models proposed in literature for semistructured information sources [10,36], a semistructured source is represented as a rooted, labeled graph where nodes contain data (e.g., an image or free-form text) and labelled edges describe the concept represented by data in the corresponding node. Figure 1 shows an example of graph-based representation of a semistructured source, called `Eating Source`, containing information related to local fast food.

In the graph model, 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 pairs (*id*,*label*), where *id* is an object identifier. 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. With reference to the source in Figure 1, there is one complex root object with four complex children objects that represent

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})

Fig. 2. Object patterns for the ED source

fast-foods. Each Fast-Food object has an atomic object name, category and specialty. Furthermore, some Fast-Food objects have an atomic address while some other a complex object address, an atomic phone, a complex object nearby (that specifies the nearest fast-food), and a complex object owner, specifying the name, the address and the job of the fast-food's owner.

To represent the schema of a semistructured source S, we introduce the notion of *object pattern*. In semistructured data models, labels are descriptive as much as possible. Furthermore, the same label is generally assigned to all objects describing the same concept in S. 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 2.1 (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 concept in S. In particular, label l of an object pattern denotes the concept and set A of an object pattern denotes the properties (or attributes) characterizing the concept in the source. Since semistructured objects can be heterogeneous, labels in A correspond to child object that 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 "*".

With respect to the ED source of Figure 1, three object patterns are extracted (see Figure 2): Fast-Food, representing objects describing eating places; Owner representing objects describing people involved; Address, representing objects describing addresses. The extraction process produces also the Address pattern to take into account the different structure of the Address objects in the ED source (i.e., in semistructured objects 2, 4, and 5 address is atomic while in object 3 it is complex).
An object pattern description follows an *open world semantics* typical of the Description Logic approach [39]. 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

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)
```

Fig. 3. Food Guide Database (FD)

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

We consider two sources in the Restaurant Guide domain, storing 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. A portion of this source has been already shown in Figure 1. 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). The schema of this source is composed of four relations (see figure 3), namely, Restaurant, Bistro, Person, and Brasserie. 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 source schemas and object patterns associated with information sources to be integrated, we introduce an object-oriented language, called ODL$_{I^3}$. According to recommendations of ODMG and to the diffusion of I^3/POB [2,20], 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 schemas of structured and semistructured data sources 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 an 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.

Integrity constraint rules. This kind of rule is introduced in ODL$_{I^3}$ in order to express, in a declarative way, *if then* integrity constraint rules at both intra- and inter-source level.

Intensional relationships. They are *terminological relationships* expressing inter-schema knowledge for the source schemas. Intensional 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. BT relationship is not symmetric. The opposite of BT is NT (Narrower Terms), or hyponymy.
- 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.

An intensional relationships is only a terminological relationship, with no implications on the extension/compatibility of the structure (domain) of the two involved classes (attributes). Consequently, our notion of intensional relationships is different from the one proposed by Catarci and Lenzerini [19], where an intensional relationships has some extensional import.

Extensional relationships. Intensional relationships SYN, BT and NT between two classes C_1 and C_2 may be "strengthened" by establishing that they are also *extensional* relationships [19]. Consequently, the following extensional relationships can be defined in ODL$_{I^3}$:

- C_1 SYN$_{ext}$ C_2: this means that the instances of C_1 are the same of C_2.
- C_1 BT$_{ext}$ C_2: this means that the instances of C_1 are a superset of the instances of C_2.
- C_1 NT$_{ext}$ C_2: this means that the instances of C_1 are a subset of the instances of C_2.

Moreover, extensional relationships "constrain" the structure of the two classes C_1 and C_2 , that is C_1 NT$_{ext}$ C_2 is semantically equivalent to an "isa" relationship. As to summarize:

- an extensional relationship C_1 NT$_{ext}$ C_2 is equivalent to an "isa" relationship C_1 ISA C_2 plus an intensional relationships C_1 NT C_2;
- an extensional relationship C_1 BT$_{ext}$ C_2 is equivalent to an "isa" relationship C_2 ISA C_1 plus an intensional relationships C_1 BT C_2;

- an extensional relationship C_1 SYN$_{ext}$ C_2 is equivalent to two "isa" relationships C_1 ISA C_2 and C_2 ISA C_1 plus an intensional relationships C_1 SYN C_2.

An "isa" relationships C_1 ISA C_2 is expressible in ODL$_{I^3}$ by the following integrity constraint rule:

```
rule Rule2 forall X in C1 then X in C2
```

Mapping Rules. This kind of rule is introduced in ODL$_{I^3}$ in order to express relationships holding between the integrated ODL$_{I^3}$ schema description of the information sources and the ODL$_{I^3}$ schema description of the original sources. These rules will be illustrated in detail in Section 4, together with examples of use.

The extraction process has the goal of translating object patterns and source schemas 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 relation of a relational source or a pattern $\langle l, A \rangle$, translation involves the following steps: i) an ODL$_{I^3}$ class name corresponds to the relation name or to l, respectively, and ii) for each relation attribute or label $l' \in A$, an attribute is defined in the corresponding ODL$_{I^3}$ class. Furthermore, attribute domains are extracted. Structure extraction can be performed as proposed in [11,33].

As an example, below we report the ODL$_{I^3}$ representation of the `ED.Fast-Food` object pattern and of the `FD.Restaurant` relation. The complete ODL$_{I^3}$ schemas representation of the `ED` and `FD` sources are reported in Appendix B.

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

interface FD.Restaurant
( source relational Food_Guide )
key r_code
foreign_key(pers_id) references Person )
{  attribute string         r_code;
    attribute string         name;
    attribute string         street;
    attribute string         zip_code;
    attribute integer        pers_id;
```

```
    attribute string        special_dish;
    attribute integer       category;
    attribute integer       tourist_menu_price; };
```

To represent object patterns in ODL$_{I^3}$, union and optional constructors are used. In particular, the `union` constructor is used to represent object patterns describing heterogeneous objects in the source. An example of use of the `union` constructor in the ODL$_{I^3}$ class representing the `Address` pattern of the `ED` source (see Figure 2 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; };
```

Fig. 4. An example of `union` constructor in ODL$_{I^3}$

2.3 The OLCD Description Logic

ODL$_{I^3}$ descriptions are translated into OLCD (*Object Language with Complements allowing Descriptive cycles*) descriptions in order to perform Description Logics inferences that will be useful for semantic integration.

In this section, we give the syntax of OLCD (the semantics is given in Appendix C); Readers interested in a formal account can refer to [5].

Types and Schemas. We assume a countable set of symbols $\mathbf{A}$ of *attribute names* (denoted by $a, a_1, a_2, \ldots$) and we assume a countable set $\mathbf{N}$ of *type names* (denoted by N, N_1,N_2, $\ldots$), which includes the set $\mathbf{B} = \{$`Integer`, `String`, `Bool`, `Real`$\}$ of base-type designators (which will be denoted by B) and the symbols $\top$, $\bot$. A *path* p is either the symbol ϵ, or a dot-separated sequence of elements $e_1 . e_2 . \cdots . e_n$, where $e_i \in \mathbf{A} \cup \{\triangle, \exists\}$ $(i = 1, \ldots, n)$. ϵ denotes the unique path of length 0. Let $\mathbf{W}$ denote the set of all paths.

$\mathbf{S}(\mathbf{A}, \mathbf{N})$ denotes the set of all *finite type descriptions* (denoted by S, S_1, S_2, $\ldots$), also briefly called *types*, over given $\mathbf{A}, \mathbf{N}$, obtained according to the following abstract syntax rule, where $a_i \neq a_j$ for $i \neq j$ (in the sequel p, p_1, p_2, $\ldots$, denote a path, d denotes a base value, θ denotes a relational operator):

$$S \rightarrow N \mid S_1 \sqcup S_2 \mid S_1 \sqcap S_2 \mid \neg S \mid \{S\}_\forall \mid \{S\}_\exists \mid [a_1 : S_1, \ldots, a_k : S_k] \mid \triangle S \mid p\theta d \mid p\uparrow$$

$\top$ denotes the *top type*, $\bot$ denotes the *empty type*, $\{\}_\forall$ and $[\ ]$ denote the usual type constructors of set and record (tuple), respectively. The $\{S\}_\exists$ construct is an existential set specification, where at least one element of the set must be of type S. The construct $\sqcap$ stands for *intersection*, the construct $\sqcup$ stands for *union*, the construct $\neg$ stands for *complement*, whereas $\triangle$ constructs class descriptions, i.e., is an object set forming constructor. $p\theta d$, $p{\uparrow}$ represent *atomic predicates*: $p\theta d$ is a *range restriction* and $p{\uparrow}$ expresses *path undefinedness*.

Given a set of type descriptions $\mathbf{S}(\mathbf{A}, \mathbf{N})$, a *schema* σ over $\mathbf{S}(\mathbf{A}, \mathbf{N})$ is a total function $\sigma : \mathbf{N} \backslash (\mathbf{B} \cup \{\top, \bot\}) \rightarrow \mathbf{S}(\mathbf{A}, \mathbf{N})$, which associates type names to descriptions. σ is partitioned into two functions: σ_P, which introduces the description of primitive type names whose extensions must be explicitly provided by the user; and σ_V, which introduces the description of virtual type names whose extensions can be recursively obtained from the extension of the types occurring in their description.

In OLCD cyclic type names are allowed: in fact, since a type name may appear in type descriptions, we can have *circular references*, that is, type names which make direct or indirect references to themselves. Giving a *type as set* semantics to type descriptions, Description Logics, and thus OLCD, allows one to provide relevant reasoning techniques: computing *subsumption* relations between types (i.e. "is-a" relationships implied by type descriptions), deciding *equivalence* between types, and detecting *inconsistent* (i.e., always empty) types.

2.3.1 ODL$_{I^3}$ to OLCD translation

In this section, we describe how ODL$_{I^3}$ source schema descriptions are translated into OLCD descriptions.

ODL$_{I^3}$ classes. In general, a ODL$_{I^3}$ class is translated into a OLCD primitive class in a simple way: each attribute of the ODL$_{I^3}$ class becomes an attribute of the corresponding OLCD class.

For example, the `Restaurant` ODL$_{I^3}$ class is translated as follows:

σ_P(`ES.Restaurant`)=$\triangle$[`r_code`: `String`, `name`: `String`, `street`: `String`,
`zip_code` : `String`, `pers_id` : `Integer`, `special_dish` : `String`,
`category` : `Integer`, `tourist_menu_price` : `Integer`]

Some aspects of an ODL$_{I^3}$ class declaration, such as `key r_code` in the `Restaurant` ODL$_{I^3}$ class, are not translated into OLCD, but will be used in the semantic information integration.

Union constructor. The `union` constructor of ODL$_{I^3}$ is translated using the construct $\sqcup$ of OLCD; for example, the `Address` pattern of figure 4 is translated in OLCD as follows:

$$\sigma_P(\texttt{ES.Address}) = \triangle\Big(\texttt{String} \sqcup [\texttt{city}: \texttt{String}, \texttt{street}: \texttt{String}, \texttt{zipcode}: \texttt{String}]\Big)$$

Optional constructor. The construct $\sqcup$ is also used 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 expressed 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 example, in the `Fast_Food` interface, the optional attributes are translated as follows:

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

Integrity constraint rules. An *if then* integrity constraint rule is integrated into an OLCD class description, by using the $\sqcap$, $\sqcup$ and $\neg$ constructs. For example, the rule:

```
rule Rule1 forall X in Restaurant :
   (X.category > 5) then X.tourist_menu_price > 100;
```

is added to the `ES.Restaurant` description as follows:

$$\begin{aligned}\sigma_P(\texttt{ES.Restaurant}) = \triangle\Big([\,&\texttt{r\_code} : \texttt{String}, \texttt{name} : \texttt{String}, \texttt{street} : \texttt{String},\\ &\texttt{zip\_code} : \texttt{String}, \texttt{pers\_id} : \texttt{Integer}, \texttt{special\_dish} : \texttt{String},\\ &\texttt{category} : \texttt{Integer}, \texttt{tourist\_menu\_price} : \texttt{Integer}\,] \sqcap\\ &(\neg(\texttt{category} > 5) \sqcup (\texttt{tourist\_menu\_price} > \texttt{100}))\Big)\end{aligned}$$

Then, in our framework, integrity constraints are statements about the world and not about the contents of the database as in Reiter's approach [37]. In other words, a schema is composed by classes + integrity constraints and we check the consistency of such a schema.

Intensional relationships. They are not translated.

Extensional relationships. An "isa" relationships C_1 ISA C_2 related to an Extensional relationships and expressed in ODL$_{I^3}$ by the rule:

```
rule Rule2 forall X in C1 then X in C2
```

is integrated in the C_1 class description, by using the $\sqcap$ construct: $\sigma_P(C_1) = C_2 \sqcap \ldots$

Mapping Rules. They are not translated.

3 Reasoning about ODL$_{I^3}$ schema descriptions to build a Common Thesaurus

To develop intelligent techniques for semantic integration, inter-schema knowledge between information sources in the considered domain has to be identified and properly represented. For this purpose, we construct a *Common Thesaurus* of terminological intensional and extensional relationships, describing inter-schema knowledge about ODL$_{I^3}$ classes and attributes of source schemas. The Common Thesaurus provides a reference on which to base the identification of ODL$_{I^3}$ classes candidate to integration and subsequent derivation of their global represention.

In the Common Thesaurus, we express inter-schema knowledge in form of terminological relationships (SYN, BT, NT, and RT) and extensional relationships (SYN$_{ext}$, BT$_{ext}$, and NT$_{ext}$ between classes and/or attribute names.

The Common Thesaurus is constructed through an incremental process during which relationships are added in the following order:

(1) *schema-derived relationships*
(2) *lexical-derived relationships*
(3) *designer-supplied relationships*
(4) *inferred relationships*

All these relationships are added to the Common Thesaurus and thus considered in the subsequent phase of semantic information integration (see next section). Terminological relationships defined in each step hold at the intensional level by definition. Furthermore, in each of the above step the designer may "strengthen" a terminological relationships SYN, BT and NT between two classes C_1 and C_2 by establishing that they hold also at the extensional level, thus defining also an extensional relationship. The specification of an extensional relationship, on one hand, implies the insertion of a corresponding intensional relationship in the Common Thesaurus and, on the other hand, enable subsumption computation (i.e., inferred relationships) and consistency check between two classes C_1 and C_2.

3.1 Schema-derived relationships

In this step, we extract terminological and extensional relationships holding at intra-schema level by analyzing each ODL$_{I^3}$ schema separately. In particular, intra-schema 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 (e.g., see `Bistro` and `Restaurant` classes in the ODL$_{I^3}$ descriptions reported in Appendix B).

Example 1 Consider the ED and FD sources. A subset of intra-schema relationships automatically extracted is 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⟩.

In this case, the designer strengths the relationship ⟨FD.Restaurant BT FD.Bistro⟩ to hold also at the extensional level, leading to the definition of the following relationship: ⟨ FD.Restaurant BT_{ext} FD.Bistro ⟩.

3.2 Lexical-derived inter-schema relationships

In this step, terminological and extensional relationships holding at inter- schema level are extracted by analyzing ODL_{I^3} schemas together. The extraction of these relationships is based upon the lexical relations holding between classes and attributes names, deriving from the mining of used words. This is a kind of knowledge which is not based on the rules of a data definition language but derives from the name assigned by the designer. It is a designer's task to assign descriptive/meaningful names or, at least, correctly interpretable names. An interpretation uncertainty is therefore inherent to the language ambiguity.

Anyway knowledge associated with schema names is an opportunity that must be exploited to extract relationships. As it is almost impossible to carry out this task manually when the number and dimensions of schema grows, it was decided to experiment the use of WordNet [32] lexical system to extract intensional inter-schema relationships and propose them to the designer.

Example 2 Consider the ED and FD sources. The relationships derived using 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⟩,
⟨ED.Fast-Food.name BT FD.Person.first_name⟩,
⟨ED.Fast-Food.name BT FD.Person.last_name⟩.

Note that, SYN relationships are also extracted for the attributes having the same name in the two sources, which are omitted from previous set for the sake of brevity.

3.3 Designer-supplied inter-schema relationships

In this step, 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 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 can produce a wrong global schema. The following Relationship validation section shows how our system help the designer in detecting wrong relationships.

Example 3 In our example, the designer supplies the following relationships for classes and attributes:

$\langle$`ED.Fast-Food` SYN_{ext} `FD.Restaurant`$\rangle$,
$\langle$`ED.Fast-Food.category` BT `FD.Bistro.type`$\rangle$,
$\langle$`ED.Fast-Food.specialty` BT `FD.Bistro.special_dish`$\rangle$.

The definition of the relationship $\langle$`ED.Fast-Food` SYN_{ext} `FD.Restaurant`$\rangle$ by the designer implies the automated definition of the relationship $\langle$`ED.Fast-Food` SYN `FD.Restaurant`$\rangle$ in the Common Thesaurus.

3.4 Relationships validation

In this step, ODB-Tools is employed to validate intensional relationships between attribute names and extensional relationships between class names.

The validation of intensional relationships between attribute names is based on the compatibility of the domains associated with the attributes. This way, *valid* and *invalid* intensional 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 intensional relationships defined for attribute names in the Common 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 (see Figure 4), a *valid relationship* is recognized if at least one domain d_t (d_q) is compatible with d_q (d_t).

Example 4 Referring to our Common Thesaurus resulting from Examples 1 to 3, the output of the validation phase is the following (for each relationship, control flag $[1]$ denotes a valid relationship while $[0]$ an invalid one):

$\langle$`ED.Fast-Food.category` BT `FD.Bistro.type`$\rangle$ [0]
$\langle$`ED.Owner.name` BT `FD.Person.first_name`$\rangle$ [1]
$\langle$`ED.Owner.name` BT `FD.Person.last_name`$\rangle$ [1]
$\langle$`ED.Fast-Food.specialty` BT `FD.Bistro.special_dish`$\rangle$ [1]
$\langle$`ED.Fast-Food.name` BT `FD.Person.first_name`$\rangle$ [1]
$\langle$`ED.Fast-Food.name` BT `FD.Person.last_name`$\rangle$ [1]
$\langle$`ED.Fast-Food.category` SYN `FD.Restaurant.category`$\rangle$ [0]

As an extensional relationship between two classes C_1 and C_2 is integrated in the description of the class C_1, its validation is performed by checking the consistency of the class C_1. For example, the extensional relationship $\langle$`FD.Restaurant` BT_{ext} `FD.Bistro` $\rangle$ stated before by the designer is expressed in the `FD.Bistro` class description as follows:

σ_P(`FD.Bistro`) = `FD.Restaurant` $\sqcap$
$\triangle$ [`r_code` : `String`, `type` : `String`, `pers_id` : `Integer`]

Since the `FD.Bistro` class description is consistent, the relationship between `FD.Bistro` and `FD.Restaurant` is validated. On the other hand, the extensional relationship $\langle$`ED.Fast-Food` SYN_{ext} `FD.Restaurant`$\rangle$ is rejected, as the class description:

σ_P(`ED.Fast-Food`) = `FD.Restaurant` $\sqcap \ldots$

is inconsistent (the attribute `category` is defined in both the classes but on disjoint domains). In the presence of integrity rules less intuitive incoherencies may arise.

In this case, the designer modifies his statement and only the terminological relationship $\langle$`ED.Fast-Food` SYN `FD.Restaurant`$\rangle$ is kept in the Common Thesaurus.

3.5 Inferring new relationships

In this step, inference capabilities of ODB-Tools are exploited to infer new relationships, in order to set up a rich Common Thesaurus to support the identification

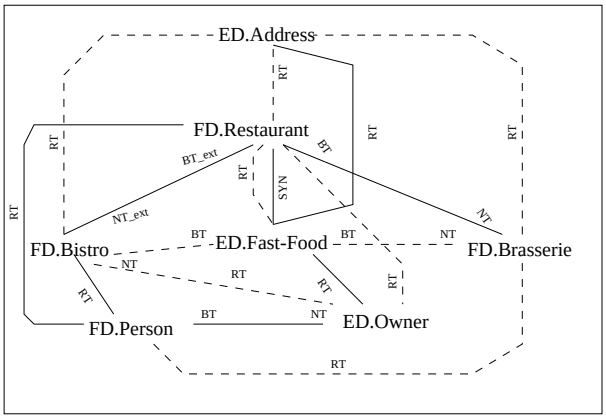

Fig. 5. Common Thesaurus for `Eating` and `Food` Source

of semantically similar ODL$_{I^3}$ classes in different sources, as will be shown in next section.

Example 5 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`.⟩

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 subscripts indicate their kind. [2]

Note that, due the simplicity of the adopted example, many of the discovered relationships are trivial. Suppose to introduce a new pattern into the `Eating Source` in order to show an example of inferences due to extensional relationships:

`New-Food-pattern =` (`New-Food`,{ `name`,`specialty`,`category`*})

The related ODL$_{I^3}$ class is :

```
interface ED.New-Food
 ( source semistructured Eating_Source )
{  attribute string      name;
   attribute set<string> specialty;
```

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

```
    attribute string      category*;  };
```

Moreover, suppose that the designer states the following extensional relationship:

$\langle$`ED.New-Food` BT_{ext} `FD.Restaurant`$\rangle$.

This extensional relationship is validated as consistent; in fact the `category` attribute is optional in `ED.New-Food`. By exploiting subsumption computation ODB-Tools obtains the following inferred relationship: $\langle$`ED.New-Food` BT_{ext} `FD.Bistro` $\rangle$.

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. The proposed process allows semi-automatic identification of ODL_{I^3} classes candidate to integration by means of clustering procedures based on the concept of affinity and on the relationship knowledge in the Common Thesaurus. Moreover, the process supports a semi-automated synthesis of each selected cluster into an integrated global ODL_{I^3} class, by handling semantic heterogeneity through the definition of suitable mapping rules for each global class.

4.1 Affinity 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 and give a measure of the level of matching of their structure.

This activity is performed with the ARTEMIS tool environment. ARTEMIS has been conceived for a semi-automatic integration of heterogeneous structured databases [16]. In the context of MOMIS, the ARTEMIS affinity framework has been extended and applied to the analysis of ODL_{I^3} schema descriptions. In the following, we describe the extensions to the affinity-based clustering to cope with object patterns 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.

Table 1
Name Affinity coefficient

Coefficient	**Value**	**Condition**
$NA(c, c')$	$A(n_c, n_{c'})$	if $A(n_c, n_{c'}) \geq \alpha$
	0	if $A(n_c, n_{c'}) < \alpha$

Legend:

$n_c, n_{c'}$ denote the name of c and c', respectively.

α is a threshold used to select high values of $NA(c, c')$.

Affinity coefficients for ODL$_{I^3}$ classes are evaluated by exploiting terminological relationships of 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 $\sigma_{\text{RT}} = 0.5$. An affinity function $A()$ is defined on top of the Common Thesaurus to evaluate the affinity of two terms. The affinity $A(t, t')$ of two terms t and t' is equal to the highest-strength path of terminological relationships between them, if at least one path exist, and is zero otherwise. Given two terms t and t' and a path of terminological relationships between them, the strength of this path is computed by multiplying the strengths of all terminological relationships involved in it. $A(t, t')$ coincides with the strength of the highest-strength path between t and t', denoted by $\rightarrow^m$, that is, $A(t, t') = \sigma_{12_{\Re}} \cdot \sigma_{23_{\Re}} \cdot \ldots \cdot \sigma_{(m-1)m}$. In the following, we use the symbol $\sim$ to denote the fact that two terms have affinity in the Common Thesaurus.

Let c and c' be two ODL$_{I^3}$ 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 ODL$_{I^3}$ classes c and c', denoted *NA*(c, c'), is the measure of the affinity between their names n_c and $n_{c'}$, if this measure exceeds a specified threshold (see Table 4.1.1).

For any pairs of classes, $NA(c, c') \in [0, 1]$. $NA(c, c')$ is equal to the strength of the path $\rightarrow^m$ of terminological relationships in the Common Thesaurus originating the highest strength value, if this value exceeds a specified threshold α (e.g., $\alpha \in [0.4, 0.6]$). In this case, the Name Affinity value is proportional to the length of the path and to the type of relationships involved in this path. In particular, $NA(c, c')$ is 1 if only SYN relationships are involved in a path. Otherwise, $NA(c', c')$ is zero.

Example 6 Consider the Common Thesaurus illustrated in figure 5.

$NA(\texttt{FD.Person}, \texttt{FD.Owner}) = 0.8$, since it corresponds to the highest- strength path $\texttt{FD.Person} \xrightarrow{\text{BT}} \texttt{FD.Owner}$ in the Common Thesaurus.
As another example, the $NA(\texttt{FD.Restaurant}, \texttt{FD.Fast} - \texttt{Food}) = 1$, due to the path $\texttt{FD.Restaurant} \xrightarrow{\text{SYN}} \texttt{FD.Fast} - \texttt{Food}$, which is the one with the highest strength in the Common Thesaurus.

4.1.2 Structural Affinity coefficient

The Structural Affinity coefficient of two ODL$_{I^3}$ classes c and c', denoted $SA(c, c')$, is the measure of the level of matching of c and c' based on attribute relationships in the Common Thesaurus (see Table 4.1.2).

Table 2
Structural Affinity coefficient

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

$F_c = \frac{\lvert\{x \in C \mid flag(x) = 1\}\rvert}{\lvert C\rvert}$, control factor where $flag(x) = 1$ stands for a valid terminological relationship in the Common Thesaurus

The Structural Affinity coefficient returns a value in the range $[0, 1]$ proportional to the number of attributes of the two classes whose names have affinity in the Common Thesaurus, refined by a control factor F_c. In particular, F_c evaluates the percentage of attributes having affinity that have a valid relationship in the Common Thesaurus (see step Validation of relationships illustrated in Section 3.4).
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 and are considered valid. The greater the number of attributes with affinity in the considered classes, and the greater the number of positive validity control results, the higher the value of $SA(c, c')$.
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 $SA(c, c')$ coefficient, we consider these multiple affinities as a single affinity correspondence 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 7 Consider classes `ED.Owner` and `FD.Person`. By applying the all attribute-based option, we have that

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

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 ODL$_{I^3}$ 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).

Table 3
Global Affinity coefficient

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.

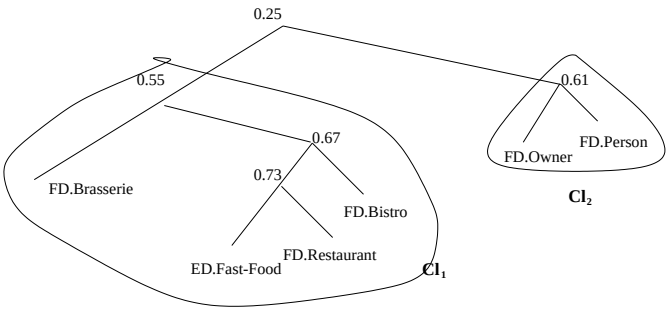

Fig. 6. Example of affinity tree

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 8 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 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 [23].

The hierarchical clustering procedure uses a matrix M of rank K where K is 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 itself. Then, at each iteration, the two clusters having the greatest affinity 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 affinity value between c_{hk} and each remaining cluster $\bar{c}$ in M is computed. The new value between c_{hk} and a remaining cluster $\bar{c}$ is set to the maximum affinity value between the affinity values that c_h and c_k had with $\bar{c}$ in M. The procedure terminates when only one cluster is left and produces as the output a tree, where leaves correspond to ODL$_{I^3}$ classes and intermediate nodes have an associated affinity value characterizing the underlying leaves.

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

Once the affinity tree has been constructed, an important issue is related to the selection of clusters to be integrated for the definition of the global ODL$_{I^3}$ classes of the integrated 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 an affinity value greater than or equal to T are selected and proposed to the designer. High values of T return small, highly homogeneous clusters. By decreasing T's value, clusters containing more ODL$_{I^3}$ 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 characteristics of retrieved clusters and of the specific application under analysis. Once clusters have been selected, ODL$_{I^3}$ classes that have an extensional terminological relationship with at least one class in the cluster and not yet included in it (if any), are forced to belong to the cluster anyway, to define an integrated global ODL$_{I^3}$ class that is representative of all possible semantically related source classes.

In our example, using a threshold $T = 0.5$, two clusters are automatically selected, namely Cl_1 and Cl_2, as shown in Figure 6. Cluster Cl_1 contains all ODL$_{I^3}$ classes describing different kinds of eating place, while cluster Cl_2 contains all ODL$_{I^3}$ classes describing persons. These two cluster are highly homogeneous and contain all involved classes also with respect to extensional relationships. The designer can confirm this threshold-based cluster selection made by the tool for the subsequent synthesis activity.

4.2 Synthesis into an integrated schema description

The generation of global ODL$_{I^3}$ classes out of selected clusters is a synthesis activity performed interactively with the designer. 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. 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 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 `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 of the attributes of a class $c_h \in$ `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 to have a cluster describing an `automobile` global 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 applica-

```
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';
}
```

Fig. 7. Example of global class specification in ODL$_{I^3}$

ble 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` regard 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.

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 :
```

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'

Fig. 8. `Food_Place` mapping table

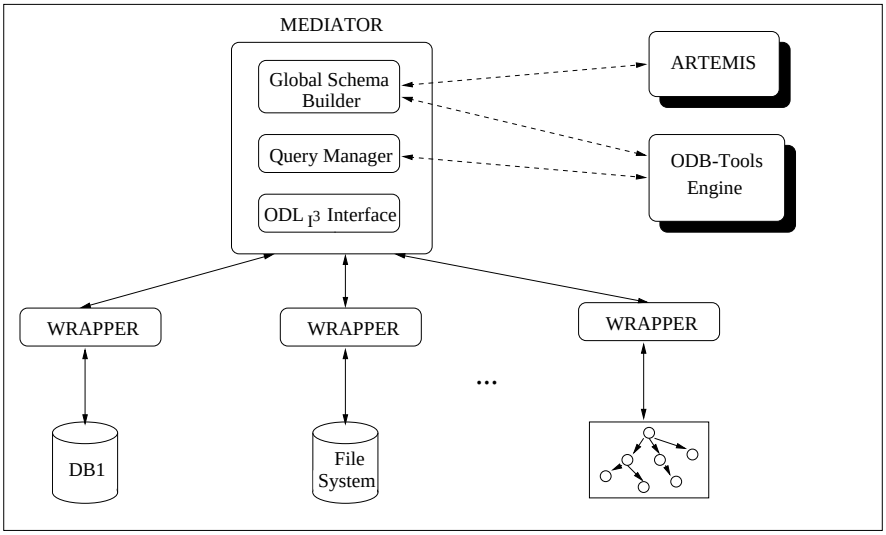

Fig. 9. Architecture of the MOMIS system

```
(X.category ='High') then X.price > 100;
```

5 The MOMIS system

In this section we describe the architecture of the MOMIS system embedding the proposed extraction and integration techniques and give some considerations on its experimentation and design choices.

5.1 Architecture of the MOMIS system

The MOMIS system has been conceived as a pool of tools, 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 data sources. MOMIS is based on the I^3 architecture [2] (see figure 9). At the bottom layer we have the schemas of information sources, while the layers above provide the semantic integration and the coordination management support. 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 *ARTEMIS Tool Environment*, a tool based on affinity and clustering [17,16], which performs classification of ODL_{I^3} classes for the synthesis of global ODL_{I^3} classes.
- The *ODB-Tools Engine*, a tool based on the OLCD Description Logics [5,8], which performs schema validation for the generation of the Common Thesaurus and query optimization.

5.2 Experimental considerations

Some activities performed in MOMIS for information source integration have an element of subjectivity, as it is intuitive, due to the fact that the knowledge and experience of the designer can be required to be sure to proceed correctly. In particular, the specification of inter-source terminological relationships, both intensional and extensional, and the affinity evaluation activities are of this kind. Our effort has been the one of making these activities even more objective, by providing interactive functionalities as well as pre-defined tool choices by means of default parameter values, that can be however interactively modified by designer if necessary. In the following, we report main feedbacks of the experimentation of MOMIS tools on practical integration examples.

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. Regarding correctness of terminological relationships, we adopt an interactive construction and validation of the Common Thesaurus with ODB-Tools. In this way, the designer can interact with WordNet to select the most appropriate terminological relationships, to be forced also at the extensional level if necessary. Moreover, he can supply additional terminological relationships typical of application domain

under consideration, if the ones retrieved from WordNet are not sufficient to cover all inter-source relationships that hold among analyzed information sources. Regarding affinity parameters, the affinity evaluation activity is also interactive and weight-based, using the ARTEMIS tool environment. This enables the designer to properly set the involved parameters and to validate the choices performed by the tool in all steps of the evaluation process (for a detailed description of these aspects the reader can refer to [17]). ARTEMIS 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 the various parameters (strengths, thresholds, weights) intervening in the affinity and clustering stages. The values of terminological relationship strengths in the Common 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 integration contexts. As a consequence, most difficult tasks involved in the integration process result simplified in that semantically related information is automatically identified and the designer is asked for a validation of the proposed results or, for ambiguous situations, for a selection among a set of pre-defined choices. Moreover, the possibility of interacting with the tool to vary default parameter configurations and compare their results, allows the tuning of the integration process. The experimented interaction with WordNet was satisfactory in sources integration when the schemata to be integrated have "meaningful names"; in this case, most of the terminological inter-schema relationships are obtained, preventing a lot of boring work for the designer. On the other hand, for many legacy applications adopted names are not meaningful. In this case, the automatic extraction of intra-schema relationships and the aid of the system in checking consistency of explicitly given relationships are good aids for the designer.
A deeper discussion of the experimentation results of the use of strengthened terminological relationships and affinity-based clustering of ARTEMIS for the integration of heterogeneous data schemas in the Italian Public Administration domain, is reported in [16,18].

6 Related work

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

Heterogeneous information integration.
In this area, many projects based on a mediator architecture have been developed. For example, the mediator-based TSIMMIS project [21] follows a 'structural' approach and uses a self-describing model (OEM) to represent heterogeneous data sources 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 must first be wrapped and the mediator rules have to be redefined to take into account new knowledge and their MSL definitions recompiled. The administrator of the system must figure out whether and how the new sources have to be assimilated in the mediator. In our case, this information is automatically discovered by means of clustering. MOMIS is based on a mediator architecture and follows the 'semantic approach'. Following the classification of integration systems proposed by Hull [28], MOMIS is in the line of the "virtual approach" and is in the category of "read-only views", that is, it is a system whose task is to support an integrated, read-only, view of data stored in multiple sources. The most similar projects are: GARLIC, SIMS [1], Information Manifold [30] and Infomaster [26].

The GARLIC project [15] 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 [1] proposes to create a global schema definition by 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 system [30], as the MOMIS project, provides a source independent and query independent mediator. The input schema of Information Manifold is a set of descriptions of the sources. Given a query, the system will create a plan for answering the query using the underlying source descriptions. Algorithms to decide the useful information sources and to generate the query plan have been implemented. The integrated schema is defined mainly manually by the designer, while in our approach it is tool-supported.
Infomaster [26] 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 to harmonize possible heterogeneities across the sources.

An approach based on Description Logics and ontologies is taken in the OBSERVER system to support semantic interoperation and formulation of rich queries over distributed information repositories where different vocabularies are used [31]. Here the idea is that each repository has its own ontology. Inter-ontology relationships are specified in a declarative way (using Description Logics) in an inter-ontology manager module to handle vocabulary heterogeneities between ontologies of different information repositories for query processing. In this respect, our Common

Thesaurus plays a similar role in that we specify inter-source terminological relationships. The focus here is more on representation of inter-ontology relationships to solve vocabulary problems at the intensional and extensional level for query processing rather than on using these relationships for deriving an integrated virtual view of the underlying information sources. Moreover, in our approach, we try to extract as much information as possible from source descriptions and from WordNet and we show how this information can be used for affinity evaluation and integration purposes.

The analysis, discovery, and representation of inter-schema properties is another critical aspect of the integration process and research proposals have appeared on this topic. In [34], semi-automatic techniques for discovering synonyms, homonyms and object inclusion relationships from database schemas are described and a semi-automatic algorithm for integrating and abstracting database schemes is presented in [35]. 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 [29], concentrating mainly on high level tasks (e.g., o-operation, planning, belief revision) related to the extraction process [22].

Semistructured data. The issue of modeling semistructured data has been investigated in the recent literature. In particular, a survey of problems concerning semistructured data modeling and querying is presented in [10]. Two similar models for semistructured data have been proposed [12,36], based on rooted, labeled graph with the objects as nodes and labels on edges. According to the model presented in [12], information resides at labels only, while according to the "Object Exchange Model" (OEM) proposed by Papakonstantinou et. al. in [36], information also resides at nodes. Very similar proposal for modelling semi-structured data come from the Artificial Intelligence area [14], where in analogy with our approach, a Description Logic 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 [27], 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 [33], where the authors use a graph-based data model derived from [12,36]. In [10], 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.

More recently, XML [9] has emerged in the framework of information represen-

tation over the Web allowing the designer to explicitly point out semantic role of data within a source. Here, the notion of Document Type Definition (DTD) is introduced to model the structure of a set of documents. DTDs play a role similar to our object patterns and can be directly used to derive the ODL_{I^3} description of the information associated with them.

Original contributions of our work. The original contribution of the work presented in this paper is related to the availability of a set of techniques for the designer to face common problems that arise when integrating pre-existing information sources, containing both semistructured and structured data. The idea of combining reasoning capabilities of Description Logics with affinity-based clustering techniques is new and allows both the validation of the inter-source knowledge used for the integration and the identification of candidates to integration in a way automated as much as possible. The interactive exploitation of WordNet from within our tools combined with subsequent affinity analysis is also a novel capability for an integration tool. In this way, we can take into account and interactively revise semantic knowledge related to the meaning of names in the considered source description, as well as the structure of classes in source schema descriptions and their level of matching to come up with as much information as possible for the extraction of global integrated classes. Furthermore, we provide the capability of explicitly introducing many kinds of knowledge as: integrity constraints, extensional relationships and to check the global consistency of implicit and designer provided knowledge.

7 Concluding remarks

In this paper, we have presented a semi-automated approach to information extraction and integration of heterogeneous information sources. The ODL_{I^3} Description Logic-based language is introduced for information extraction and integration, by taking into account also semistructured information sources. A Description Logic module (ODB-Tools engine) provides inference capabilities to construct a Common Thesaurus of inter-source terminological relationships and a cluster generator module (ARTEMIS tool environment) provides capabilities to identify candidates to integration for the abstraction of global ODL_{I^3} classes. The proposed approach has been implemented in the MOMIS system following a conventional wrapper/mediator architecture. MOMIS provides a set of tools and associated techniques for performing semantic intgration. MOMIS provides several techniques and associated tools for helping the designer in the integration of heterogeneous information sources, and not all such techniques may be necessary in all integration processes. To cope with this point, an application with a graphical interface, called SI-Designer, has been developed (to be presented at the demo session of the VLDB2000 [4]). SI-Designer interfaces all the employed tools with the goal of al-

lowing an interactive and customized use of MOMIS techniques by the designer, based on the specific requirements of a given integration process.

Future research will be devoted to the development of the Query Manager component of MOMIS with query optimization and "answer composition" functionalities, based on definition of extensional axioms and integrity constraints defined on global ODL_{I^3} classes. 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. One of the original aspects of the Query Manager will consists in employing Description Logics based components (i.e., ODB-Tools) to perform semantic optimization steps on both on global and local queries, to minimize the number of accessed sources and the volume of data to be integrated as the result of sub-query execution. Research will proceed also in the direction of designing and developing tools to help the designer in extensional axiom specification and to reason about specified axioms, to increase the knowledge available for the integration process. The methodology that is meant to be used in order to exploit extensional knowledge consists in the identification of the "base extensions", as recently proposed by [38] 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. Finally, the approach will be extended to take into account XML [9] data sources.

Acknowledgements

Authors wish to thank anonymous referees for their insightful comments that led us to an improved presentation of the paper. Silvana Castano wish to thank Valeria De Antonellis for her collaboration in developing affinity and clustering techniques in ARTEMIS.

References

[1] Y. Arens, C. A. Knoblock and C. Hsu, Query Processing in the SIMS Information Mediator, *Advanced Planning Technology* (AAAI Press, Menlo Park, CA, 1996).

[2] ARPA, ARPA I^3 Reference Architecture,
(Available at `http://www.isse.gmu.edu/I3_Arch/index.html`)

[3] M. Bates, Subject access in online catalogs: A design model, *Journal of the American Society for Information Science* **11** (1986) 357–376.

[4] D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori and M. Vincini, Information Integration: the MOMIS Project Demonstration, *Proc. Int. Conf. on Very Large Data Bases VLDB-2000* (Cairo, Egypt, 2000). (to appear)

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

[6] D. Beneventano, S. Bergamaschi, C. Sartori and M. Vincini, ODB-Tools: A Description Logics Based Tool for Schema Validation and Semantic Query Optimization in Object Oriented Databases, *Proc. Int. Conf. on Data Engineering ICDE-97* (Birmingham, UK, 1997) 578.

[7] S. Bergamaschi, S. Castano and M. Vincini, Semantic Integration of Semistructured and Structured Data Sources, *SIGMOD Record Special Issue on Semantic Interoperability in Global Information* **28(1)** (1999) 54–59.

[8] 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** (1994) 185–203.

[9] B. Bos, The XML Data Model (1997)
(Available at `http://www.w3.org/XML/Datamodel.html`).

[10] P. Buneman, Semistructured Data, *Proc. Symposium on Principles of Database Systems PODS-97,* (Tucson, Arizona, 1997) 117–121.

[11] P. Buneman, S. B. Davidson, M. F. Fernandez and D. Suciu, Adding Structure to Unstructured Data, *Proc. Int. Conf. on Database Theory ICDT-97* (Delphi, Greece, 1997 - Lecture Notes in Computer Science, Vol. 1186 Springer) 336–350.

[12] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu, A Query Language and Optimization Techniques for Unstructured Data, *Proc. Int. Conf. ACM SIGMOD-96,* (Montreal, Canada, 1996) 505–516.

[13] L. Cardelli, A semantics of multiple inheritance, *Semantics of Data Types,* (Lecture Notes in Computer Science Vol. 173, Springer-Verlag, 1984) 51–67.

[14] D. Calvanese, G. De Giacomo and M. Lenzerini, What can knowledge representation do for semi-structured data?, *Proc. National Conf. on Artificial Intelligence AAAI-98* (1998) 205–210.

[15] 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.

[16] S. Castano, V. De Antonellis, S. De Capitani Di Vimercati, Global Viewing of Heterogeneous Data Sources, *IEEE Transactions on Knowledge and Data Engineering,* (2000).

[17] S. Castano and V. De Antonellis, A Schema Analysis and Reconciliation Tool Environment for Heterogeneous Databases, *IEEE Proc. of IDEAS'99 International Database Engineering and Applications Symposium* (Montreal 1999) 53–62. (http://bsing.ing.unibs.it/ deantone/interdata_tema3/forms/t3-s13-h.htm)

[18] S. Castano and V. De Antonellis, A Discovery-Based Approach to Database Ontology Design, *Distributed and Parallel Databases - Special Issue on Ontologies and Databases* **7(1)** (1999) 67–98.

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

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

[21] S. Chawathe, H. Garcia Molina, J. Hammer, K. Ireland, Y. Papakostantinou, J.Ullman, and J.Widom, The TSIMMIS project: Integration of Heterogeneous Information Sources, *Proceedings Meeting of the Information Processing Society of Japan IPSJ-94* (Tokyo, Japan, 1994) 7–18.

[22] A. F. Dragoni, Belief Revision: from Theory to Practice, *The Knowledge Engineering Review* **12(2)** (1997) 147–179.

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

[24] C. Fahrner and G. Vossen, Transforming Relational Database Schemas into Object Oriented Schemas according to ODMG-93, *Proc. Int. Conf. Deductive and Object-Oriented Databases DOOD-95*, (Lecture Notes in Computer Science, Vol. 1013, Springer, 1995) 429-446.

[25] 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, *Journal of Intelligent Information Systems* **8(2)** (1997) 117–132.

[26] M. R. Genesereth, A. M. Keller and O. Duschka, Infomaster: An Information Integration System, *Proc. Int. Conf. ACM SIGMOD-97* (Tucson, Arizona, 1997) 539–542.

[27] R. Goldman and J. Widom, DataGuides: Enabling Query Formulation and Optimization in Semistructured Data, *Proc. Int. Conf. on Very Large Data Bases VLDB-97*, (Athens, Greece, 1997) 436–445.

[28] R. Hull, Managing Semantic Heterogeneity in Databases: A Theoretical Perspective. *ACM Symp. on Principles of Database Systems*, (Tucson, Arizona, 1997) 51–61.

[29] V. Lesser, B. Horling, F. Klassner, A. Raja, T. Wagner and S. XQ. Zhang, BIG: A Resource-Bounded Information Gathering Agent, *Proc. AAAI-98* (Madison, Wisconsin 1998) 539–546.

[30] A. Y. Levy, A. Rajaraman, and J. J. Ordihe, Querying heterogeneous information sources using source descriptions, *Proc. Int. Conf. on Very Large Data Bases VLDB-96*, (Mumbai, India, 1996) 251–262.

[31] E. Mena, V. Kashyap, A. Sheth and A. Illarramendi, OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies, *Proc. Int. Conf. on Cooperative Information Systems CoopIS-96* (Brussels, Belgium, 1996) 14–25.

[32] A.G. Miller, WordNet: A Lexical Database for English, *Communications of the ACM* **38(11)** (1995) 39–41.

[33] S. Nestorov, S. Abiteboul and R. Motwani, Extracting Schema from Semistructured Data, *Proc. Int. Conf. ACM SIGMOD-98* (Seattle, Washington, 1998) 295–306.

[34] L. Palopoli, D. Saccà and D. Ursino, Automatic Derivation of Terminological Properties from Database Schemes, *Proc. DEXA'98* (Wien, Austria, 1998) 90–99.

[35] L. Palopoli, D. Saccà and D. Ursino. Semi-Automatic Semantic Discovery of Properties from Database Schemes, *Proc. IDEAS'98* (Cardiff, UK, 1998) 244–253.

[36] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom, Object Exchange Across Heterogeneous Information Sources, *Proc. of Int. Conf. on Data Engineering, ICDE'95* (Taipei, Taiwan, 1995) 251–260.

[37] R. Reiter, What should a database know?, *Proceedings of the Seventh Symposium on Principles of Database Systems* (Austin, Texas, 1988) 302-304.

[38] I. Schmitt and G. Saake, Merging Inheritance Hierarchies for Database Integration, *Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems* (New York, 1998) 322–331.

[39] W.A. Woods and J.G. Schmolze, The kl-one family, *Special Issue of Computers & Mathematics with Applications* **23** (1989)

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

The following is a BNF description for the ODL$_{I^3}$ description language. We included the main syntax fragments which differ from the original ODL grammar (see `http://sparc20.dsi.unimo.it/Momis/documents/odli3_syntax.pdf` for the complete syntax)

⟨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⟩ ::= **file** | **relational** | **nfrelational**
| **object** | **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⟩

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⟩ : : = ⟨intensional_relationship⟩ | ⟨extensional_relationship⟩
⟨intensional_relationship⟩ : : = **SYN** | **BT** | **NT** | **RT**
⟨extensional_relationship⟩ : : = **SYN_E** | **BT_E** | **NT_E**

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                    interface Address
( source semistructured                ( source semistructured
   Eating_Source )                        Eating_Source )
{ attribute string      name;          { attribute string city;
  attribute Address     address;         attribute string street;
  attribute integer     phone*;          attribute string zipcode;};
  attribute set<string> specialty;         union
  attribute string      category;      { string;};
  attribute Restaurant  nearby*;
  attribute integer     midprice*;
  attribute Owner       owner*;};

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

Food_Guide_Source (FD):
interface Restaurant                     interface Person
( source relational Food_Guide           ( source relational Food_Guide
   key r_code                               key pers_id)
   foreign_key(pers_id)                  { attribute integer pers_id;
        references Person )                 attribute string  first_name;
{ attribute string      r_code;            attribute string  last_name;
  attribute string      name;              attribute integer qualification;
  attribute string      street;
  attribute string      zip_code;
  attribute integer     pers_id;
  attribute string      special_dish;
  attribute integer     category;
  attribute integer     tourist_menu_price;};

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

C OLCD : Interpretations and Database Instances

We assume the union of the integers, the strings, the booleans, and the reals as the set $\mathcal{D}$ of *base values*. To build *complex values*, we further assume a countable, set disjoint from $\mathcal{D}$, of *object identifiers* (denoted by $o, o_1, o_2, \ldots$). The set $\mathcal{V}$ of all *values over* O is defined as the smallest set containing $\mathcal{D}$ and O, such that, if $v_1, \ldots, v_p$ are values, then the set $\{v_1, \ldots, v_p\}$ is a value, and a partial function $t\colon \mathbf{A} \to \{v_1, \ldots, v_p\}$ is a value. The function t is the usual tuple value; the standard notation $[a_1\colon v_1, \ldots, a_p\colon v_p]$ will be henceforth used.

Let $=, \neq, >, <, \geq, \leq$ be the equality, inequality and total order relations, denoted by θ, defined as usual on $\mathcal{D}$. Equality and inequality can be extended from $\mathcal{D}$ to all $\mathcal{V}$: the equality operator ($=$) has the meaning of *identity*, i.e., two objects are equal if they have the same identifier, two sets are equal iff they have equal elements, two tuples, say $v_a = [a_1\colon v_1, \ldots, a_p\colon v_p]$ and $v_b = [a'_1\colon v'_1, \ldots, a'_q\colon v'_q]$, are equal if they have the same attributes and equal attribute labels are mapped to equal values. Object identifiers are assigned values by a *total value function* δ from O to $\mathcal{V}$.

Let $\mathbf{W}$ denote the set of all paths. Given a set of object identifiers O and a value function δ, let $\mathcal{J}\colon \mathbf{W} \longrightarrow 2^{\mathcal{V}\times\mathcal{V}}$ a function defined as follows:

- empty path: $\mathcal{J}[\epsilon] = \Big\{(v, v) \in \mathcal{V} \times \mathcal{V}\Big\}$
- single element path: $\mathcal{J}[a] = \Big\{(v_1, v_2) \in \mathcal{V} \times \mathcal{V} \mid v_1 = [\ldots, a : v_2, \ldots]\Big\}$
  $$\mathcal{J}[\triangle] = \Big\{(o, v) \in O \times \mathcal{V} \mid \delta(o) = v\Big\}$$
- multiple element path: $\mathcal{J}[e_1 \,.\, e_2 \,.\, \cdots \,.\, e_n] = \mathcal{J}[e_1] \circ \mathcal{J}[e_2] \circ \cdots \circ \mathcal{J}[e_n]$
  where $\circ$ is the symbol of function composition.

Notice that, for all p, $\mathcal{J}[p]$ is undefined on set values. Let v be a value and p be a path. By $\mathcal{J}[p](v)$ we mean the unique value (when it exists) reachable from v following p, that is the value of the partial function $\mathcal{J}[p]$ in v.

Let $\mathcal{I}_{\mathbf{B}}$ be the (fixed) standard interpretation function from $\mathbf{B}$ to $2^{\mathcal{D}}$. For a given object assignment δ, each type expression S is mapped to a set of values (its interpretation). An *interpretation function* is a function $\mathcal{I}$ from $\mathbf{S}$ to $2^{\mathcal{V}}$ satisfying the following equations:

$$\begin{aligned}\mathcal{I}[\top] &= \mathcal{V}\\ \mathcal{I}[\bot] &= \emptyset\\ \mathcal{I}[B] &= \mathcal{I}_{\mathbf{B}}[B]\end{aligned}$$

$$
\begin{aligned}
\mathcal{I}[\{S\}_\forall] &= \{M \mid M \subseteq \mathcal{I}[S]\} \\
\mathcal{I}[\{S\}_\exists] &= \{M \mid M \cap \mathcal{I}[S] \neq \emptyset\} \\
\mathcal{I}\Big[[a_1 : S_1, \ldots, a_p : S_p]\Big] &= \{t \colon \mathbf{A} \to \mathcal{V} \mid t(a_i) \in \mathcal{I}[S_i], 1 \leq i \leq p\} \\
\mathcal{I}[S_1 \sqcap S_2] &= \mathcal{I}[S_1] \cap \mathcal{I}[S_2] \\
\mathcal{I}[S_1 \sqcup S_2] &= \mathcal{I}[S_1] \cup \mathcal{I}[S_2] \\
\mathcal{I}[\neg S] &= \mathcal{V} \setminus \mathcal{I}[S] \\
\mathcal{I}[\triangle S] &= \left\{ o \in O \middle| \delta(o) \in \mathcal{I}[S] \right\} \\
\mathcal{I}[(p \theta d)] &= \left\{ v \in \mathcal{V} \mid \mathcal{J}[p](v) \theta d \right\} \\
\mathcal{I}[(p\!\uparrow)] &= \{v \in \mathcal{V} \mid v \notin \text{ dom } \mathcal{J}[p]\}
\end{aligned}
$$

Note that the interpretation of tuples implies an open world semantics for tuple types similar to the one adopted by Cardelli [13], and that $(p\uparrow)$ selects objects which do not have the path p. It should be noted than an interpretation does not necessarily imply that the extension of a named type is identical to the type description associated with the type name via the schema σ. For this purpose, we have to further constrain the interpretation function: An interpretation function $\mathcal{I}$ is a *legal instance* of a schema σ iff the set O is finite, and for all $N \in \mathbf{N}$:

$$
\begin{aligned}
\mathcal{I}[N] &\subseteq \mathcal{I}[\sigma_P(N)] \qquad \text{if } N \in \text{dom } \sigma_P \\
\mathcal{I}[N] &= \mathcal{I}[\sigma_V(N)] \qquad \text{if } N \in \text{dom } \sigma_V
\end{aligned}
$$

From the above definition, we see that the interpretation of a primitive type name *is included* in the interpretation of its description, while the interpretation of a virtual type *is* the interpretation of its description. In other words, the interpretation of a primitive type name has to be provided by the user, according to the given description, while the interpretation of a virtual type name is drawn from its definition and from the interpretation of primitive type names, thus corresponding to a view in database context.

Given a type S of a schema σ, we say that S is *consistent* if and only if there is a legal instance $\mathcal{I}$ of σ such that $\mathcal{I}[S] \neq \emptyset$. Given two types S_1, S_2 of a schema σ, we say that S_1 *subsumes* S_2 iff $\mathcal{I}[S_1] \supseteq \mathcal{I}[S_2]$ for all legal instances $\mathcal{I}$ of σ. Consistency and subsumption can be reduced to each other, according to the following rules: S_1 is subsumed by S_2 iff $S_1 \sqcap \neg S_2$ is inconsistent, and S is consistent iff it is not subsumed by $\perp$. The consistency problem is PSPACE-hard; in [5], an algorithm for checking the consistency of a type (which can also be used for subsumption computation), based on the *tableaux calculus*, is given.