Chapter 14
Using OWL in Data Integration

Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo,
Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi

Abstract One of the outcomes of the research work carried out on data integration in the last years is a clear architecture, comprising a global schema, the source schema, and the mapping between the source and the global schema. In this chapter, we study data integration under this framework when the global schema is specified in OWL, the standard language for the Semantic Web, and discuss the impact of this choice on computational complexity of query answering under different instantiations of the framework in terms of query language and form and interpretation of the mapping. We show that query answering in the resulting setting is computationally too complex, and discuss in detail the various sources of complexity. Then, we show how to limit the expressive power of the various components of the framework in order to have efficient query answering, in principle as efficient as query processing in relational DBMSs. In particular, we adopt OWL 2 QL as the ontology language used to express the global schema. OWL 2 QL is one of the tractable profiles of OWL 2, and essentially corresponds to a member of the *DL-Lite* family, a family of Description Logics designed to have a good trade-off between expressive power of the language and computational complexity of reasoning.

14.1 Introduction

Data integration is the problem of combining data from different sources, and to provide a single interface for the consumers of information. The main purpose of such an interface is to free the client from the knowledge about where the source

Diego Calvanese
KRDB Research Centre, Free University of Bozen-Bolzano, Piazza Domenicani 3, I-39100 Bolzano, Italy, e-mail: `calvanese@inf.unibz.it`

Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati and Marco Ruzzi
Dip. di Informatica e Sistemistica, SAPIENZA Università di Roma, Via Ariosto 25, I-00185 Roma, Italy, e-mail: `<lastname>@dis.uniroma1.it`

data are, and how they can be accessed. Data Integration is considered one of the main challenges that Information Technology (IT) currently faces [10]. It is highly relevant in classical IT applications, such as enterprise information management and data warehousing, as well as in scenarios like scientific computing, e-government, and web data management.

The need of integrating data arises not only within a single organization, but also when different organizations interoperate (inter-organization integration), like in supply-chain scenarios, or in industrial districts. One of the reasons for such a need is that, on the one hand, large amounts of heterogeneous data, often collected and stored by different applications and systems, are nowadays available, and on the other hand, the need of accessing such data by means of unified mechanisms is becoming more and more crucial, both within and outside the organization boundaries.

In the last two decades, the research on data integration has produced many significant results. One important outcome of this research work is a clear architecture for data integration [70, 63]. According to this architecture, the main components of a data integration system are the global schema, the sources, and the mapping between the two. The sources represent the repositories where the data are, the global schema, also called mediated schema, represents the unified structure presented to the client, and the mapping relates the source data with the global schema. A typical service provided by the system is query answering, i.e., computing the answer to a query posed to the global schema by accessing the sources, collecting the relevant data, and packaging such data in the final answer. Much of the work carried out by the scientific community has concentrated on this task. Many surveys indicate that the market for information integration software is expected to grow considerably in the next years, and to reach about 4 billion USA dollars in 2012. Despite the above-mentioned research work, and despite the urgent need for effective solutions, information integration is still largely unresolved. Each commercial data integration tool covers only one of the aspects of the problem, e.g., extraction and loading of data, wrapping data sources, or answering federated queries. The result is that a comprehensive solution to the problem is very difficult to achieve with the current technology. There are several reasons why the current methods and techniques for information integration are still far from providing such a solution. We argue that one of the most important reasons is that current tools express the global schema in terms of a so-called logical database model, typically the relational data model. The term logical model refers to the data models of current Data Base Management Systems (DBMSs), and it is well-known that the abstractions and the constructs provided by these models are influenced by implementation issues. It follows that the global schema represents a sort of unified data structure accommodating the various data at the sources, and the client, although freed from physical aspects of the source data (where they are, and how they can be accessed), is still exposed to logical aspects of such data, i.e., how they are packed into specific data structures.

In this chapter we discuss a different approach, which is based on the idea of posing the semantics of the application domain at the center of the scene. According to this approach, the usual global schema of traditional data integration systems is replaced by the conceptual model of the application domain, and such a conceptual schema is expressed through OWL, the logic-based ontology language used in the Semantic Web. With this approach, the integrated view that the system provides to information consumers is not merely a data structure accommodating the various data at the sources, but a semantically rich description of the relevant concepts in the domain of interest. The distinction between the conceptual model and the data sources reflects the separation between the conceptual level, the one presented to the client, and the logical/physical level of the information system, the one stored in the sources, with the mapping acting as the reconciling structure between the two levels. By using OWL, the global schema of the integration system becomes a declarative, formal specification of the domain of interest, i.e., a logic-based conceptualization of the relevant concepts in the domain, and the relevant relationships among these concepts.

There are several crucial advantages in the semantic approach we pursue in our proposal. First, the conceptual layer in the architecture is the obvious mean for pursuing a declarative approach to information integration. By making the representation of the domain explicit, we gain *re-usability* of the acquired knowledge, which is not achieved when the global schema is simply a unified description of the underlying data sources. This may also have consequences on the design of user interfaces, since conceptual models are close to the user perception of the domain. Second, the mapping layer explicitly specifies the relationships between the domain concepts on the one hand and the data sources on the other hand. Such a mapping is not only used for the operation of the information system, but also for *documentation purposes*. The importance of this aspect clearly emerges when looking at large organizations where the information about data is widespread into separate pieces of documentation that are often difficult to access and non-necessarily conforming to common standards. The conceptual model built for data integration can thus provide a common ground for the documentation of the organization data, with obvious advantages in the maintenance phase of the system. A third advantage has to do with the *extensibility* of the system. One criticism that is often raised to data integration is that it requires merging and integrating the source data in advance, and this merging process can be very costly. However, the conceptual approach we advocate does not impose to fully integrate the data sources at once. Rather, after building even a rough skeleton of the domain model, one can incrementally add new data sources or new elements therein, when they become available, or when needed, thus amortizing the cost of integration. Therefore, the overall design can be regarded as the incremental process of understanding and representing the domain, the available data sources, and the relationships between them.

The adoption of an expressive ontology language for specifying the global schema comes, however, at a price. Indeed, it not hard to see that query answering in the resulting framework for data integration is computationally too complex. Starting from this observation, the goals of this chapter are as follows:

- We provide the formal definition (syntax and semantics) of a data integration framework where the global schema is an ontology expressed in OWL (Section 14.2).

- We discuss in detail the various sources of complexity in such a data integration framework (Section 14.3).
- We show how to limit the expressive power of the various components of the framework in such a way that query answering becomes tractable (Section 14.4). In particular, in order to achieve efficient query answering, in principle as efficient as query processing in relational DBMSs, we adopt a language of the *DL-Lite* family [19] as the ontology language used to express the global schema. More precisely, we consider OWL 2 QL, one of the tractable profiles of OWL 2.

We end the chapter by discussing related in Section 14.5, and by presenting some final observations in Section 14.6.

14.2 The data integration framework

As pointed out in the introduction, one of the major outcomes of the research in information integration [85, 70] is a conceptual architecture for a data integration system formed by a *global schema*, which provides a representation of the domain of interest, a *source schema*, which describes the set of sources involved in the integration process, and a *mapping*, which establishes the semantic relationship between the global and the source schema. In such a scenario, the user accesses the global schema asking her queries, which will be automatically processed taking into account the knowledge specified by the global schema, the mapping, and the data at the sources, which are suitably accessed to retrieve the data that form the answer for the user's query.

Following the idea that the domain of interest should be represented through a conceptual model, we find it natural to specify the global schema as an ontology. The ontology language we consider is OWL, the Web Ontology Language, in its forthcoming version, called OWL 2[1], which is currently being standardized by the World Wide Web Consortium (W3C)[2]. More precisely we adopt OWL 2 DL, the fragment of OWL 2, that is based on Description Logics (DLs) [8]. DLs are logics specifically designed to represent structured knowledge and to reason upon it, and as such are perfectly suited as languages for representing ontologies. DLs are based on the idea that the knowledge in the domain to represent should be structured by grouping into classes objects of interest that have properties in common, and explicitly representing those properties through the relevant relationships holding among such classes. It follows that the global schema is given in terms of *classes*, each representing a set of objects, *object properties*, i.e., binary relations between classes, and *data properties*, i.e., binary relations between classes and data-types.

From atomic classes and properties, it is possible to construct complex classes and properties. Then, the domain is modeled through assertions involving atomic and complex classes and properties (such as ISA relations, cardinality restrictions,

[1] `http://www.w3.org/TR/owl2-overview/`

[2] `http://www.w3c.org/`

specifications of domain and range of properties, etc.). We do not provide here the detailed syntax and semantics of OWL 2 DL, and refer the reader to the W3C technical report on on the subject[3]. It is however worth noticing that OWL 2 DL is essentially a variant of $\mathit{SROIQ}(D)$, a DL that extends the basic DL ALC with transitive roles, regular hierarchies, inverse rules, functionalities of roles, data types, and qualified number restrictions [65]. Therefore OWL 2 DL has a rigorous logical underpinning and a formal semantics, which essentially corresponds to the semantics of $\mathit{SROIQ}(D)$.

It is important to point out that the global schema only represents intensional knowledge and therefore for its specification we only allow for the use of OWL intensional constructs. In other words, the only instances available for the global schema are the instances retrieved from the sources through the mappings and no additional extensional knowledge is specified for it. Using the DL terminology, we say that the global schema is a TBox, and that no ABox, is explicitly specified.

As for the source schema, we assume it is specified as a relational schema. Indeed, very many data sources are available that store data using relational DBMSs. Furthermore, several software companies provide a variety of wrapper based tools enabling a database-like access to non-relational data sources. Therefore, we will describe the sources as a set of relational tables managed by the same relational DBMS and refer to them as if they were stored locally rather than distributed. This is indeed a situation that is always possible to achieve, by making use of capabilities provided by commercial relational data federation tools, which wrap heterogeneous and distributed data sources and present them as if they were a single relational database. We also assume that no integrity constraints are imposed over data sources, that is, each data source is responsible for maintaining its data correct with respect to its own constraints.

The mapping establishes the relationship between the source schema and the global schema, thus specifies how data stored at the sources are linked to the instances of the classes and the properties in the global schema.

Two basic approaches for specifying the mapping have been proposed in the literature. In the first approach, called *global-as-view* (GAV), a query over the source schema is associated with an element of the global schema, so that its meaning is specified in terms of the data residing at the sources. Conversely, the second approach, called *local-as-view* (LAV), requires every relation of the source schema to be defined as a query over the global schema. More recently, a further approach has been considered, which allows for specifying mapping assertions in which a query over the global schema is put in correspondence with a query over the source schema [70]. Such an approach, called (generalized) *global-local-as-view* (GLAV), since it generalizes both the LAV and the GAV approaches [70], is the one we adopt in this paper.

In the data integration settings described above, the mapping specification has to take into account the *impedance mismatch problem*, i.e., the mismatch between the way in which data is (and can be) represented in a data source, and the way in which

[3] `www.w3.org/TR/owl2-syntax/`

the corresponding information is rendered through the global schema. Specifically, the impedance mismatch problem arises from the fact that instances of the source relations are tuples of values, whereas instances of the classes and properties in the global schema are objects, each one denoted by an ad hoc identifier (e.g., a constant in logic), not to be confused with any value item.

More in detail, mapping assertions keep data values separate from object identifiers, and construct identifiers as (logic) terms over values. Each such object identifier has the form $f(d_1,\dots,d_n)$, where f is a function symbol of arity $n > 0$, and $d_1,\dots,d_n$ are values retrieved from the sources.

We now present in detail how mapping assertions are specified. Given a source schema $\mathscr{S}$ and a global schema $\mathscr{G}$, a *mapping assertion* from $\mathscr{S}$ to $\mathscr{G}$ is an expression of the form

$$\Phi(\mathbf{v}) \leadsto_t \Psi(\mathbf{w}),$$

where

- $\Phi(\mathbf{v})$, called the *body* of the mapping, is a first-order logic (FOL) query of arity $n > 0$, with distinguished variables (i.e., free variables) $\mathbf{v}$, over the source schema $\mathscr{S}$;
- $\Psi(\mathbf{w})$, called the *head*, is a first-order query over the global schema $\mathscr{G}$, whose distinguished variables $\mathbf{w}$ appear in the body, i.e., are variables in $\mathbf{v}$. The atoms in $\Psi(\mathbf{w})$ are built over the variables $\mathbf{w}$ and over terms, each one of the form $f(w_1,\dots,w_n)$ where f is a function symbol of arity $n > 0$ and $w_1,\dots,w_n$ are variables in $\mathbf{w}$;
- the subscript t stands for the *type* of the mapping assertion, which specifies the underlying semantic assumption. We distinguish the following types of mapping assertions:
  - $\leadsto_s$, denoting a *sound* mapping assertion, with the intended meaning that the data in the answer to the query over the sources, modulo the construction of object identifiers by means of function symbols, are a subset of the data in the answer to the query over the global schema;
  - $\leadsto_c$, denoting a *complete* mapping assertion, whose intended meaning is the converse of that of a sound assertion;
  - $\leadsto_e$, denoting an *exact* mapping assertion, whose intended meaning is that of an assertion that is both sound and complete.

For specifying the two queries in a mapping assertion, we use the languages that are commonly used for querying respectively relational databases and OWL ontologies, and that capture the expressive power of FOL queries. More precisely, we specify the body of the mapping in SQL, whereas for the head we use the SPARQL syntax [4].

In the following, we will denote a *data integration system* $\mathscr{I}$ as a tuple $\langle \mathscr{G}, \mathscr{S}, \mathscr{M} \rangle$, where $\mathscr{G}$ denotes the global schema, $\mathscr{S}$ denotes the source schema, and $\mathscr{M}$ denotes the *mapping*, which is a set of mapping assertions between $\mathscr{S}$ and $\mathscr{G}$.

[4] `http://www.w3.org/TR/rdf-sparql-query/`

SubClassOf(*Dean Professor*)
SubClassOf(*University Organization*)
ObjectPropertyDomain(*advisor Student*)
ObjectPropertyDomain(*headOf Dean*)
ObjectPropertyDomain(*takesCourse Student*)
DisjointClasses(*Student Professor*)
SubClassOf(*College Organization*)
ObjectPropertyRange(*advisor Professor*)
ObjectPropertyRange(*headOf College*)
ObjectPropertyRange(*takesCourse Course*)
FunctionalObjectProperty(*headOf*)
EquivalentClasses(*Person* ObjectUnionOf(*Student Professor*))
SubClassOf(*Student* ObjectSomeValuesFrom(*takesCourse Course*))
SubClassOf(*Professor* ObjectSomeValuesFrom(*worksFor University*))
SubClassOf(*Professor* ObjectSomeValuesFrom(*teacherOf Course*))
SubClassOf(*Dean* ObjectSomeValuesFrom(*headOf College*))
DisjointClasses(*Dean* ObjectSomeValuesFrom(*teacherOf Course*))

Fig. 14.1 Global schema of Example 14.1.

Example 14.1. We now present an example of data integration system extracted from a real integration experiment involving data from different sources in use at SAPIENZA University of Rome. The global schema that we adopt is defined by means of the OWL 2 assertions shown in Figure 14.1. It is in fact a portion of the Lehigh University Benchmark (LUBM) ontology[5], an ontology that is commonly used for testing ontology-based applications in the Semantic Web. In particular, the global schema contains the classes *Person*, *Student*, *Professor*, *Organization*, *College*, *Dean*, and *Course*, and the object properties *headOf*, *worksFor*, *takesCourse*, and *advisor*. For the sake of simplicity, we do not report in this example assertions involving data properties, but they are obviously allowed in our framework.

The source schema is a set of relational tables resulting from the federation of several data sources of the School of Engineering of the SAPIENZA University of Rome, and the portion that we consider in this example is constituted by the relational tables shown in Figure 14.2.

As for the mapping, referring to the global and source schemas presented above, we provide in Figure 14.3 some sample mapping assertions[6].

faculty(*UNIVERSITY_CODE, CODE, DESCRIPTION*)
students(*ID, FNAME, SNAME, DOB, ADDRESS*)
course(*FACULTY_CODE, CODE, DESCRIPTION*)
assignment(*COURSE_CODE, PROFESSOR, YEAR*)
professor(*CODE, FNAME, SNAME, ADDRESS, PHONE*)
exam(*STUD_ID, COURSE_CODE, DATE, RATING*)
career(*STUD_ID, ACADEMIC_YEAR, FACULTY_CODE*)
degree(*STUD_ID, YEAR, PROF_ID, TITLE*)

Fig. 14.2 Relational tables of the source schema of Example 14.1.

[5] `http://swat.cse.lehigh.edu/projects/lubm/`

[6] With some abuse of notation we make use, in the SPARQL query in the head of the mapping assertions, of the logical terms that construct object identifiers, that is, **st**(`ID`), **pr**(`PROF_ID`) and **st**(`STUD_ID`) which have to be considered as simple SPARQL variables.

M_1:

```
SELECT ID
FROM STUDENTS
WHERE DOB <= '1990/01/01'
```

$\leadsto_s$

```
SELECT ?st(ID) {
  ?st(ID) rdf:type Student
}
```

M_2:

```
SELECT STUD_ID,PROF_ID
FROM DEGREE
WHERE YEAR > 2000
```

$\leadsto_e$

```
SELECT ?st(STUD_ID) ?pr(PROF_ID){
  ?st(STUD_ID) advisor ?pr(PROF_ID)
}
```

M_3:

```
SELECT STUD_ID
FROM EXAM
WHERE COURSE_CODE NOT IN
  (SELECT COURSE_CODE
   FROM ASSIGNMENT
   WHERE YEAR < 1990)
```

$\leadsto_s$

```
SELECT ?st(STUD_ID) {
  ?st(STUD_ID) advisor ?X
}
```

Fig. 14.3 Mapping assertions of Example 14.1.

The mapping assertion M_1 specifies that the tuples from the source table students provide the information needed to build the instances of the class *Student*. In particular, the query in the body of M_1 retrieves the code for the students whose date of birth is before 1990; each such code is then used to build the object identifier for the student by means of the unary function symbol **st**. Similarly, the mapping M_2 extracts data from the table degree, containing information on the student's master degree, such as the year of the degree, the title of the thesis, and the code of the advisor. The tuples retrieved by the query in the body of M_2, involving only degree titles earned after 2000, are used to build instances for the object property *advisor*: the instances are constructed by means of the function symbols **pr** and **st**. Finally, the mapping assertion M_3 contributes to the construction of the domain of *advisor*, taking from the source table exam only codes of students that have passed the exam of courses that have never been assigned to some professor before 1990. Notice that M_1 and M_3 are sound mapping assertions, whereas M_2 is an exact mapping assertion. □

The semantics of a data integration system $\mathscr{I} = \langle \mathscr{G}, \mathscr{S}, \mathscr{M} \rangle$ is defined with respect to a database instance D for the source schema $\mathscr{S}$, and is given in terms of first-order interpretations of the global schema $\mathscr{G}$: an interpretation $I = (\Delta^I, \cdot^I)$ is a *model* for the data integration system $\mathscr{I}$, if it satisfies all the assertions in $\mathscr{G}$, i.e., it is a model of $\mathscr{G}$, and it satisfies the mapping assertions in $\mathscr{M}$ with respect to D. As for the notion of satisfying global assertions, we adopt the standard OWL semantics[7], whereas to give the precise definition of mapping satisfaction, we need to introduce some preliminary notions.

Let γ be a FOL formula with free variables $\mathbf{x} = (x_1, \ldots, x_n)$, and let $\mathbf{s} = (s_1, \ldots, s_n)$ be a tuple of values. A ground instance $\gamma[\mathbf{x}/\mathbf{s}]$ of γ is obtained from γ by substituting every occurrence of x_i with s_i, for $i \in \{1, \ldots, n\}$.

Now we are ready to formally define the semantic interpretation of mapping assertions, which reflects the different semantic assumptions that one wants to adopt

[7] http://www.w3.org/TR/owl2-semantics/

DEGREE			
STUD_ID	**YEAR**	**PROF_ID**	**TITLE**
1001	2003	P12	Handling Incomplete Information in Data Integration

EXAM			
STUD_ID	**COURSE_CODE**	**DATE**	**RATING**
1002	B15	1999-10-10	27

Fig. 14.4 Source database of Example 14.2.

and that influence the meaning of mapping satisfaction. Specifically: (*i*) a *sound* mapping assertion corresponds to a first-order logic implication from the sources to the global schema (intuitively, the body of the assertion logically implies its head); (*ii*) a *complete* mapping assertion corresponds to a first-order logic implication from the global schema to the sources (intuitively, the head of the assertion logically implies its body); finally (*iii*) an *exact* mapping assertion is one that is both sound and complete, i.e., logical implication is in both directions.

Formally, we say that an interpretation $I = (\Delta^I, \cdot^I)$ *satisfies* the mapping assertion

$$\Phi(\mathbf{v}) \leadsto_t \Psi(\mathbf{w})$$

with respect to D, if for every ground instance $\Phi[\mathbf{v}/\mathbf{s}] \leadsto_t \Psi[\mathbf{v}/\mathbf{s}]$ of the mapping we have that

- $\Phi[\mathbf{v}/\mathbf{s}]^D = \mathit{true}$ implies $\Psi[\mathbf{v}/\mathbf{s}]^I = \mathit{true}$, if the mapping is sound, i.e., $t = s$,
- $\Psi[\mathbf{v}/\mathbf{s}]^I = \mathit{true}$ implies $\Phi[\mathbf{v}/\mathbf{s}]^D = \mathit{true}$, if the mapping is complete, i.e., $t = c$,
- $\Phi[\mathbf{v}/\mathbf{s}]^D = \mathit{true}$ if and only if $\Psi[\mathbf{v}/\mathbf{s}]^I = \mathit{true}$, if the mapping is exact, i.e., $t = e$,

where $\Psi[\mathbf{v}/\mathbf{s}]^I$ (resp., $\Phi[\mathbf{v}/\mathbf{s}]^D$), denotes the evaluation of a (ground) FOL formula over the interpretation I (resp., database D) [2].

In our framework, we allow mapping assertions to be either sound or exact, which are the two semantic assumptions most commonly adopted and more thoroughly studied in data integration.

Finally, queries posed by users over the global schema are FOL queries (specified in the SPARQL syntax). Given a FOL query q, with free variables $\mathbf{x}$, expressed over a data integration system $\mathscr{J}$, the set of *certain answers* to q over $\mathscr{I}$ with respect to a source database D is the set of tuples $\mathbf{c}$ of constants from D such that $q[\mathbf{x}/\mathbf{c}]^I = \mathit{true}$ for every model I of $\mathscr{I}$, i.e., $\mathbf{c}$ is in the answer to q in every model I of $\mathscr{I}$.

Example 14.2. To see the different semantic behavior of sound and exact mappings, we refer again to the data integration system presented in Example 14.1, and consider the sample source database D given in Figure 14.4 (notice that we are assuming that all non-mentioned source tables are empty).

It is easy to see that no interpretation of the global schema satisfies the mapping. Intuitively, this is caused by the fact that mapping assertion M_3 forces $\mathbf{st}(1002)$ to be someone for which an advisor exists, but this contradicts the exact assumption on the mapping assertion M_2, which then implies that 1002 must be a $STUD\_ID$ of any tuple in the extension of the relational table degree, but this tuple does not exist in the source database D. Since there is no interpretation satisfying the mapping, there is no model for the data integration system, thus the system is considered inconsistent. Conversely, if we assume that M_2 is a sound mapping, we have that the system is no longer inconsistent, since M_2 is no longer contradicted by the fact that $\mathbf{st}(1002)$ has an advisor.

Let us now consider the following query over the system (in which M_2 is assumed to be sound).

```
SELECT ?S {
   ?S  rdf:type  Student
}
```

Intuitively, the query is asking for all the students. Now, since everyone that has an advisor is a student (according to the assertion ObjectPropertyDomain(*advisor Student*)), we can easily conclude (according to the mapping) that the set of certain answers to the query with respect to the source database D is $\{\mathbf{st}(1001), \mathbf{st}(1002)\}$. □

14.3 Computational characterization of query answering

The framework we have presented in the above section explains how semantic web languages like OWL or SPARQL can be used for data integration. The resulting setting turns out to be really expressive, since it allows for very expressive mappings (GLAV) that can be interpreted under different semantic assumptions (sound and exact), an expressive global schema (OWL), and expressive query languages, both over the global schema and over the sources. This expressiveness is useful for modelling purposes: however, it makes reasoning in such a framework problematic from a computational point of view.

In this section, we study computational issues connected to query answering, with the aim of showing which are the sources of complexity in the given framework that make this task difficult to deal with (or even impossible, with a sound and complete procedure). What we want to highlight here is that the use in ontology-based data integration systems of standard or common languages and assumptions adopted in Semantic Web, Knowledge Representation, and Database applications, requires careful attention. Indeed, reasoning over ontologies and data integration systems means reasoning over incomplete information, which is in general a hard task. As we will see in the following, in order to keep the complexity of query answering low, several limitations need to be imposed over the framework given in Section 14.2. Ideally, the complexity we aim at is the one required for evaluation of SQL queries over relational databases. This complexity, indeed, would allow us to rely over relational database technology that is nowadays the only available technology for dealing with the large amounts of data (e.g., of the order of millions of instances) typical of data integration applications. In other words, we aim at rightsizing the framework in such a way that query answering is reducible to plain evaluation of an SQL query posed over a relational DBMS or data federation tool which is in charge of managing the data sources. Formally, such a property is called *FOL-reducibility* of query answering (since FOL queries can be straightforwardly expressed in SQL) [19]. In terms of computational complexity, this means that the whole query answering process is in AC^0 in data complexity, i.e., the complexity measured only in the size of the data [2][8].

We proceed as follows:

1. We first show that query answering in the general framework is undecidable, and that the cause of undecidability resides in the use of FOL as user query language and mapping language. We then renounce to having FOL queries over the global schema, both as users' queries and as mapping queries, and adopt Union of Conjunctive Queries (UCQs), i.e., FOL queries expressible as a union of select-project-join SQL queries;
2. We then show that, even if the global schema is a flat schema (i.e., a set of predicates, without any terminological axiom over them), query answering of UCQs is intractable. There are two independent sources of complexity for this intractability: the presence of exact mappings, and the use of UCQs over the global schema in the head of GLAV mappings;
3. We finally show that, even if we do not allow for exact mappings and UCQs in the head of mapping assertions, when the global schema is specified in OWL, query answering is intractable as well. Therefore, OWL as global schema language turns out to be a third source of complexity for intractability.

Undecidability of query answering mentioned at the first point is a straightforward consequence of the undecidability of the validity problem in FOL, i.e., checking wether a FOL sentence evaluates to true in every interpretation. Indeed, a reduction can be constructed from such a problem to a query answering problem in data integration, exploiting the fact that the user query language is FOL. Furthermore, even if the user may only ask atomic queries (i.e., we disallow FOL as user query language), it is still possible to reduce FOL validity to query answering in data integration by exploiting the fact that the query in the head of mapping assertions is a FOL query. Notice that this holds even if the global schema is simply a set of concept and property predicates, and mappings are sound LAV mappings. Therefore, to have decidable query answering, queries over the global schema (both user queries and head mapping queries) cannot be expressed in FOL.

We therefore decide to limit the expressive power of the query language in order to have decidable query answering. At the same time, however, we need flexible mechanisms for extracting data from an integration system, such as those ensured by database query languages. Traditional DL inference tasks, like instance checking and instance retrieval [42, 81], are indeed not suited for this purpose, since they cannot refer to the same object via multiple navigation paths in the ontology, i.e., they

[8] The complexity class AC^0 is essentially the one that corresponds to the data complexity of evaluating a first-order (i.e., an SQL) query over a relational database.

allow only for a limited form of join, namely chaining. UCQs provide a good trade-off between expressive power and nice computational properties. They are also the most expressive fragment of FOL for which query answering over expressive DLs has been shown to be decidable [8, 23].

Even though it is open whether answering UCQs over ontologies specified in OWL 2 DL (or in its DL counterpart $\mathcal{SROIQ}(\mathcal{D})$) is decidable, we will see in the following that such a choice causes intractability of query answering already for ontologies expressed in fragments of OWL 2 DL, and therefore some further right-sizing of the framework is needed to achieve our ultimate goal of having query answering in AC^0.

Before delving into this aspect, however, we first notice that having UCQs as the language for expressing the query in the head of mapping assertions has a bad computational impact, even when the global schema is a plain theory, i.e., without axioms. Indeed, as shown in [1], answering UCQs is coNP-hard in data complexity in the presence of (either exact or sound) LAV mappings under a global schema that is a flat relational schema. For the case of sound mappings only, the intractability holds if queries over the global schema in the mapping assertions are UCQs, while for the case of exact mappings, the intractability holds even if such queries are CQs, i.e., disjunction is not allowed. This obviously implies that query answering in our framework, independently from the language adopted for the global schema, is coNP-hard in the size of the data. As further shown in [1], the combination of sound mappings and mapping queries over the global schema that are Conjunctive Queries (CQs) leads to membership of query answering in PTIME. For modeling reasons, we do not want to renounce to the expressive power of CQs over the global schema in mapping assertions, thus the above results force us to discard exact mappings and to rely on sound mappings only.

Finally, we turn back to UCQs as user query language over the OWL 2 DL global schema. Recent studies on DLs have in fact shown that query answering of UCQs is coNP-complete in data complexity for fragments of OWL 2 DL that correspond to expressive DLs of the $\mathcal{SH}$ family, which constitutes the logical underpinning of the OWL languages [78, 52]. In fact, in [18] it was already shown that query answering of CQs is coNP-hard already for very simple ontology languages (fragments of OWL 2 DL) allowing for some basic ontology constructs, such as covering (which allows to say, for instance, that the set of persons is the union of men and women). Therefore, there is no hope to have tractable query answering in a framework in which the global schema is specified in OWL 2 DL. To overcome this problem, in the next section we present a suitable fragment of OWL 2 DL characterized by a good trade-off between expressive power and complexity of reasoning, and for which query answering turns out to be tractable (in fact in AC^0) in data complexity.

14.4 Data integration using OWL 2 QL

In this section, we analyze data integration under global schemas expressed in OWL 2 QL, a tractable profile[9] of OWL 2, which is essentially based on the description logic *DL-Lite*$_R$ [19]. We will show that expressing the global schema in OWL 2 QL allows for efficient query answering. More precisely query answering is first-order rewritable, i.e., it can be reduced to evaluation of a first-order query (directly expressible in SQL) over a database instance, provided that the user's query is a UCQs, each mapping assertion is sound, and has a FOL query over the source schema in its head and a CQ over the global schema in its body (cf. Section 14.2 and Section 14.3).

To ease exposition, we will focus here on *DL-Lite*$_R$, the DL on which OWL 2 QL is essentially based, but we remark that all the results and the techniques for *DL-Lite*$_R$ presented here also hold when the global schema is expressed instead in OWL 2 QL. For a more detailed description of OWL 2 QL, we refer the reader to the W3C technical report on the OWL 2 profiles.

To introduce *DL-Lite*$_R$, we use the following notation:

- A denotes an *atomic concept*, B a *basic concept*, and C a *general concept*;
- P denotes an *atomic role*, Q a *basic role*, and R a *general role*.

Then, *DL-Lite*$_R$ expressions are defined as follows[10].

- Basic and general *concept expressions*:

  $$B ::= A \mid \exists Q \qquad\qquad C ::= \top \mid B \mid \neg B \mid \exists Q.C$$

  where the concept $\top$ is the universal concept, and corresponds to the OWL class `owl:Thing`, which denotes the set of all individuals; $\neg B$ denotes the negation of a basic concept B; the concept $\exists Q$ is the unqualified existential restriction, which denotes the domain of a role Q, i.e., the set of objects that Q relates to some object, and corresponds to the OWL expression ObjectSomeValuesFrom(Q, `owl:Thing`); the concept $\exists Q.C$, also called qualified existential restriction, denotes the qualified domain of Q w.r.t. C, i.e., the set of objects that Q relates to some instance of C, and corresponds to the OWL expression ObjectSomeValuesFrom(Q, C).
- Basic and general *role expressions*:

  $$Q ::= P \mid P^- \qquad\qquad R ::= Q \mid \neg Q$$

  where P^- denotes the inverse of an atomic role, and $\neg Q$ denotes the negation of a basic role.

[9] `http://www.w3.org/TR/owl2-profiles/`

[10] *DL-Lite*$_R$ can be easily extended with attribute expressions, i.e., assertions involving data properties (cf. [79]). For the sake of simplicity we do not introduce such expressions here (they are however available in OWL 2 QL).

A *DL-Lite*$_R$ *TBox* allows one to represent intensional knowledge by means of *inclusion assertions*, i.e., expressions of the following forms:

$$\begin{array}{ll} B \sqsubseteq C & \textit{concept inclusion assertion} \\ Q \sqsubseteq R & \textit{role inclusion assertion} \end{array}$$

A concept inclusion assertion expresses that a (basic) concept B is subsumed by a (general) concept C. Analogously for role inclusion assertions.

For the semantics of *DL-Lite*$_R$ knowledge bases (constituted by a TBox and an ABox, which specifies the instances of concept and roles) we refer the reader to [19]. As shown in [19], reasoning over a *DL-Lite*$_R$ knowledge base is tractable. More precisely, TBox reasoning is in NLOGSPACE and answering unions of conjunctive queries is FOL-reducible, and hence in AC^0 w.r.t. data complexity. Thus, *DL-Lite*$_R$ appears particularly suited for integration of large amounts of data.

We now analyze data integration systems of the form $\mathscr{I} = \langle \mathscr{G}, \mathscr{S}, \mathscr{M} \rangle$ in which $\mathscr{G}$ is a *DL-Lite*$_R$ TBox, $\mathscr{S}$ is a relational schema, and $\mathscr{M}$ is a set of sund GLAV mappings from $\mathscr{S}$ to $\mathscr{G}$, of the form described above. More precisely, given such a system $\mathscr{I}$, a database instance D for the source schema $\mathscr{S}$, and a UCQ q over $\mathscr{G}$, we want to compute the certain answers to q over $\mathscr{I}$ w.r.t. D. We assume that $\mathscr{I}$ is satisfiable with respect to D, i.e., there exists a model for $\mathscr{I}$ and D.

The query answering algorithm is constituted by the following four steps, which we describe in the following:

1. *Schema-Rewriting*
2. *LAV-Rewriting*
3. *GAV-Rewriting*
4. *Source-Evaluation*

Schema-Rewriting

Given a UCQ Q over a data integration system $\mathscr{I} = \langle \mathscr{G}, \mathscr{S}, \mathscr{M} \rangle$, and a source database D for $\mathscr{I}$, the Schema-Rewriting step computes a new UCQ Q' over $\mathscr{I}$, where the assertions of $\mathscr{G}$ are compiled in. In computing the rewriting, only inclusion assertions of the form $B_1 \sqsubseteq B_2$, and $Q_1 \sqsubseteq Q_2$ are taken into account, where B_1 and B_2 are basic concepts, and Q_1 and Q_2 are basic roles. Intuitively, the query Q is rewritten according to the knowledge specified in $\mathscr{G}$ that is relevant for answering Q, in such a way that the rewritten query Q' is such that the certain answers to Q over $\mathscr{I}$ and D are equal to the certain answers to Q' over $\mathscr{I}' = \langle \emptyset, \mathscr{S}, \mathscr{M} \rangle$ and D, i.e., the Schema-Rewriting step allows to get rid of $\mathscr{G}$.

We refer the reader to [19] for a formal description of the Schema-Rewriting algorithm and for a proof of its soundness and completeness. We only notice here that the Schema-Rewriting step does not depend on the source database D, runs in polynomial time in the size of $\mathscr{G}$, and returns a query Q' whose size is at most exponential in the size of Q.

LAV-Rewriting

Given the UCQ Q' over $\mathscr{I}$ computed by the Schema-Rewriting step, the LAV-Rewriting step considers the "LAV-part" of the mapping $\mathscr{M}$, denoted by $LAV(\mathscr{M})$ in the following.

First, we define the "LAV part" and the "GAV part" of a GLAV mapping $\mathscr{M}$ as follows. Let $\mathscr{M} = \{M_1,\ldots,M_k\}$ be a set of k mapping assertions[11]. Suppose the GLAV mapping assertion M_i is of the form

$$SQL_s(x_1,\ldots,x_n) \rightsquigarrow_s CQ_g(x_1,\ldots,x_n)$$

where $SQL_s(x_1,\ldots,x_n)$ is an SQL query over $\mathscr{S}$ and $CQ_g(x_1,\ldots,x_n)$ is a conjunctive query over $\mathscr{G}$, both with distinguished variables $x_1,\ldots,x_n$. Then, we define M_i' as the LAV mapping assertion

$$aux_i(x_1,\ldots,x_n) \rightsquigarrow_s CQ_g(x_1,\ldots,x_n)$$

where aux_i is a new auxiliary relation symbol of arity n, and define M_i'' as the GAV mapping assertion

$$SQL_s(x_1,\ldots,x_n) \rightsquigarrow_s aux_i(x_1,\ldots,x_n)$$

Then, we denote by $LAV(\mathscr{M})$ the set of LAV mappings $\{M_1',\ldots,M_k'\}$, and we denote by $GAV(\mathscr{M})$ the set of GAV mappings $\{M_1'',\ldots,M_k''\}$. Notice that each $\mathscr{M}'$ is a set of LAV mappings from the set of relations $aux_1,\ldots,aux_k$ to $\mathscr{G}$. In other words, such auxiliary relations play the role of source predicates in $\mathscr{M}'$, whereas they have to be considered global predicates in M_i''.

Now, the LAV-Rewriting step reformulates the UCQ Q' to a UCQ Q'' over the set $AUX = \{aux_1,\ldots,aux_k\}$ of auxiliary relations in such a way that, for every database instance D_{aux} for the schema AUX, the set of certain answers to Q' over $\langle\emptyset, AUX, LAV(\mathscr{M})\rangle$ w.r.t. D_{aux} is equal to the evaluation of the query Q'' over D_{aux}. This is realized by applying any of the well-known methods that are able to rewrite a union of conjunctive queries with respect to a set of LAV mappings (e.g., the inverse rules algorithm [1] or the Minicon algorithm [80]). Moreover, the LAV-Rewriting procedure does not depend on D and runs in polynomial time in the size of $LAV(\mathscr{M})$ (i.e., in the size of $\mathscr{M}$).

GAV-Rewriting

Given the UCQ Q'' over the auxiliary schema AUX computed by the previous step, and the set of GAV mappings $GAV(\mathscr{M})$, the GAV-Rewriting step computes, by using logic programming technology, an SQL query Q''' over the source schema $\mathscr{S}$. It can be shown (see [79], where this step is called the *unfolding* step) that Q''' is such

[11] To ease the exposition we do not consider in the following the presence in the mapping of logical terms that construct object identifiers. The treatment can be easily generalized to this case.

that, for every database instance D for the schema $\mathscr{S}$, the set of certain answers to Q'' over $\langle \emptyset, \mathscr{S}, GAV(\mathscr{M}) \rangle$ w.r.t. D is equal to the evaluation of the query Q''' over D. Moreover, the GAV-Rewriting step does not depend on D, runs in polynomial time in the size of $GAV(\mathscr{M})$ (i.e., in the size of $\mathscr{M}$), and returns a query whose size is polynomial in the size of Q''.

Source-Evaluation

The final step consists in simply evaluating the SQL query Q''', produced by the GAV-Rewriting step, over D. Notice that, to actually perform such an evalutation, a data federation tool managing the data sources is needed, because the query Q''' is distributed over several autonomous data sources.

It can be shown that the query answering procedure described above correctly computes the certain answers to UCQs, i.e., for every database instance D for the schema $\mathscr{S}$, the set of certain answers to Q over $\langle \mathscr{G}, \mathscr{S}, \mathscr{M} \rangle$ w.r.t. D is equal to the evaluation of the query Q''' over D. Based on the computational properties of such an algorithm, we can then characterize the complexity of our query answering method.

Theorem 14.1. *Let $\mathscr{I} = \langle \mathscr{G}, \mathscr{S}, \mathscr{M} \rangle$ be a data integration system where $\mathscr{G}$ is a DL-Lite$_R$ TBox and $\mathscr{M}$ is a set of GLAV mappings, and let D be a source database for $\mathscr{I}$. Answering a UCQ over $\mathscr{I}$ with respect to D can be reduced to the evaluation of an SQL query over D, i.e., it is FOL-reducible, and it is in AC^0 in the size of D.*

Finally, we remark that, as we said at the beginning of this section, we have assumed that the data integration system $\mathscr{I}$ is satisfiable with respect to the database D. Notably, it can be shown that all the machinery we have devised for query answering can also be used for checking satisfiability of $\mathscr{I}$ with respect to D. Therefore, checking satisfiability can also be reduced to sending appropriate SQL queries to the source database [79].

Example 14.3. Consider again Example 14.1. According to what has been said in Section 14.3 and in this section, we have to renounce to some of the modeling choices initially adopted. To this aim, we substitute M_2 with the mapping assertion

```
M2':  SELECT STUD_ID,PROF_ID  ~>s  SELECT ?st(STUD_ID) ?pr(PROF_ID){
      FROM DEGREE                    ?st(STUD_ID) advisor ?pr(PROF_ID)
      WHERE YEAR > 2000            }
```

which is analogous to M_2 but is sound rather than exact. Furthermore, we adapt the global schema described in 14.1 in such a way to obtain its OWL 2 QL approximation. In particular, we eliminate the assertion FunctionalObjectProperty(*headOf*), since functionality is not allowed in OWL 2 QL, and approximate EquivalentClasses(*Person* ObjectUnionOf(*Student Professor*)) with the two assertions SubClassOf(*Student Person*) and SubClassOf(*Professor Person*), since the use of the union

$$\begin{array}{ll}
\mathit{Dean} \sqsubseteq \mathit{Professor} & \mathit{Student} \sqsubseteq \neg\mathit{Professor} \\
\mathit{University} \sqsubseteq \mathit{Organization} & \mathit{College} \sqsubseteq \mathit{Organization} \\
\exists \mathit{advisor} \sqsubseteq \mathit{Student} & \exists \mathit{advisor}^{-} \sqsubseteq \mathit{Professor} \\
\exists \mathit{headOf} \sqsubseteq \mathit{Dean} & \exists \mathit{headOf}^{-} \sqsubseteq \mathit{College} \\
\exists \mathit{takesCourse} \sqsubseteq \mathit{Student} & \exists \mathit{takesCourse}^{-} \sqsubseteq \mathit{Course} \\
\mathit{Student} \sqsubseteq \mathit{Person} & \mathit{Professor} \sqsubseteq \mathit{Person} \\
\mathit{Student} \sqsubseteq \exists \mathit{takesCourse}.\mathit{Course} & \\
\mathit{Professor} \sqsubseteq \exists \mathit{worksFor}.\mathit{University} & \\
\mathit{Professor} \sqsubseteq \exists \mathit{teacherOf}.\mathit{Course} & \\
\mathit{Dean} \sqsubseteq \exists \mathit{headOf}.\mathit{College} & \\
\mathit{Dean} \sqsubseteq \neg\exists \mathit{teacherOf}.\mathit{Course} &
\end{array}$$

Fig. 14.5 Global schema of Example 14.3 in the *DL-Lite*$_R$ syntax.

in the right-hand side of inclusion assertions is not allowed in OWL 2 QL. In Figure 14.5 we give the refined global schema expressed in the *DL-Lite*$_R$ syntax.

Then, consider again the query asking for all students

```
SELECT ?S {
   ?S rdf:type Student
}
```

and process it according to the query answering algorithm described above. After the schema-rewriting step we obtain the following query

```
SELECT ?S {
   ?S rdf:type Student
}
   UNION
SELECT ?S {
   ?S advisor ?X
}
   UNION
SELECT ?S {
   ?S takesCourse ?X
}
```

To get the intuition behind the schema-rewriting step, consider that, in this example, the global schema specifies that everyone that has an advisor is a student ($\exists \mathit{advisor} \sqsubseteq \mathit{Student}$), and that everyone that takes a course is a student ($\exists \mathit{takesCourse} \sqsubseteq \mathit{Student}$). After the schema rewriting step, this knowledge is compiled in the query, which now, besides asking for (explicit) students, asks also for those that have an advisor and those that take a course. This rewritten query can be now evaluated over the system in which the global schema is considered empty.

Let us now consider the mapping. We notice that in this example the "GAV part" of the mapping is constituted by the two GAV assertions M_1 and M_2 and the "LAV part" is constituted by the LAV assertion M_3[12]. Therefore, there is no need to introduce auxiliary predicates. Then, in the LAV-rewriting step we rewrite the query

[12] M_3 is a LAV assertion since not all the variables occurring in the head are mapped to distinguished variables in the body, i.e., M_3 has existentially quantified variables in the head.

according to M_3, whereas in the GAV-rewriting step we rewrite it on the basis of M_1 and M_2. We thus obtain the following query, which we express in SQL[13].

```
SELECT st(ID) FROM STUDENTS
WHERE DOB <= '1990/01/01'
  UNION
SELECT st(STUD_ID) FROM DEGREE
WHERE YEAR > 2000
  UNION
SELECT st(STUD_ID) FROM EXAM
WHERE COURSE_CODE NOT IN
  (SELECT COURSE_CODE FROM ASSIGNMENT
   WHERE YEAR < 1990)
```

It is easy to see that the evaluation of such a query over the source database D given in Figure 14.4 returns the set of certain answers $\{\mathbf{st}(1001), \mathbf{st}(1002)\}$. □

14.5 Related Work

Research in information integration has been very active in the last fifteen years, and has produced a number of results and technical contributions, from both the theoretical and practical points of view [85, 62, 70, 68].

The setting that has been more deeply investigated is the traditional centralized (as known as mediator-based) data integration setting, where the integration is performed through a centralized global schema connected to sources via semantic mappings. As already said, this is also the setting considered in the present chapter.

Among the various problems related to data integration, the problem of answering queries posed over the global schema is the one that has been addressed most intensively. The first proposals, developed in the middle 90s, faced such a problem in a procedural way, thus not providing the users with a real declarative support to data integration. Systems like TSIMMIS (The Stanford-IBM Manager of Multiple Information Sources) [33], or Garlic [31] are essentially hierarchies of wrappers (cf. Section 14.2) and mediators, which are in charge of triggering the right wrappers and putting together the data that they return into the final answers to users' queries (or feeding in turn other mediators). Both TSIMMIS and Garlic can be considered primitive forms of systems adopting GAV mapping, but, since each mediator works in an independent manner, it turns out that no real integration is ever achieved.

A different (declarative) approach to the problem of query answering in data integration has been instead followed in the setting in which LAV mapping is considered. In such setting, query answering calls for facing the issue of query processing using views [85, 73, 70, 27], i.e., the issue of how to use the information about the global schema, the mappings, and the data stored at the sources, to answer the

[13] In the SELECT clause, with some abuse of notation, we make use of the logical terms that construct object identifiers. These logical terms can be easily obtained resorting to SQL functions for string manipulation, as suggested in [79].

users' queries posed over the global schema. Two notable examples of systems following such approach in a relational context are Information Manifold (IM) [74, 75], and INFOMASTER [51, 1, 43]. For query answering, both systems propose query rewriting procedures, called the bucket algorithm and the inverse rules algorithm, respectively. Such algorithms have been originally designed for the case in which both the users' and the mapping queries are CQs, but some extensions have been proposed that consider also more expressive settings (see, e.g., [43] and [59]). An interesting optimization for the bucket algorithm, called the Minicon algorithm, can be found in [80], where it is shown that Minicon significantly speeds up query processing with respect to previously proposed procedures. It is worth noticing that all the above mentioned algorithms in principle can be used in the LAV-Rewriting step of the query answering procedure presented in Section 14.4.

Many other studies have considered the query answering problem in data integration systems in various settings. For example, the relational setting (under various assumptions on the languages used for the mapping and the queries) has been analyzed in [55, 72], whereas the impact on query answering of specifying different forms of integrity constraints on a relational global schema has been considered in [43, 59] for LAV mappings, and in [15, 16] for GAV mappings. Also, query answering in the presence of semistructured data sources and global schemas has been considered in [28, 30, 29], and is still the subject of intensive investigations.

Despite the intensive research of the last years described so far, only few efforts have been dedicated to the study of data integration through conceptual models, and in particular through ontologies. This problem has been considered for example in [22], where the authors propose a formal framework for Ontology Integration Systems (OISs). Their view of a formal framework deals with a situation where there are various local ontologies, developed independently from each other, assisting the task to build an integrated, global ontology as a means for extracting information from the local ones. Ontologies in their framework are expressed as DL knowledge bases, and mappings between ontologies are expressed through suitable mechanisms based on queries, which actually correspond to the GAV and LAV approaches adopted in data integration.

We point out that most of the work carried out so far on ontology-based information integration is on which language or which method to use to build a global ontology on the basis of the local ones [13, 40], whereas the problem of querying an integrated ontology, i.e., a global schema of a mediator-based data integration system expressed in terms of an ontology, still needs further investigation. Among the first studies on this problem, we mention [14, 24, 25]. In particular, in [14], global schemas specified as UML class diagrams, translated into simple DL theories, are considered, and reasoning (i.e., query processing) over them is studied (in fact, the mediator-based integration framework is only envisaged in the paper, but the proposed techniques are easily extendible to GAV data integration systems). In [24], the use in data integration systems of logic theories expressed in *DLR*, an expressive DL allowing for the use of *n*-ary relationships, is investigated, but no complete query answering processing algorithms are provided. In [25], the above theoretical studies are applied to a data warehouse context.

When the global schema of a data integration system is specified in terms of a DL ontology, it is natural to look at the research carried out in the DL field and at the tools developed for specifying and reasoning over DL knowledge bases, and investigate if and how they can be applied in a data integration setting.

In this respect, it is worth noticing that current reasoners for expressive DLs perform indeed well in practice, and show that even procedures that are exponential in the size of the knowledge base might be acceptable under suitable conditions. This has been achieved during the years, thanks to the intensive research aimed at understanding the frontier between tractability (i.e., solvable by a polynomial time algorithm) and intractability of reasoning over concept expressions. The maximal combinations of constructs (among those most commonly used) that still guarantee polynomial time inference procedures were identified, which allowed to exactly characterize the tractability frontier [8, 41]. It should be noted that the techniques and technical tools that were used to prove such results, namely tableaux-based algorithms, are at the basis of the modern state of the art DL reasoning systems [77], such as Fact [64], Racer [60], and Pellet [84, 83]. However, such reasoners have not specifically been tailored to deal with large amounts of data, which is a critical issue in all those settings in which ontologies are used as a high-level, conceptual view over data repositories, as in data integration.

In data integration, data are typically very large and dominate the intensional level of the ontologies. Hence, while one could still accept reasoning that is exponential on the intensional part, it is mandatory that reasoning is polynomial in the data (i.e., in data complexity). Traditionally, research carried out in DLs has not paid much attention to the data complexity of reasoning, and only recently efficient management of large amounts of data [66, 34] has become a primary concern in ontology reasoning systems, and data complexity has been studied explicitly [67, 18, 78, 69, 6, 7]. Unfortunately, research on the trade-off between expressive power and computational complexity of reasoning has shown that many DLs with efficient reasoning algorithms lack the modeling power required for capturing conceptual models and basic ontology languages. An interesting line of research, followed, e.g., in [19, 79, 18], is therefore the one that is aimed at defining ontology languages which allow for both efficient reasoning, and in particular answering of complex queries as CQs, and good modeling power, that means enabling the specification of basic ontology constructs. The OWL2 QL profile described in Section 14.4 is indeed one of such languages.

It is worth noticing that all the above mentioned approaches to data integration do not consider the problem of data that may result inconsistent with respect to integrity constraints specified over the global schema. This is in fact a quite common situation, since data stored at autonomous sources in general are not required to be compliant with global integrity constraints. This problem is even more evident when the global schema is given in terms of an ontology, which provides the conceptualization of the domain of interest, constructed separately, and in principle independently, from the sources to be integrated.

In the cases in which data may contradict global integrity constraints, the main problem that arises is how to obtain significant answers from inconsistent systems.

Traditionally, the approach adopted to remedy this problem has been through data cleaning [11]. This approach is procedural in nature, and is based on domain-specific transformation mechanisms applied to the data retrieved from the sources. Only very recently, the first academic prototype implementations have appeared, which provide declarative approaches to the treatment of inconsistency of data, in the line of the studies on consistent query answering [4]. In such approaches, the common basic idea is that inconsistency might be eliminated by modifying the database representing the extension of the system (e.g., the models of the data integration system), and reasoning on the "repaired" database. Since several repairs are possible, the "consistent answers" are the answer to users' query returned by the evaluation of the query in every repair. In many papers, it has been proposed to formalize repair semantics by using logic programs [56, 16, 12]. The common idea is to encode the constraints of the global schema into a logic program, using unstratified negation or disjunction, such that the stable models of this program [50] yield the repairs of the global database, and the user query can be compiled in this program in such a way that its evaluation returns the consistent answers. This is for example the approach followed by the INFOMIX system [71]. INFOMIX processes users' queries (expressed in Datalog) posed over a relational global schema with key, inclusion, and exclusion dependencies, in a GAV setting, by means of a query rewriting technique that produces a Datalog program enriched with negation, under stable model semantics [16, 58].

Other interesting proposals on inconsistency management are the Hippo system [36, 35], and the ConQuer system [47, 49], which are focused on identifying cases in which computing consistent answers is tractable. However, such proposals have been essentially developed in the context of a single database system, and therefore do not deal with all aspects of a complex data integration environment.

A different approach in mediator-based information integration looks at data management under the perspective of exchanging data between the sources and the global schema, called the target schema in data exchange terminology. Data exchange has similar logical foundations to those of virtual mediator-based data integration discussed in the present paper. The basic notions of data exchange and its first formalization were given in [44]. In particular, a *solution* to the data exchange problem for a given source instance is a finite target instance that, together with the source instance, satisfies both *target dependencies*, i.e., constraints specified over the target schema, and *source-to-target dependencies*, i.e., mappings between the source and the target schema. A *universal solution* is a special solution that is homomorphic to every possible solution. Universal solutions are particularly important in data exchange, since, as shown in [44], the certain answers to a union of conjunctive queries q can be obtained by evaluating q over any universal solution.

Among all universal solutions, the *core* assumes a crucial importance, for being it the "smallest" one, and therefore in principle the best universal solution to compute [45]. The problem of computing the core is studied in [45, 53, 54], under different forms of target dependencies. In particular, in [54] a polynomial-time procedure for computing the core under classical data exchange constraints is provided.

Other works on data exchange studied, respectively, query answering (by first-order rewriting) for first-order logic (FOL) queries [3], schema mapping compositions [46], exchange of XML documents when both the source and the target schemas are XML DTDs [5], and relationship between data exchange and incomplete information [76].

Even if all the above studies are of great interest for ontology-based data integration, the role of ontologies in data exchange has not been investigated so far. Therefore, materialized data integration in the Semantic Web is a subject that still needs to be taken into account.

More recently, the issue of data integration has been considered in the more dynamic context of Peer-to-Peer (P2P) data management [61]. In a nutshell, a P2P system is characterized by an architecture constituted by various autonomous nodes that hold data and that are linked to other nodes by means of mappings.

Opposed to the first, “semantic-less” approaches, focused essentially on file sharing, data-oriented approaches to P2P have been proposed recently [61, 9, 57]. Differently from the traditional mediator-based setting, integration in data-oriented P2P systems is not based on a global schema. Instead, each peer represents an autonomous information system, and information integration is achieved by establishing P2P mappings, i.e., mappings among the various peers. Queries are posed to one peer, and the role of query processing is to exploit both the data that are internal to the peer, and the mappings between the peer and the other peers in the system.

While techniques for query answering and data exchange have been studied and developed extensively in the mediator-based setting, there is still a fundamental lack of understanding behind the basic issues of data integration in P2P systems. In particular, it needs to be investigated whether the usual approach of resorting to a first-order logic interpretation of P2P mappings (followed, e.g., by [32, 61, 9]), is still appropriate in the presence of possibly cyclic mappings, or whether alternative semantic characterizations should be adopted [26, 20]. Also, data exchange in a P2P setting still remains largely unexplored. Two exceptions are [48], where the problem is studied in a setting in which only two peers interact with different roles and capabilities, and [37], where a preliminary investigation of data exchange in a full-fledged P2P setting is proposed.

The case in which peers export an ontology (rather than a simple relational schema), has been studied in [82, 17]. In these papers it is shown that query answering in this setting is a very complex task, and to have tractable cases, very severe limitations have to be imposed on the expressivity of the ontology language and the form of P2P mappings, even in very simple settings, e.g., when the whole system is constituted by two peers, as in [17].

14.6 Conclusions

In this paper we have shown the impact on the computational complexity of query answering of adopting the OWL language (more precisely OWL 2 DL) for specifying the global schema of a data integration system. To precisely characterize this impact, we have chosen very expressive formalisms for instantiating the various components of a data integration system (GLAV sound and exact mappings, FOL queries over the global schema, both for the users' queries and head queries in mapping assertions). Even though interesting from a modeling point of view, we have shown that such choices soon lead to undecidability of query answering or, under some limitations, to intractability of query answering. We have identified the various sources of complexity, and have eliminated them by rightsizing the overall framework, in such a way that query answering turns out to be in principle as efficient as standard query processing in relational DBMSs. At the core of the new framework we have OWL 2 QL, a tractable fragment of OWL 2, which we use for specifying the global schema of data integration systems. OWL 2 QL is essentially a variant of *DL-Lite*$_R$ and presents the same nice computational behavior typical of all DLs of the *DL-Lite* family.

The approach illustrated in this paper is under development in a prototype system called MASTRO-I, a tool for ontology-based data integration supporting global schemas specified in the *DL-Lite* family (and therefore in OWL 2 QL), which makes use of the QUONTO[14] reasoner for *DL-Lite*. Such a tool is in fact able to deal with ontologies even more expressive than OWL 2 QL, allowing for the specification, in a controlled way, of functionality assertions and complex forms of identification constraints (see [21]), and is also able to answer full SQL queries, under a suitable semantic approximation.

Other aspects, which are important for the problem of semantic data integration, have not been addressed yet in the development of the system, but are under investigation. Among them we mention the problem of handling inconsistencies in the data, possibly using a declarative, rather than an ad-hoc procedural approach, in the line of the work on consistent query answering (cf. Section 14.5). A second interesting problem for further work is looking at "write-also" data integration, i.e., allowing support also for updates expressed on the global schema (e.g., in the line of the work described in [38, 39]). How to express an update formulated over the global ontology in terms of series of insert and delete operations executed over the underlying data sources is a challenging issue in this context.

References

1. Abiteboul, S., Duschka, O.: Complexity of answering queries using materialized views. In: Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98), pp. 254–265 (1998)
2. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley Publ. Co. (1995)
3. Arenas, M., Barcelo, P., Fagin, R., Libkin, L.: Locally consistent transformations and query answering in data exchange. In: Proc. of the 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2004), pp. 229–240 (2004)

[14] `http://www.dis.uniroma1.it/quonto/`

4. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'99), pp. 68–79 (1999)
5. Arenas, M., Libkin, L.: XML data exchange: Consistency and query answering. In: Proc. of the 24rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2005), pp. 13–24 (2005)
6. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: DL-Lite in the light of first-order logic. In: Proc. of the 22nd Nat. Conf. on Artificial Intelligence (AAAI 2007), pp. 361–366 (2007)
7. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The *DL-Lite* family and relations. Tech. Rep. BBKCS-09-03, School of Computer Science and Information Systems, Birbeck College, London (2009). Available at `http://www.dcs.bbk.ac.uk/research/techreps/2009/bbkcs-09-03.pdf`
8. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
9. Bernstein, P.A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., Zaihrayeu, I.: Data management for peer-to-peer computing: A vision. In: Proc. of the 5th Int. Workshop on the Web and Databases (WebDB 2002) (2002)
10. Bernstein, P.A., Haas, L.: Information integration in the enterprise. Communications of the ACM **51**(9), 72–79 (2008)
11. Bouzeghoub, M., Lenzerini, M.: Introduction to the special issue on data extraction, cleaning, and reconciliation. Information Systems **26**(8), 535–536 (2001)
12. Bravo, L., Bertossi, L.: Logic programming for consistently querying data integration systems. In: Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003), pp. 10–15 (2003)
13. Broekstra, J., Klein, M., Fensel, D., Horrocks, I.: Adding formal semantics to the Web: building on top of RDF Schema. In: Proc. of the ECDL 2000 Workshop on the Semantic Web (2000)
14. Calì, A., Calvanese, D., De Giacomo, G., Lenzerini, M.: Reasoning on UML class diagrams in description logics. In: Proc. of IJCAR Workshop on Precise Modelling and Deduction for Object-oriented Software Development (PMD 2001) (2001)
15. Calì, A., Calvanese, D., De Giacomo, G., Lenzerini, M.: Data integration under integrity constraints. Information Systems **29**, 147–163 (2004)
16. Calì, A., Lembo, D., Rosati, R.: Query rewriting and answering under constraints in data integration systems. In: Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003), pp. 16–21 (2003)
17. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: What to ask to a peer: Ontology-based query reformulation. In: Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004), pp. 469–478 (2004)
18. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006), pp. 260–270 (2006)
19. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. of Automated Reasoning **39**(3), 385–429 (2007)
20. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Inconsistency tolerance in P2P data integration: An epistemic logic approach. Information Systems **33**(4), 360–384 (2008)
21. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Path-based identification constraints in description logics. In: Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2008), pp. 231–241 (2008)
22. Calvanese, D., De Giacomo, G., Lenzerini, M.: 2ATAs make DLs easy. In: Proc. of the 2002 Description Logic Workshop (DL 2002), *CEUR Electronic Workshop Proceedings,* `http://ceur-ws.org/`, vol. 53, pp. 107–118 (2002)

23. Calvanese, D., De Giacomo, G., Lenzerini, M.: Conjunctive query containment and answering under description logics constraints. ACM Trans. on Computational Logic **9**(3), 22.1–22.31 (2008)
24. Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Description logic framework for information integration. In: Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98), pp. 2–13 (1998)
25. Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Data integration in data warehousing. Int. J. of Cooperative Information Systems **10**(3), 237–271 (2001)
26. Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Logical foundations of peer-to-peer data integration. In: Proc. of the 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2004), pp. 241–251 (2004)
27. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.Y.: What is query rewriting? In: Proc. of the 7th Int. Workshop on Knowledge Representation meets Databases (KRDB 2000), *CEUR Electronic Workshop Proceedings,* `http://ceur-ws.org/`, vol. 29, pp. 17–27 (2000)
28. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.Y.: Rewriting of regular expressions and regular path queries. J. of Computer and System Sciences **64**(3), 443–465 (2002)
29. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.Y.: View-based query processing: On the relationship between rewriting, answering and losslessness. In: Proc. of the 10th Int. Conf. on Database Theory (ICDT 2005), *Lecture Notes in Computer Science*, vol. 3363, pp. 321–336. Springer (2005)
30. Calvanese, D., De Giacomo, G., Vardi, M.Y.: Decidable containment of recursive queries. Theoretical Computer Science **336**(1), 33–56 (2005)
31. Carey, M.J., Haas, L.M., Schwarz, P.M., Arya, M., Cody, W.F., Fagin, R., Flickner, M., Luniewski, A., Niblack, W., Petkovic, D., Thomas, J., Williams, J.H., Wimmers, E.L.: Towards heterogeneous multimedia information systems: The Garlic approach. In: Proc. of the 5th Int. Workshop on Research Issues in Data Engineering – Distributed Object Management (RIDE-DOM'95), pp. 124–131. IEEE Computer Society Press (1995)
32. Catarci, T., Lenzerini, M.: Representing and using interschema knowledge in cooperative information systems. J. of Intelligent and Cooperative Information Systems **2**(4), 375–398 (1993)
33. Chawathe, S.S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J.D., Widom, J.: The TSIMMIS project: Integration of heterogeneous information sources. In: Proc. of the 10th Meeting of the Information Processing Society of Japan (IPSJ'94), pp. 7–18 (1994)
34. Chen, C., Haarslev, V., Wang, J.: LAS: Extending Racer by a Large ABox Store. In: Proc. of the 2005 Description Logic Workshop (DL 2005), *CEUR Electronic Workshop Proceedings,* `http://ceur-ws.org/`, vol. 147 (2005)
35. Chomicki, J., Marcinkowski, J., Staworko, S.: Computing consistent query answers using conflict hypergraphs. In: Proc. of the 13th Int. Conf. on Information and Knowledge Management (CIKM 2004), pp. 417–426 (2004)
36. Chomicki, J., Marcinkowski, J., Staworko, S.: Hippo: a system for computing consistent query answers to a class of SQL queries. In: Proc. of the 9th Int. Conf. on Extending Database Technology (EDBT 2004), pp. 841–844. Springer (2004)
37. De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: On reconciling data exchange, data integration, and peer data management. In: Proc. of the 26th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2007), pp. 133–142 (2007)
38. De Giacomo, G., Lenzerini, M., Poggi, A., Rosati, R.: On the update of description logic ontologies at the instance level. In: Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006), pp. 1271–1276 (2006)
39. De Giacomo, G., Lenzerini, M., Poggi, A., Rosati, R.: On the approximation of instance level update and erasure in description logics. In: Proc. of the 22nd Nat. Conf. on Artificial Intelligence (AAAI 2007), pp. 403–408 (2007)
40. Decker, S., Fensel, D., van Harmelen, F., Horrocks, I., Melnik, S., Klein, M., Broekstra, J.: Knowledge representation on the web. In: Proc. of the 2000 Description Logic Workshop

(DL 2000), *CEUR Electronic Workshop Proceedings,* `http://ceur-ws.org/`, vol. 33, pp. 89–97 (2000)

41. Donini, F.M., Lenzerini, M., Nardi, D., Nutt, W.: The complexity of concept languages. Information and Computation **134**, 1–58 (1997)
42. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A.: Deduction in concept languages: From subsumption to instance checking. J. of Logic and Computation **4**(4), 423–452 (1994)
43. Duschka, O.M., Genesereth, M.R., Levy, A.Y.: Recursive query plans for data integration. J. of Logic Programming **43**(1), 49–73 (2000)
44. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. Theoretical Computer Science **336**(1), 89–124 (2005)
45. Fagin, R., Kolaitis, P.G., Popa, L.: Data exchange: Getting to the core. ACM Trans. on Database Systems **30**(1), 174–210 (2005)
46. Fagin, R., Kolaitis, P.G., Popa, L., Tan, W.C.: Composing schema mappings: Second-order dependencies to the rescue. ACM Trans. on Database Systems **30**(4), 994–1055 (2005)
47. Fuxman, A., Fazli, E., Miller, R.J.: ConQuer: Efficient management of inconsistent databases. In: Proc. of the ACM SIGMOD Int. Conf. on Management of Data, pp. 155–166 (2005)
48. Fuxman, A., Kolaitis, P.G., Miller, R.J., Tan, W.C.: Peer data exchange. ACM Trans. on Database Systems **31**(4), 1454–1498 (2005)
49. Fuxman, A., Miller, R.J.: First-order query rewriting for inconsistent databases. J. of Computer and System Sciences **73**(4), 610–635 (2007)
50. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proc. of the 5th Logic Programming Symposium, pp. 1070–1080. The MIT Press (1988)
51. Genereseth, M.R., Keller, A.M., Duschka, O.M.: Infomaster: An information integration system. In: Proc. of the ACM SIGMOD Int. Conf. on Management of Data, pp. 539–542 (1997)
52. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic $\mathcal{SHIQ}$. J. of Artificial Intelligence Research **31**, 151–198 (2008)
53. Gottlob, G.: Computing cores for data exchange: New algorithms and practical solutions. In: Proc. of the 24rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2005), pp. 148–159 (2005)
54. Gottlob, G., Nash, A.: Data exchange: Computing cores in polynomial time. In: Proc. of the 25th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2006), pp. 40–49 (2006)
55. Grahne, G., Mendelzon, A.O.: Tableau techniques for querying information sources through global schemas. In: Proc. of the 7th Int. Conf. on Database Theory (ICDT'99), *Lecture Notes in Computer Science*, vol. 1540, pp. 332–347. Springer (1999)
56. Greco, G., Greco, S., Zumpano, E.: A logical framework for querying and repairing inconsistent databases. IEEE Trans. on Knowledge and Data Engineering **15**(6), 1389–1408 (2003)
57. Gribble, S., Halevy, A., Ives, Z., Rodrig, M., Suciu, D.: What can databases do for peer-to-peer? In: Proc. of the 4th Int. Workshop on the Web and Databases (WebDB 2001) (2001)
58. Grieco, L., Lembo, D., Ruzzi, M., Rosati, R.: Consistent query answering under key and exclusion dependencies: Algorithms and experiments. In: Proc. of the 14th Int. Conf. on Information and Knowledge Management (CIKM 2005), pp. 792–799 (2005)
59. Gryz, J.: Query rewriting using views in the presence of functional and inclusion dependencies. Information Systems **24**(7), 597–612 (1999)
60. Haarslev, V., Möller, R.: RACER system description. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001), *Lecture Notes in Artificial Intelligence*, vol. 2083, pp. 701–705. Springer (2001)
61. Halevy, A., Ives, Z., Suciu, D., Tatarinov, I.: Schema mediation in peer data management systems. In: Proc. of the 19th IEEE Int. Conf. on Data Engineering (ICDE 2003), pp. 505–516 (2003)
62. Halevy, A.Y.: Answering queries using views: A survey. Very Large Database J. **10**(4), 270–294 (2001)
63. Halevy, A.Y., Rajaraman, A., Ordille, J.: Data integration: The teenage years. In: Proc. of the 32nd Int. Conf. on Very Large Data Bases (VLDB 2006), pp. 9–16 (2006)

64. Horrocks, I.: Using an expressive description logic: FaCT or fiction? In: Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98), pp. 636–647 (1998)
65. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006), pp. 57–67 (2006)
66. Horrocks, I., Li, L., Turi, D., Bechhofer, S.: The Instance Store: DL reasoning with large numbers of individuals. In: Proc. of the 2004 Description Logic Workshop (DL 2004), *CEUR Electronic Workshop Proceedings,* `http://ceur-ws.org/`, vol. 104 (2004)
67. Hustadt, U., Motik, B., Sattler, U.: Data complexity of reasoning in very expressive description logics. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), pp. 466–471 (2005)
68. Kolaitis, P.G.: Schema mappings, data exchange, and metadata management. In: Proc. of the 24rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2005), pp. 61–75 (2005)
69. Krisnadhi, A., Lutz, C.: Data complexity in the $\mathcal{EL}$ family of description logics. In: Proc. of the 14th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2007), pp. 333–347 (2007)
70. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002), pp. 233–246 (2002)
71. Leone, N., Eiter, T., Faber, W., Fink, M., Gottlob, G., Greco, G., Kalka, E., Ianni, G., Lembo, D., Lenzerini, M., Lio, V., Nowicki, B., Rosati, R., Ruzzi, M., Staniszkis, W., Terracina, G.: The INFOMIX system for advanced integration of incomplete and inconsistent data. In: Proc. of the ACM SIGMOD Int. Conf. on Management of Data, pp. 915–917 (2005)
72. Levy, A.Y.: Logic-based techniques in data integration. In: J. Minker (ed.) Logic Based Artificial Intelligence. Kluwer Academic Publishers (2000)
73. Levy, A.Y., Mendelzon, A.O., Sagiv, Y., Srivastava, D.: Answering queries using views. In: Proc. of the 14th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'95), pp. 95–104 (1995)
74. Levy, A.Y., Rajaraman, A., Ordille, J.J.: Querying heterogenous information sources using source descriptions. In: Proc. of the 22nd Int. Conf. on Very Large Data Bases (VLDB'96) (1996)
75. Levy, A.Y., Srivastava, D., Kirk, T.: Data model and query evaluation in global information systems. J. of Intelligent Information Systems **5**, 121–143 (1995)
76. Libkin, L.: Data exchange and incomplete information. In: Proc. of the 25th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2006), pp. 60–69 (2006)
77. Möller, R., Haarslev, V.: Description logic systems. In: Baader et al. [8], chap. 8, pp. 282–305
78. Ortiz, M., Calvanese, D., Eiter, T.: Data complexity of query answering in expressive description logics via tableaux. J. of Automated Reasoning **41**(1), 61–98 (2008)
79. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics **X**, 133–173 (2008)
80. Pottinger, R., Levy, A.Y.: A scalable algorithm for answering queries using views. In: Proc. of the 26th Int. Conf. on Very Large Data Bases (VLDB 2000), pp. 484–495 (2000)
81. Schaerf, A.: On the complexity of the instance checking problem in concept languages with existential quantification. J. of Intelligent Information Systems **2**, 265–278 (1993)
82. Serafini, L., Ghidini, C.: Using wrapper agents to answer queries in distributed information systems. In: Proc. of the 1st Int. Conf. on Advances in Information Systems (ADVIS-2000), *Lecture Notes in Computer Science*, vol. 1909. Springer (2000)
83. Sirin, E., Parsia, B.: Pellet system description. In: Proc. of the 2006 Description Logic Workshop (DL 2006), *CEUR Electronic Workshop Proceedings,* `http://ceur-ws.org/`, vol. 189 (2006)
84. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: a practical OWL-DL reasoner. Tech. rep., University of Maryland Institute for Advanced Computer Studies (UMIACS) (2005)

85. Ullman, J.D.: Information integration using logical views. Theoretical Computer Science **239**(2), 189–210 (2000)

Index