AI Magazine Volume 26 Number 1 (2005) (© AAAI)

Data Integration

A Logic-Based Perspective

Diego Calvanese and Giuseppe De Giacomo

■ Data integration is the problem of combining data residing at different autonomous, heterogeneous sources and providing the client with a unified, reconciled global view of the data. We discuss data-integration systems, taking the abstract viewpoint that the global view is an ontology expressed in a class-based formalism. We resort to an expressive description logic, ALCQI, that fully captures class-based representation formalisms, and we show that query answering in data integration, as well as all other relevant reasoning tasks, is decidable. However, when we have to deal with large amounts of data, the high computational complexity in the size of the data makes the use of a full-fledged expressive description logic infeasible in practice. This leads us to consider DL-Lite, a specifically tailored restriction of ALCQI that ensures tractability of query answering in data integration while keeping enough expressive power to capture the most relevant features of class-based formalisms.

Data integration is the problem of combining data residing at different autonomous, heterogeneous sources and providing the client with a unified, reconciled view of these data. The typical architecture of a data-integration system in depicted in figures 1 and 2. In such a system, the actual data resides in a set of data sources. The user, however, does not access such data sources directly, but poses his or her queries to the integration system, and is thus freed from the necessity of knowing where the actual data reside and how to access the data sources to extract it. It is the task of the integration system to decide which sources are relevant for answering the user query, to distribute the query over such sources, to collect the returned answers, to combine and reconcile them, and to present the overall answer to the user. Two types of software modules called *wrappers* and *mediators* typically accomplish these tasks. *Wrappers* are responsible for directly accessing the sources and returning the data therein in a unified form (for example, as sets of tuples conforming to a relational schema). *Mediators* are responsible for combining the data coming from wrappers or other mediators and presenting them according to a specified structure (for example, a relational schema with certain attributes). The problem of setting up data-integration systems, and specifically wrappers and mediators, is becoming increasingly important, especially in enterprise applications, and is characterized by a number of issues that are interesting, both from a theoretical and from a practical point of view.

Most of the current work on data integration in databases (Hull 1997; Ullman 1997; Halevy 2001; and Lenzerini 2002) takes a declarative approach to the problem. This approach assumes that a data-integration system is characterized by giving explicitly to the client a global, virtual, reconciled, and unified view of the data. The virtual concepts are mapped to the concrete data sources, where the actual data resides, through explicit mapping assertions. Thus, the user formulates his or her queries in terms of the global view, and the system decides how to exploit the mappings in order to reformulate the user query in terms of the data sources. The abstract architecture corresponding to such an approach is depicted in figure 2. It maps to the concrete architecture in figure 1 by considering that the mediators implement the query reformulation process and the actual execution of the reformulated query. Also, in this abstract view, we do not deal with the issues related to wrapping the sources, and we assume that all sources are represented through their schema in a uniform data model, specifically, the relational model.

Different approaches for specifying map-

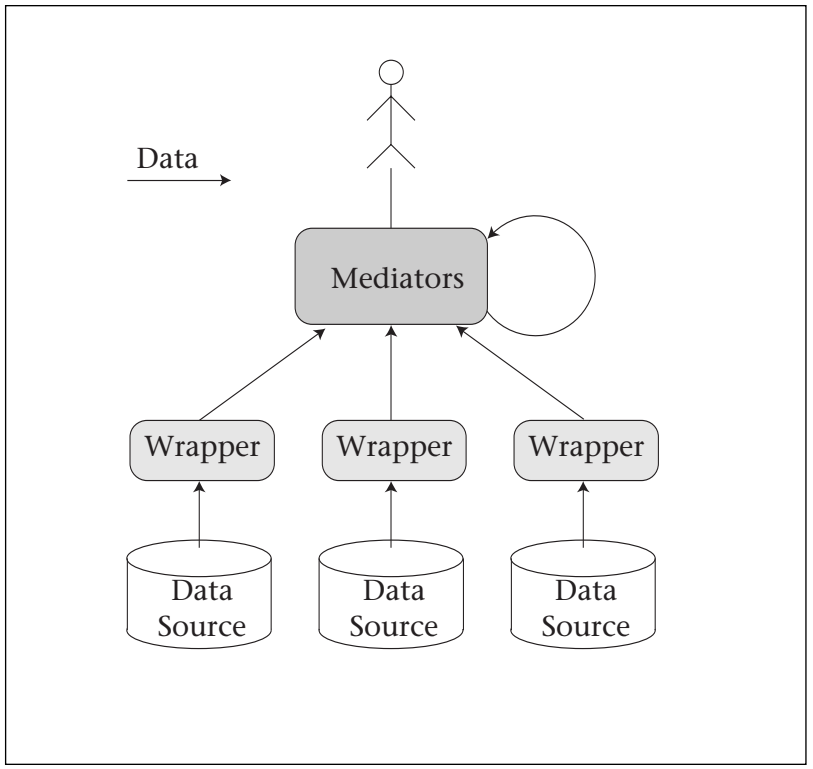

Figure 1. The Concrete Architecture of a Data-Integration System.

pings in a data-integration system have been proposed (Halevy 2001, Lenzerini 2002). In the *global-as-view* (GAV) approach, each concept of the global view is mapped to a query over the data sources. In other words, it is assumed that the data corresponding to a concept of the global view, which the user expects to obtain when she or he formulates her or his queries, can actually be retrieved from the data sources through a specific query, specified in a certain query language (such as select-project-join queries in structured query language [SQL]). In this way, query processing is conceptually easy, because it amounts to replacing (or *unfolding*) each global concept in the user query with the associated query over the sources and then executing the unfolded query over the sources.[1] However, the approach does not cope well with dynamicity and changes in the sources, since such changes potentially affect all mappings and require restructuring the global view. In contrast, in the *local-as-view* (LAV) approach, each concept in the data sources is defined in terms of a query over the global view. Thus, the information content of the sources is described in terms of the global view—in other words, in terms of those concepts that are familiar to the user and in terms of which the user accesses the system. This complicates query processing, since now the system is not told explicitly how to reformulate the concepts in the global view mentioned in the user query in terms of the data sources. On the other hand, changes in the sources require only changing the associated mappings and have no impact on the global view. A generalized approach, in which a mapping assertion relates a query over the global view to a query over the sources, called *GLAV* (global-local-as-view), has also been considered (Friedman, Levy, and Millstein 1999).

The loose coupling between data sources and global view by means of the mappings results in having incomplete information on the extensions of the concepts of the global view. In other words, if we fix the actual data at the source, there are in general many possible ways to get the extension of the concepts of the global view that are compatible with the data at the sources and with the mapping. Hence, when answering queries posed over the global view, such incompleteness must be taken into account. This results in an interest in computing the *certain* answers (Halevy 2001, Lenzerini 2002), or in other words, those answers that hold for all extensions of the global view that are compatible with the provided information.

More recently, the work on data integration in databases has started to consider also constraints expressed over the concepts of the global view, ranging from keys and foreign keys to more complex forms of assertions expressible in semantic data models, such as the entity-relationship model or unified modeling language (UML) class diagrams. Such constraints help to capture the complex interrelationships in the domain of interest better. However, they have a deep impact on how certain answers are computed, and hence they must be fully taken into account during query answering (Calì et al. 2001, 2002a, and 2004). We observe that, once we allow for constraints on the global view, the differences between the various approaches for establishing mappings become blurred, since, with the help of constraints, one can mimic one approach in the other (Calì et al. 2002b).

If we take an AI point of view, we can consider the whole integration system, constituted by the global view (with constraints), the data sources, and the mapping, as a knowledge base. In such a knowledge base, knowledge about specific data items (that is, extensional knowledge) and knowledge about how the information of interest is organized (that is, intensional knowledge) are clearly separated: extensional knowledge is constituted by the data sources, while intensional knowledge is formed by the global view and the mapping. Under this view, computing certain answers essentially corre-

sponds to logical inference: the certain answers are those data that are logically implied to be in answer to the query by the data present in the sources and the informaion on the global view and mapping.

Building on the above considerations, the ultimate realization of a global view is an ontology that clients can access and that gives them a semantically rich framework in which to understand the information gathered by the system. The mappings relate such ontology to the data sources used to retrieve the extensional information. In this article we take this perspective and consider an integration system as formed by a (global) ontology, a set of data sources, and mappings between the two. In particular, we discuss in detail ontologies that are expressed in terms of classes and relationships between classes. Such an approach stems from representation formalisms developed in various areas, ranging from entity-relationship diagrams in databases (Batini, Ceri, and Navathe 1992), UML class diagrams in software engineering,[2] and ontology languages for the semantic web, such as OWL-DL.[3]

Specifically, we start by introducing a general formal framework for describing information integration systems based on an ontology for the global view (in the "General Framework for Semantic Integration" section). Then, in the "Semantic Integration Using Description Logics" section, we look at systems whose ontology is expressed in terms of an expressive description logic, namely ALCQI (Baader et al 2003), which is a description logic that fully captures class-based representation formalism (Calvanese, Lenzerini, and Nardi 1999; Calì et al. 2002) and that is at the base of the current proposals for standard ontology languages. Notably, although we are using a full-fledged class-based language, all reasoning tasks, including computing certain answers in integration systems, are decidable. However, in information integration systems, since we typically deal with large amounts of data, it is crucial not only that reasoning tasks be decidable but that they also remain tractable in the size of data. Unfortunately, this is not the case for ALCQI nor for any representation formalism that aims at fully capturing class-based modeling. This leads us to consider, in the section "Why DL-Lite is a 'Rich' DL," a specifically tailored restriction of ALCQI that we call *DL-Lite,* which, on the one hand, provides enough expressive power to capture the most relevant features of class-based formalisms and, on the other hand, ensures tractability with respect to the size of the data. In our conclusion, we discuss further research directions.

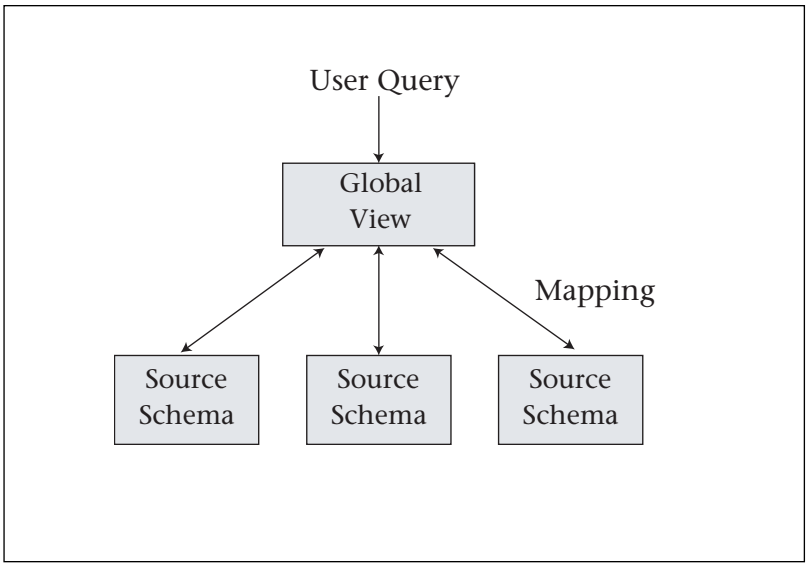

Figure 2. The Abstract Architecture of a Data-Integration System.

General Framework for Semantic Integration

In this section we present a general formal framework for semantic-integration systems. Following the standard approach in information integration, we will refer to integration systems whose components are the following: (1) a set of data sources, containing the actual information users are interested in; (2) a global ontology, which provides a reconciled, integrated, and virtual view of the underlying sources in terms of which users access the system; and (3) the mapping between the two, which is used to relate the information in the sources to the concepts in the global ontology.

In what follows, one of the main aspects is the definition of the semantics of both the integration system and of queries posed to the global ontology. To keep things simple, we will use in the following a unique semantic domain Δ, constituted by a fixed, infinite set of symbols. We also assume a fixed set of constants, and we fix the interpretation of such constants so that (1) each constant denotes an element in Δ; (2) different constants denote different elements of Δ; and (3) each element in Δ is denoted by a constant.[4] In the following, with some abuse of notation, we will not distinguish between constants and the domain elements they denote.

Formally, an ontology-based integration system (OIS) $\mathcal{M}$ is a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{G}$ is the global ontology, $\mathcal{S}$ is the set of data sources, and $\mathcal{M}$ is the mapping between $\mathcal{G}$ and $\mathcal{S}$.

We assume that the global ontology $\mathcal{G}$ of an OIS is expressed as a theory (named simply $\mathcal{G}$) in

some logic (for example, first-order logic) involving a set of predicates interpreted over Δ.

We assume to have a set $\mathcal{S}$ of n data sources S_1, ..., S_n, each one consisting of extensions of predicates over Δ. We assume that the (predicate) alphabets of the various data sources are mutually disjoint and that each one is disjoint from the alphabet of the global ontology. For simplicity of exposition, without loss of generality, we assume that each source is constituted by the extension of a single predicate.

The mapping $\mathcal{M}$ is the heart of the OIS, in that it specifies how the predicates in the global ontology $\mathcal{G}$ and in the data sources $\mathcal{S}$ are mapped to each other. In particular, such mappings are established by relating open formulas (that is, queries) over the global ontology to open formulas over the data sources.

Notice that we have assumed that data sources are seen as databases. In turn, such sources may be complex ontologies, thus containing dependencies and interrelationships among their various concepts at the intensional level. However, we consider a setting in which such data sources are completely autonomous and hence may not conform to the global ontology that the clients of an OIS can access. Neither do we want to integrate their intensional knowledge into the global ontology seen by the client. We want just to take into account how the data at the sources is used to feed the predicate extension of the global ontology. Intuitively, in specifying the semantics of an OIS, we have to start with an extension of the data sources, called the *source database,* and the crucial point is to determine which are the models of the global ontology that correspond to such a source database. In doing so, both the constraints specified in the global ontology and the mapping are taken into account. More precisely, the semantics of an OIS is defined as the set of all models of the global ontology that satisfy the mapping with respect to the source database. What it means to satisfy a mapping depends on the form of the mapping and is discussed in the section "OIS Based on ALCQI" (see also Calvanese, De Giacomo, and Lenzerini 2002).

Queries posed to an OIS $\mathbb{O}$ are expressed in terms of a certain query language over the alphabet of the global ontology and are intended to extract a set of tuples of elements of the semantic domain Δ. In accordance with what is typical in databases, we require that each query have an associated arity and that it extract only tuples of that arity. Given a source database for $\mathbb{O}$, the tuples we are interested in are those that are guaranteed to be in the answer of the query for every model for $\mathbb{O}$ with respect to the source database. In other words, we are interested in *certain* answers.

One of the most common ways to express knowledge on a domain of interest is to resort to class-based formalisms, in which knowledge is represented in terms of objects grouped into classes and relationships between classes. Examples are entity-relationship diagrams in databases, UML class diagrams in software engineering, and ontology languages for the semantic web such as OWL-DL. All such formalisms can be captured in a fragment of first-order logic in which one can express inclusions and equivalences between classes and possibly pose additional constraints on the relations between classes. Such fragments correspond to a class of logics called *description logics* (Baader et al. 2003).

On the other hand, for the mapping, which represents the heart of an OIS, it is in general not sufficient to limit the expressive power to direct correspondences between classes, since this does not allow one to capture the complex interrelations that may exist between the data in the sources and the (virtual) data in the global view. In a real-world setting, one needs a much more powerful mechanism for establishing mappings between the sources $\mathcal{S}$ and the global view $\mathcal{G}$. Specifically, one would like, on the one hand, to acquire the relevant information to be extracted from $\mathcal{S}$ by navigating and aggregating several concepts and, on the other hand, to characterize these data in terms of the elements of $\mathcal{G}$ as precisely as possible. To achieve this, it is necessary to resort to mappings that relate to each other a *query* Q_s over $\mathcal{S}$ and a *query* Q_g over $\mathcal{G}$, both expressed in an appropriate query language. As is common in data integration, we assume the mappings to be sound, that is, the data extracted from the sources through Q_s are in general only a subset of those satisfying the corresponding query Q_g in the global models for $\mathbb{O}$ with respect to a source database.

Semantic Integration Using Description Logics

The considerations made in the previous section lead us to provide a formalization of an OIS, which is based on the use of description logics to represent ontologies (Calvanese et al. 1998a, 1998b). Description logics (DLs) (Baader et al. 2003) are knowledge representation formalisms that are able to capture the core features of virtually all class-based representation formalisms used in AI, software engineering, and databases (Calvanese, Lenzerini, and Nardi 1998, 1999). Recently, DLs have gained an increased popularity as the formalisms that provide the theoretical foundation for the lan-

guages becoming standard for the semantic web, specifically OWL-DL. One of the distinguishing features of these logics is that they are equipped with optimal reasoning algorithms, and practical systems implementing such algorithms are now available (Horrocks 1998, Haarslev and Möller 2001, Möller and Haarslev 2003).

In the following section, we first introduce a specific DL and then illustrate how such logic is used to define an OIS.

The Description Logic ALCQI

In DLs, the domain of interest is modeled by means of *concepts* and *roles*. Concepts are unary predicates, which denote classes of objects called *instances of the concept.* Roles instead are binary predicates, which denote binary relationships between objects. The simplest forms of concepts and roles are atomic concepts and roles, which are constituted just by a name. Each DL is then equipped with a set of specific constructs that allow one to obtain, starting from atomic concepts and roles, complex concept expressions (or simply concepts). Each construct has a precise set-theoretic semantics, and therefore the meaning of complex concepts is determined on the basis of the meaning of their constituents and the constructs combining them. Similarly, a DL may be equipped with constructs for obtaining complex role expressions (or simply roles).

We focus our attention on a specific DL, ALCQI, which belongs to the well-studied family of AL languages (Baader et al. 2003). ALCQI is a notable example of an expressive DL that features constructs that are typical of conceptual modeling formalisms and that in fact allow ALCQI to capture the most important features of such formalisms (Berardi, Calvanese, and De Giacomo 2001; Berardi et al. 2003). Here, we do not provide a formal presentation of ALCQI; instead we introduce its constructs by means of examples. Also, instead of the abstract notation typical of the DL literature (compare with Baader et al. 2003), we make use of a more verbose, textual notation that is easier for readers not familiar with the DL syntax to understand.

The ALCQI DL provides concept constructs for complement, intersection, union, existential restriction, universal quantification, and number restrictions. As for roles, it provides the construct for inverse roles. Recall that roles denote binary relations between objects; in the following we say that an object o_1 is connected to another object o_2 through a role *R,* meaning that the pair (o_1, o_2) is in the relation represented by *R*. We now discuss the various constructs in more detail. *Complement, intersection,* and *union* denote simply the corresponding set operations on the sets of instances of the involved concepts. Existential restriction and universal quantification represent restricted forms of existential and universal quantification, respectively. More precisely, through existential restriction on a role *R,* one can denote all those objects connected through *R* to at least one instance of a concept *C*. For example, *(Staff and (teaches some Course))* denotes those individuals that are staff members and that teach some course. The dual construct, *universal quantification* on a role *R,* denotes objects that are connected through *R* only to instances of a concept *C*. For example, *(teaches only UGCourse)* denotes those individuals that teach only undergraduate courses. Also, through number restrictions on a role *R,* one can express restrictions on the minimum and maximum number of connections that an object may have through *R* to instances of a concept *C*. Thus, number restrictions represent a generalization of existential, functionality, and multiplicity constraints in data models. For example, *(teaches at-most 3 Course)* denotes those individuals that teach at most three courses. Finally, through an inverse role *(inverse R)* one can denote the inverse of the relationship denoted by a role *R*. For example, *(Course and ((inverse teaches) some Postdoc)* denotes all those courses that are taught by a postgraduate. This is done by referring to the role *teaches,* whose inverse is the taught-by relation.

In ALCQI, a knowledge base is constituted by two components, a *TBox,* used to express intensional knowledge, and an *ABox*, used to express extensional knowledge. Specifically, a TBox is constituted by a set of *inclusion assertions*, each of the form $(C_1$ *is-a* $C_2)$, where C_1 and C_2 are two arbitrary ALCQI concepts. Such an inclusion assertion states a subclass-superclass relationship in which C_1 is the subclass and C_2 is the superclass. For example, *((Staff and (teaches some Course)) is-a Busy)* expresses that each staff member teaching a course is busy. There is no restriction on the set of assertions that may constitute a TBox, and, in particular, they may involve cycles.

The ABox of an ALCQI knowledge base is constituted by a set of membership assertions involving concepts or roles, of the form $C(z)$ and $R(z_1, z_2)$, stating respectively that the object z is an instance of the concept *C* and that the pair of objects (z_1, z_2) is an instance of the role *R*. For example, *Staff(ann), Course(ai), teaches(ann,ai)* express respectively that *ann* is a staff member, that *ai* is a course, and that *ann* teaches *ai.*

Being logics, DLs in general and ALCQI in

particular are equipped with a formal semantics and with reasoning services defined in accordance with the semantics. The basic reasoning services over DL knowledge bases are (1) knowledge base satisfiability, that is, determining whether a knowledge base can be populated without violating any of the inclusion or membership assertions; (2) concept satisfiability with respect to a knowledge base, that is, determining whether it is possible to populate a knowledge base in such a way that a given concept is populated with at least one instance; and (3) logical implication, that is, determining whether a given TBox or ABox assertion necessarily holds whenever all assertions in a given knowledge base hold.

Finally, we introduce the notion of query in ALCQI. Remember that the answer to a query, when the query is evaluated over a knowledge base, is a set of tuples of objects. The types of queries we consider are conjunctive queries, which correspond to SQL select-project-join queries but have a notation that is more convenient for formal manipulations. A *conjunctive query* over an ALCQI knowledge base is a conjunction of atoms in which each atom involves a predicate applied to a variable or a constant. Each predicate is either an atomic concept (hence, a unary predicate) or an atomic role (hence, a binary predicate), which may also freely be used in the assertions of the knowledge base. When evaluating the query, the constants denote specific domain objects, while the variables are instantiated on the domain objects, in accordance with the predicates in which they appear. For example, the variable x in the atom *UGCourse(x)* may be instantiated only on undergraduate courses. Each variable may be either free or existentially quantified. The free variables (also called *distinguished variables*) denote the components of the tuples that are in the answer to the query. Existentially quantified variables, instead, are used to relate to each other the various atoms in the query, but they do not directly contribute to the answer to the query. For example, the conjunctive query $\{x, y \mid \text{Staff}(x) \wedge \text{Staff}(y) \wedge \text{teaches}(x, z) \wedge \text{teaches}(y, z) \wedge \text{UGCourse}(z)\}$ denotes all pairs of staff members that have at least one undergraduate course they teach in common. The distinguished variables are x and y, while z is an existentially quantified variable that stands for the commonly taught undergraduate course (notice that the existential quantifier on z is not explicitly present in the query, but is implicit in its semantics).

The basic reasoning services that are of interest in the presence of queries are query answering and query containment. Query answering consists in determining all tuples of objects that are in the answer to the query, whenever all assertions of the knowledge base are satisfied. Observe that, as a special case of query answering, we have concept satisfiability and logical implication of ABox assertions. Query containment consists in determining, given a knowledge base and two queries of the same arity, whether the answer to one query is contained in the answer to the other one whenever all assertions of the knowledge base are satisfied. As a special case of query containment, we have logical implication of inclusion assertions involving atomic concepts on both sides. In fact, it can also be shown that query containment can be reformulated as query answering (Abiteboul and Duschka 1998).

ALCQI is equipped with effective reasoning techniques that are sound and complete with respect to the semantics. In particular, all reasoning tasks involving a knowledge base only (and not queries) are EXPTIME-complete. Checking query containment, and hence also query answering, are instead EXPTIME-hard and solvable in 2EXPTIME in the size of the knowledge base (Calvanese, De Giacomo, and Lenzerini 1998). Note that such an exponential bound depends also on the size of the data (that is, the ABox).

OIS Based on ALCQI

We now set up a framework for ontology integration, which extends ideas developed for data integration over DL knowledge bases (Calvanese et al. 1998a; Calvanese, De Giacomo, and Lenzerini 2000). In particular, we describe the main components of the ontology integration system, and we provide the semantics both of the system and of query answering.

In this setting, an OIS $\mathcal{O} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ is defined as follows:

The global ontology $\mathcal{G}$ is an ALCQI knowledge base constituted only by a TBox. In accordance with what we discussed earlier, this means that only intensional knowledge (that is, describing how the information is organized) and no extensional knowledge (that is, about specific data items) can be maintained by such a global ontology.

Each data source in $\mathcal{S}$ is constituted simply by a relational alphabet and by the extensions of the relations in such an alphabet. For example, such extensions may be expressed as relational databases. Observe that we are assuming that no intensional relation between terms is present in the local ontologies.

The mapping $\mathcal{M}$ between $\mathcal{G}$ and $\mathcal{S}$ is given by a set of correspondences of the form $Q_s \rightsquigarrow Q_g$, where Q_s is a conjunctive query over one of the data sources in $\mathcal{S}$, and Q_g is a conjunctive query

over the global ontology $\mathcal{G}$. As mentioned, the mapping correspondences are assumed to be sound. This means that the correspondence $Q_s \rightarrow Q_g$ is satisfied whenever the data extracted from the sources through Q_s is a subset of (not necessarily equal to) the global data satisfying Q_g.

The form of mapping we have considered here is quite general and represents a generalization of the types of mappings that have been considered in the literature on data integration (Hull 1997, Halevy 2001, Lenzerini 2002). Indeed, two basic approaches for defining such a mapping have been proposed: (1) the local-as-view (LAV) approach, in which each relation of the data sources in $\mathcal{S}$ is mapped to a query over the global ontology $\mathcal{G}$; and (2) the global-as-view (GAV) approach, in which each concept of the global ontology G is mapped to a query over the data sources in $\mathcal{S}$.

The GAV approach has been traditionally considered simpler, since, in order to answer a query over the global ontology, it is sufficient to unfold all concepts referenced in the query with their definition in terms of the data sources specified in the mapping. However, in the presence of intensional constraints in the global ontology, this is in general not sufficient any more, and query answering becomes more involved (Calì et al. 2001, 2002a, 2004).

Many authors point out that, despite its difficulty, the LAV approach better supports a dynamic environment, where data sources can be added to the system without the need for restructuring the global ontology. Hence, recent research work on data integration has followed this approach (Ullman 1997; Halevy 2001; Levy, Srivastava, and Kirk 1995; Calvanese et al. 1998a; Calvanese et al. 2000). The major challenge in this case is that, to answer a query expressed over the global ontology, one must be able to reformulate the query in terms of queries to the sources. While in the GAV approach such a reformulation is guided by the correspondences in the mapping, in LAV the problem requires a reasoning step, so as to infer how to use the sources for answering the query.

The type of mapping we have considered here—GLAV (Friedman, Levy, and Millstein 1999)—combines the flexibility of the LAV and GAV approaches by allowing one to establish directly mappings between two queries. We will see later that, also in our setting, the added expressive power provided by GLAV mappings does not add complexity to the techniques that have already been adopted to handle LAV mappings.

Query answering in this setting requires quite sophisticated techniques. Indeed, in order to answer a query posed over the global ontology with the data contained in the local ontologies, one has to take into account the knowledge both in the global ontology and in the mapping. Such query-answering techniques are studied by Calvanese, De Giacomo, and Lenzerini (2000) for the case of LAV and are essentially based on encoding the data extracted through the mappings into an ABox. The techniques can be applied to GLAV mappings as well, by observing that a GLAV mapping $Q_s \rightarrow Q_g$ can be rephrased by introducing a new source relation R of the same arity as the queries Q_s and Q_g in the mapping. The extension of the new source R is given by evaluating Q_s on the data sources, and R is then mapped to the global ontology through LAV mapping $\{x_1,\ldots,x_n \mid R(x_1,\ldots,x_n)\} \rightarrow Q_g$.

Consider for example the OIS $\mathcal{O}_d = \langle \mathcal{G}_d, \mathcal{S}_d, \mathcal{M}_d \rangle$ defined in figure 3.

Simplifying Reasoning Tasks

One of the most important lines of research in DLs is concerned with the trade-off between expressive power and computational complexity of sound and complete reasoning. Research on this topic has shown that DLs with efficient, that is, worst-case polynomial time, reasoning algorithms lack the modeling power required in capturing conceptual models and basic ontology languages, while DLs with sufficient modeling power, such as ALCQI, suffer from inherently worst-case exponential time behavior of reasoning (Calvanese, Lenzerini, and Nardi 1998, 1999; Borgida and Brachman 2003). This is reflected also when addressing ontology-based integration, in which the inherently high computational complexity of the underlying DL has a negative effect on the computational complexity of query answering, and makes it infeasible in practice.

In this section we introduce a DL called DL-Lite (Calvanese, De Giacomo, Lenzerini, and Rosati 2004; Calvanese, De Giacomo, Lembo, Lenzerini, and Rosati 2004) to be used as the formalism underlying an ontology-based integration system. Such a DL provides a very good trade-off between expressive power and complexity of reasoning, both over a knowledge base and over queries. On the one hand it has sufficient expressive power, being specifically tailored to capture the fundamental aspects of conceptual data models (such as entity-relationship diagrams) (Batini, Ceri, and Navathe 1992), object-oriented formalisms (such as basic UML class diagrams), and basic ontology languages, such as OWL-DL. On the other hand, it admits advanced forms of sound and complete reasoning, which take into account

The global ontology $\mathcal{G}_d$ is the ALCQI knowledge base

$$\begin{array}{rll} \textit{Postdoc} & \textit{is-a} & \textit{Staff} \\ \textit{UGCourse} & \textit{is-a} & \textit{Course} \\ \textit{Staff and (teaches some Course)} & \textit{is-a} & \textit{Busy} \end{array}$$

expressing that each postdoc is a staff member, that each undergraduate course is a course, and that each staff member who teaches a course is busy.

The set $\mathcal{S}_d$ of data sources consists of two data sources, containing respectively the unary relations $\mathbf{T}_1$ and $\mathbf{T}_2$.

The mapping $\mathcal{M}_d$ is

$$\{x \mid \mathbf{T}_1(x)\} \rightarrow \{x \mid \textit{Staff}(x) \wedge \textit{teaches}(x,y) \wedge \textit{UGCourse}(y)\}$$
$$\{x \mid \mathbf{T}_1(x)\} \rightarrow \{x \mid \textit{Postdoc}(x)\}$$

that is, it is constituted by two LAV correspondences, associating to each source relation a conjunctive query over the global ontology. The first correspondence expresses that source relation $\mathbf{T}_1$ contains a set of staff members teaching an undergraduate course. Notice that, since the mapping is sound, there may be also other staff members teaching an undergraduate course besides those listed in $\mathbf{T}_1$. Similarly, the second correspondence expresses that source relation $\mathbf{T}_2$ contains postgraduate students.

Consider that the conjunctive query $\mathbf{Q}_w = \{x \mid \textit{Postdoc}(x) \wedge \textit{Busy}(x)\}$ over $\mathcal{G}_d$ is asking for the postdocs who are busy. Given the source database $\mathcal{D}$, in which $\mathbf{T}_1$ has extension {*ann, bill*} and $\mathbf{T}_2$ has extension {*ann, dan*}, we have that the certain answer is constituted by the single tuple *ann*. Observe that, to obtain such an answer, we have to reason using the mappings and the inclusions in the global ontology.

Figure 3. Example 1.

both a knowledge base (constituted by a TBox and an ABox), and queries, and which are polynomial time in the size of the knowledge base, including the data.

DL-Lite

DL-Lite is a DL that is quite simple from the language point of view. The constructs it provides are complement and intersection of concepts (but no union), simplified forms of existential restriction and number restrictions, and inverse roles (recall that roles denote binary relations). Moreover, the concept constructs may not be combined freely but need to respect certain syntactic conditions. Namely, starting again from atomic concepts and atomic roles, we define basic concepts as either an atomic concept or an unqualified existential restriction (Baader and Nutt 2003). Such a construct denotes all objects that are connected through a role *R* to some other object *o*. In an existential restriction in ALCQI, we specify the concept that this object o must be an instance of. In DL-Lite, we just say that such an object exists but do not further qualify it. For example, the concept *(teaches something)* denotes those objects that teach something without further qualifying what is taught. General concepts in DL-Lite are then conjunctions of basic concepts and their complements. Note that, in DL-Lite, the use of complement is restricted to basic concepts only and that we cannot express union of concepts. As for roles, similar to ALCQI, DL-Lite has a construct for inverse roles.

Using this simple language, in a DL-Lite knowledge base we allow the making of assertions of specific forms only. Specifically, in a DL-Lite TBox, we allow for inclusion assertions of the form *(B is-a C)*, for which on the left-hand side we must have a basic concept while on the right-hand side we may have an arbitrary DL-Lite concept. Observe that, as in ALCQI, we do allow for cyclic assertions. Indeed, we can enforce the cyclic propagation through the role P of the property of belonging to concept *A* using the two DL-Lite inclusion assertions *(A is-a (P something))* and *(((inverse P) something) is-a A)*. The first assertion states that all objects in *A* are connected through role *P* to some object. The second assertion states that all objects to which role *P* connects are in *A*. Hence, if we start by considering an object o_1 in *A*, then o_1 must be connected through *P* to some object o_2, which itself must be in *A*, and hence connected through *P* to some object o_3, and so on. This could go on forever, or we could close the "chain" by connecting a certain o_i to one of the previous objects, for example, o_1.

Also, in addition to the above inclusion assertions, in DL-Lite we have a specific form of inclusion assertions that make use of number restrictions. Such assertions, called functionality assertions, have the form *(functional P)* and *(functional (inverse P))* and express, respectively, the functionality of an atomic role *P* and of the inverse *(inverse P)* of an atomic role. Functionality of *P* means that each object may be connected through *P* to at most one object; similarly for functionality of *(inverse P)*. Note that, in contrast to ALCQI, in DL-Lite functionality of a certain role can be expressed only as a global property and not locally, that is, for the instances of a certain concept. Thus, we are not allowed to assert, for instance, that postgraduates teach at most one course, while professors can teach an arbitrary number of courses. This would require stating that *teaches* is functional for the instances of *Postdoc* while it is not so for the instances of *Professor,* and this is not possible in DL-Lite .

Finally, the *ABox* of a DL-Lite knowledge base has the same form as that of an ALCQI

knowledge base and is thus constituted by a set of membership assertions involving concepts and roles. Recall that the former have the form $C(z)$, where C is a concept and z is an individual, while the latter have the form $R(z_1,z_2)$, where R is a role and z_1, z_2 are individuals.

We consider queries in DL-Lite that have the same form as those in ALCQI and hence are conjunctive queries over a (DL-Lite) knowledge base.

Why DL-Lite is a "Rich" DL

Although equipped with advanced reasoning services, at first sight DL-Lite seems to be rather weak in modeling intensional knowledge and hence of limited use in practice. In fact this is not the case. Despite the simplicity of its language and the specific form of inclusion assertions allowed, DL-Lite is able to capture the main notions (though not all, obviously) of conceptual modeling formalism used in databases and software engineering, such as entity-relationship and UML class diagrams. In particular, DL-Lite assertions allow us to specify the following important constructs (relying on database terminology, we use the terms *class* and *relation* to denote respectively an atomic concept and an atomic role):

ISA, or subclass-superclass relations, using assertions of the form *(A_1 is-a A_2),* stating that the class A_1 is a subclass of the class A_2. For example, *(UGCourse is-a Course)* states that each undergraduate course is a course.

Class disjointness, using assertions of the form (A_1 *is-a not* A_2), stating disjointness between the two classes A_1 and A_2. For example, *(Course is-a not Staff)* states that courses and staff members are disjoint.

Role typing, using assertions of the form *((P something) is-a A_1) (resp., (((inverse P) something) is-a A_2)),* stating that the first (resp., second) component of the relation P is of type A_1 (resp., A_2).[5] Notice that these kinds of assertions correspond to domain *(resp., range)* assertions. For example, *((teaches something) is-a Staff)* types the domain of *teaches* to be a staff member, while *(((inverse teaches) something) is-a Course)* types the range of *teaches* to be a course.

Participation constraints, using assertions of the form *(A is-a (P something)) (resp., (A is-a ((inverse P) something)))*, stating that instances of class A participate to the relation P as the first *(resp., second)* component. For example, *(Postdoc is-a (teaches something))* states that each postdoc has to teach something, while *(UGCourse is-a ((inverse teaches) something))* states that undergraduate courses need to be taught by someone.

Nonparticipation constraints, using assertions of the form *(A is-a not (P something)) (resp., (A is-a not ((inverse P) something))),* stating that instances of class A do not participate to the relation P as the first *(resp., second)* component. For example, *(Student is-a not (teaches something))* states that a student cannot teach anything.

Functionality restrictions, using assertions of the form *(functional P) (resp., (functional (inverse P))),* stating that an object can be the first *(resp., second)* component of the relation P at most once. For example, *(functional (inverse teaches))* states that a course may be taught by at most one individual.

We reexpress example 1 (shown in figure 3) in DL-Lite. The OIS $\mathcal{O}_d = \langle \mathcal{G}_d, \mathcal{S}_d, \mathcal{M}_d \rangle$ is defined as follows:

The global ontology $\mathcal{G}_d$ is the DL-Lite knowledge base

Postdoc	*is-a*	*Staff*
UGCourse	*is-a*	*Course*
(teaches something)	*is-a*	*Staff*
((inverse teaches) something)	*is-a*	*Course*
(teaches something)	*is-a*	*Busy*

As in figure 3, we have that each postdoc is a staff member and that each undergraduate course is a course. Here we also have that teaching is always performed by a staff member and involves a course. Moreover, who teaches is busy. Observe that the DL-Lite typing assertions on *teaches,* together with the last assertion, imply the ALCQI assertion *(Staff and (teaches some Course) is-a Busy)* of figure 3.

The set $\mathcal{S}_d$ of data sources consists of the same two data sources.

The mapping $\mathcal{M}_d$ is

$$\{x \mid \mathbf{T}_1(x)\} \rightarrow \{x \mid teaches(x,y) \wedge UGCourse(y)\}$$
$$\{x \mid \mathbf{T}_2(x)\} \rightarrow \{x \mid Postdoc(x)\}$$

Note that, with respect to figure 3, we have removed the *Staff(x)* atom from the first assertion, since it is implied by the ontology.

Considering again the conjunctive query $\mathbf{Q}_w = \{ x \mid Busy(x) \}$ over $\mathcal{G}_d$, and the same source database, we get the same answers as in figure 3.

Figure 4. A Reexpression of Figure 3 in DL-Lite.

Notably two important modeling features of class-based formalisms, which can be captured by ALCQI, are missing in DL-Lite: (1) the ability of stating covering constraints, that is, stating that each instance of a class must be an instance of (at least) one of its subclasses; and (2) the ability of stating subset constraints between relations. Note that these features are present in full-fledged entity-relationship diagrams and UML class diagrams. They are missing in DL-Lite exactly to get the nice computational characteristics that we are after. Instead, observe that the limitation to binary roles only is not crucial. Indeed, it is possible to extend the rea-

soning techniques discussed next to *n*-ary relations without losing most nice computational properties.

Reasoning

The techniques that have been developed for reasoning in DL-Lite are quite different from those of traditional DLs since they are based on a series of results developed in databases for query containment and query answering under constraints (Johnson and Klug 1984; Calì, Lembo, and Rosati 2003a, 2003b). Indeed, differently from more complex DLs, all reasoning tasks in DL-Lite, both those involving the knowledge base and those involving queries, can be done in polynomial time in the size of the knowledge base.

Hence, by resorting to DL-Lite instead of ALCQI as the formalism for representing the global ontology of an OIS and exploiting the results presented in the subsection "OIS Based on ALCQI," we obtain that all tasks related to query answering in an OIS, in particular computing certain answers to queries, can be done in polynomial time in the size of the knowledge base, including the data, and in exponential time in the size of the query. Interestingly, this continues to hold even if we consider DL-Lite extended with relations of arbitrary arity.

On the other hand, the results reported in Calvanese, De Giacomo, Lembo, Lenzerini, and Rosati (2004) imply that the introduction of inclusion assertions on roles (that is, role inclusion assertions) makes the polynomial techniques at the base of reasoning in DL-Lite inapplicable.

Conclusions

We have discussed information integration under a logical perspective in which the global view is seen as an ontology expressed in class-based formalisms. Data sources have been considered simply as systems that provide data but make no further contribution to the query-answering process.

The next step is to consider data sources as ontology-based systems themselves, equipped with both intensional and extensional information and with query-answering capabilities. This leads us to a form of information integration that is based on autonomous peers that collaborate in making available to clients the information distributed in the system. This form of information integration is referred to as *peer to peer* (Bernstein et al. 2002; Halevy et al. 2003; Franconi et al. 2003). Its formalization typically requires going one step further and making a distinction between what is part of the extension and what is *known* to be part of the extension (Calvanese, De Giacomo, Lenzerini, and Rosati 2004; Calvanese, De Giacomo, Lembo, Lenzerini, and Rosati 2004).

Acknowledgments

We would like to thank Natasha Noy for her careful reading of the manuscript and her very useful comments, which helped to improve the readability of this article substantially.

Notes

1. Of course, system-level problems, such as how to distribute the query over the sources, how to collect and combine the answers, and so on, still remain to be addressed, but we are not concerned with these aspects here.

2. www.omg.org/uml/

3. www.w3.org/2001/sw/WebOnt/

4. That is, such constants act as standard names (Levesque and Lakemeyer 2001).

5. Observe that this has nothing to do with the qualified restrictions (P some A) (resp., (P only A)), which are not used to type the role P but are used to select those objects that are the first component of P and that are related (through P) to some object (resp., only to objects) belonging to A.

References

Abiteboul, S.; and Duschka, O. 1998. Complexity of Answering Queries Using Materialized Views. In Proceedings of the Seventeenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (Pods '98), 254–265. New York: Association for Computing Machinery.

Baader, F.; Calvanese, D. Mcguinness, D.; Nardi, D.; and Patel-Schneider, P. F. 2003. eds. *The Description Logic Handbook: Theory, Implementation and Applications*. New York: Cambridge University Press.

Baader, F.; and Nutt, W. 2003. Basic Description Logics. In chapter 2, *The Description Logic Handbook: Theory, Implementation, and Applications,* ed. F. Baader, D. Calvanese, D. Mcguinness, D. Nardi, and P. F. Patel-Schneider, 43–95. New York: Cambridge University Press.

Batini, C.; Ceri, S.; and Navathe, S. B. 1992. *Conceptual Database Design, An Entity-Relationship Approach*. Menlo Park, CA: Benjamin Cummings Publishing.

Berardi, D.; Calì, A.; Calvanese, D.; and De Giacomo, G. 2003. Reasoning on UML Class Diagrams. Technical Report 11-03, Dipartimento Di Informatica E Sistemistica, Università Di Roma "La Sapienza," Rome, Italy.

Berardi, D.; Calvanese, D.; and De Giacomo, G. 2001. Reasoning on UML Class Diagrams Using Description Logic Based Systems. Paper presented at the KI-2001 Workshop on Applications of Description Logics. In Ceur Electronic Workshop Proceedings (Ceur-ws.org/vol-44/), Technical University of Aachen, Aachen, Germany..

Bernstein, P. A.; Giunchiglia, F.; Kementsietsidis, A.;

Mylopoulos, J.; Serafini, I. and Zaihrayeu, I. 2002. Data Management for Peer-to-Peer Computing: A Vision. Paper presented at the Fifth International Workshop on the Web and Databases (Webdb 2002), Madison, WI, June 6–7.

Borgida A.; and Brachman, R. J. 2003. Conceptual Modeling with Description Logics. In chapter 10, *The Description Logic Handbook: Theory, Implementation, and Applications,* ed. F. Baader, D. Calvanese, D. Mcguinness, D. Nardi, and P. F. Patel-Schneider, 349–372. New York: Cambridge University Press.

Calì, A.; Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 2001. Accessing Data Integration Systems through Conceptual Schemas. In *Proceedings of the Twentieth International Conference on Conceptual Modeling (ER 2001),* volume 2503, Lecture Notes in Computer Science, 270–284. Berlin: Springer-Verlag

Calì, A.; Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 2002a. Data Integration under Integrity Constraints. In *Proceedings of the Fourteenth International Conference on Advanced Information Systems Engineering (CAISE 2002),* volume 2348, Lecture Notes in Computer Science, 262–279. Berlin: Springer-Verlag.

Calì, A.; Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 2002b. A Formal Framework for Reasoning on UML Class Diagrams. In *Proceedings of the Thirteenth International Symposium on Methodologies for Intelligent Systems (ISMIS 2002),* volume 2366, Lecture Notes in Computer Science, 503–513. Berlin: Springer-Verlag.

Calì, A.; Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 2002c. On the Expressive Power of Data Integration Systems. In *Proceedings of the Twenty-First International Conference on Conceptual Modeling (ER 2002),* volume 2503, Lecture Notes in Computer Science. Berlin: Springer-Verlag.

Calì, A.; Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 2004. Data Integration under Integrity Constraints. *Information Systems* 29(3): 147–163.

Calì, A.; Lembo, D.; and Rosati, R. 2003a. On the Decidability and Complexity of Query Answering over Inconsistent and Incomplete Databases. In Proceedings of the Twenty-Second ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2003), 260–271. New York: Association for Computing Machinery.

Calì, A.; Lembo, D.; and Rosati, R. 2003b. Query Rewriting and Answering under Constraints in Data Integration Systems. In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003), 16–21. San Francisco: Morgan Kaufmann Publishers.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2004. What to Ask to a Peer: Ontology-Based Query Reformulation. In *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR 2004),* 469–478. Menlo Park, CA: AAAI Press.

Calvanese, D.; De Giacomo, G.; Lenzerini, M. 2002. A Framework for Ontology Integration. In *The Emerging Semantic Web—Selected Papers from the First Semantic Web Working Symposium,* ed. I. Cruz, S. Decker, J. Euzenat, and D. Mcguinness, 201–214. Amsterdam, The Netherlands: IOS Press.

Calvanese, D.; De Giacomo, G.; Lenzerini, M. 2000. Answering Queries Using Views over Description Logics Knowledge Bases. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence* (AAAI 2000), 386–391. Menlo Park, CA: AAAI Press.

Calvanese, D.; De Giacomo, G.; Lenzerini, M. 1998. On the Decidability of Query Containment under Constraints. In Proceedings of the Seventeenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS '98), 149–158. New York: Association for Computing Machinery.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; Nardi, D.; and Rosati, R. 1998a. Description Logic Framework for Information Integration. In Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), 2–13. San Francisco: Morgan Kaufmann Publishers.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; Nardi, D.; and Rosati, R. 1998b. Information Integration: Conceptual Modeling and Reasoning Support. In Proceedings of the Sixth International Conference on Cooperative Information Systems (Coopis '98), 80–291. Los Alamitos, CA: IEEE Computer Society.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Rosati, R. 2004. Logical Foundations of Peer-to-Peer Data Integration. In Proceedings of the Twenty-Third ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS2004), 241–251. New York: Association for Computing Machinery.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; Rosati, R.; and Vetere, G. 2004. DL-Lite: Practical Reasoning for Rich DLs. Paper presented at the 2004 Description Logic Workshop (DL 2004). In Ceur Electronic Workshop Proceedings, (Ceur-ws.org/vol-104), Technical University of Aachen, Aachen, Germany.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Vardi, M. Y. 2000. View-Based Query Processing and Constraint Satisfaction. In Proceedings of the Fifteenth IEEE Symposium on Logic in Computer Science (LICS 2000), 361–371. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Calvanese, D.; Lenzerini, M.; and Nardi, D. 1999. Unifying Class-Based Representation Formalisms. *Journal of Artificial Intelligence Research* 11: 199–240.

Calvanese, D.; Lenzerini, M.; and Nardi, D. 1998. Description Logics for Conceptual Data Modeling. In *Logics for Databases and Information Systems,* ed. J. Chomicki and G. Saake, 229–264. Dortrecht, The Netherlands: Kluwer Academic Publishers.

Franconi, E.; Kuper, G.; Lopatenko, A.; and Serafini, L. 2003. A Robust Logical and Computational Characterisation of Peer-To-Peer Database Systems. In *Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2003),* Lecture Notes in Computer Science volume 2944. Berlin: Springer-Verlag.

Friedman, M.; Levy, A.; and Millstein, T. 1999. Navigational Plans for Data Integration. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (AAAI '99), 67–73. Menlo Park, CA: AAAI Press/ MIT Press.

Haarslev, V. and R. Möller, R. 2001. Racer System Description. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2001),* vol-

ume 2083 of Lecture Notes in Artificial Intelligence, 701–705. Berlin: Springer-Verlag.

Halevy, A. Y. 2001. Answering Queries Using Views: A Survey. *Journal of Very Large Database*s 10(4): 270–294.

Halevy, A. Y.; Ives, Z.; Suciu, D.; and Tatarinov, I. 2003. Schema Mediation in Peer Data Management Systems. In Proceedings of the Nineteenth IEEE International Conference on Data Engineering (ICDE 2003), 505–516. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Horrocks, I. 1998. The Fact System. In *Proceedings of the Second International Conference on Analytic Tableaux and Related Methods (Tableaux '98),* volume 1397, Lecture Notes in Artificial Intelligence, ed. H. De Swart, 307–312. Berlin: Springer-Verlag.

Hull, R. 1997. Managing Semantic Heterogeneity in Databases: A Theoretical Perspective. In Proceedings of the Twenty-First ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 1997), 51–61. New York: Association for Computing Machinery.

Johnson, D. S.; and Klug, A. C. 1984. Testing Containment of Conjunctive Queries under Functional and Inclusion Dependencies. *Journal of Computer and System Sciences* 28(1): 167–189.

Lenzerini, M. 2002. Data Integration: A Theoretical Perspective. In Proceedings of the Twenty-First ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002), 233–246. New York: Association for Computing Machinery.

Levesque, H. J.; and Lakemeyer, G. 2001. *The Logic of Knowledge Bases*. Cambridge, MA: MIT Press.

Levy, A. Y.; Srivastava, D.; and Kirk, T. 1995. Data Model and Query Evaluation in Global Information Systems. *Journal of Intelligent Information Systems* 5(2): 121–143.

Möller, R.; and Haarslev, V. 2003. Description Logic Systems. Chapter 8 in *The Description Logic Handbook: Theory, Implementation, and Applications,* ed. F. Baader, D. Calvanese, D. Mcguinness, D. Nardi, and P. F. Patel-Schneider, 282–305. New York: Cambridge University Press.

Ullman, J. D. 1997, Information Integration Using Logical Views. In *Proceedings of the Sixth International Conference on Database Theory (ICDT '97),* volume 1186 of Lecture Notes in Computer Science, 19–40. Berlin: Springer-Verlag.

Diego Calvanese (calvanese@inf.unibz.it) graduated in electrical engineering in 1990 and earned his Ph.D. in computer science in 1995, both at the Università di Roma "La Sapienza." Currently, he is associate professor in computer science at the Free University of Bolzano/Bozen, where he is a member of a research group on databases and artificial intelligence. His main research interests are in data modeling, information integration, semistructured data, e-services, and logics for knowledge representation and reasoning. He is the author of more than 100 refereed publications in international conferences and journals.

Giuseppe De Giacomo (degiacomo@dis.uniroma1.it) earned his Ph.D. in computer engineering in 1994 and is currently an associate professor at the Università di Roma "La Sapienza," Dipartimento di Informatica e Sistemistica, where he has conducted research for more than ten years in the fields of knowledge representation and reasoning in databases, data integration, semantics interoperability, including service and process synthesis, and reasoning on dynamic systems. He is the author of more than 100 publications in international journals and conferences in the areas of artificial intelligence, databases, information systems, and cognitive robotics.