Description Logic Framework for Information Integration

Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, Riccardo Rosati
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
{calvanese,degiacomo,lenzerini,nardi,rosati}@dis.uniroma1.it

Abstract

Information Integration is one of the core problems in distributed databases, cooperative information systems, and data warehousing, which are key areas in the software development industry. Two critical factors for the design and maintenance of applications requiring Information Integration are conceptual modeling of the domain, and reasoning support over the conceptual representation. We demonstrate that Knowledge Representation and Reasoning techniques can play an important role for both of these factors, by proposing a Description Logic based framework for Information Integration. We show that the development of successful Information Integration solutions requires not only to resort to very expressive Description Logics, but also to significantly extend them. We present a novel approach to conceptual modeling for Information Integration, which allows for suitably modeling the global concepts of the application, the individual information sources, and the constraints among different sources. Moreover, we devise inference procedures for the fundamental reasoning services, namely relation and concept subsumption, and query containment. Finally, we present a methodological framework for Information Integration, which can be applied in several contexts, and highlights the role of reasoning services within the design process.

1 INTRODUCTION

In recent years there has been a growing interest in Information Integration, whose goal is to access, relate and combine data from multiple sources. Indeed, Information Integration is one of the core problems in distributed databases, cooperative information systems, and data warehousing, which are key areas in the software development industry (Wiederhold, 1996; Knoblock & Levy, 1995; Widom, 1995; Hull, 1997).

Early work on integration was carried out in the context of database design, and focused on the so-called *schema integration* problem, i.e. designing a global, unified schema for a database application starting from several subschemata, each one produced independently from the others (Batini, Lenzerini, & Navathe, 1986). More recent efforts have been devoted to *data integration*, which generalizes schema integration by taking into account actual data in the integration process. Here the input is a collection of source data sets (each one constituted by a schema and actual data), and the goal is to provide an integrated and reconciled view of the data residing at the sources, without interfering with their autonomy (Ullman, 1997). We only deal with the so-called read-only integration, which means that such a reconciled view is used for answering queries, and not for updating information.

Data integration can be either *virtual* or *materialized*. In the first case, the integration system acts as an interface between the user and the sources (Sheth & Larson, 1991; Hurson, Bright, & Pakzad, 1994), and is typical of multidatabases, distributed databases, and more generally open systems. In virtual integration query answering is generally costly, because it requires accessing the sources. In the second case, the system maintains a replicated view of the data at the sources (Gupta & Mumick, 1995; Inmon, 1996), and is typical, for example, both in information system re-engineering and data warehousing. In materialized data integration, query answering is generally more efficient, because it does not require acessing the sources, whereas maintaining the materialized views is costly, especially when the views must be up-to-date with respect to the updates at the sources (view refreshment). In the rest of this paper, we do not deal with the problem of view refreshment.

There are two basic approaches to the data integration problem, called *procedural* and *declarative*. In the procedural approach, data are integrated in an ad-hoc manner with respect to a set of predefined information needs. In this case, the basic issue is to design suitable software modules that access the sources in order to fulfill the predefined information requirements. Several data integration (both virtual and materialized) projects, such as TSIMMIS (Chawathe, Garcia-Molina, Hammer, Ireland, Papakonstantinou, Ullman & Widom, 1994; Ullman, 1997), *Squirrel* (Zhou, Hull, & King, 1996; Hull & Zhou, 1996), and WHIPS (Hammer, Garcia-Molina, Widom, Labio, & Zhuge, 1995; Wiener, Gupta, Labio, Zhuge, Garcia-Molina, & Widom, 1996) follow this idea. They do not require an explicit notion of integrated data schema, and rely on two kinds of software components: *wrappers* that encapsulate sources, converting the underlying data objects to a common data model, and *mediators* (Wiederhold, 1992) that obtain information from one or more wrappers or other mediators, refine this information by integrating and resolving conflicts among the pieces of information from the different sources, and provide the resulting information either to the user or to other mediators. The basic idea is to have one mediator for every query pattern required by the user, and generally there is no constraint on the consistency of the results of different mediators.

In the declarative approach, the goal is to model the data at the sources by means of a suitable language, to construct a unified representation, to refer to such a representation when querying the global information system, and to derive the query answers by means of suitable mechanisms accessing the sources and/or the materialized views. This is the idea underlying systems such as *Carnot* (Collet, Huhns, & Shen, 1991; Huhns, Jacobs, Ksiezyk, Shen, Singh, & Cannata, 1993), SIMS (Arens, Chee, Hsu, & Knoblock, 1993; Arens, Knoblock, & Chen, 1996) and *Information Manifold* (Levy, Srivastava, & Kirk, 1995; Kirk, Levy, Sagiv, & Srivastava, 1995; Levy, Rajaraman, & Ordille, 1996). The declarative approach provides a crucial advantage over the procedural one: although building a unified representation may be costly, it allows maintaining a consistent global view of the information sources, which represents a reusable component of the Information Integration systems.

We adopt a declarative approach to integration, and argue that two critical factors for the design and maintenance of applications requiring Information Integration are the *conceptual modeling of the domain*, and the possibility of *reasoning over the conceptual representation.* We demonstrate that Knowledge Representation and Reasoning techniques can play an important role for both of these factors, by proposing a Description Logic (Borgida, 1995; Donini, Lenzerini, Nardi, & Schaerf, 1996) based framework for Information Integration. In particular, our work provides the following main contributions:

(1) We use Description Logics for the *conceptual modeling* of both the global domain and the various sources. Since the development of successful Information Integration solutions requires specific modeling features, we propose a new Description Logic, which treats *n*-ary relations as first-class citizens. Note that the usual characteristic of many Description Logics to model only unary predicates (concepts) and binary predicates (roles) would represent an intolerable limit in our case.

(2) We provide suitable mechanisms for expressing what we call the *intermodel assertions*, i.e. interrelationships between concepts in different sources. Thus, integration is seen as the incremental process of understanding and representing the relationships between data in the sources, rather than simply producing a unified data schema. The fact that our approach is incremental is also important in amortizing the cost of integration.

(3) For an accurate description of the information sources, we incorporate in our logic the possibility of describing the data at the sources in terms of a set of *relational structures*. Each relational structure is defined as a view over the conceptual representation, thus providing a formal mapping between the description of data and the conceptual representation of the domain.

(4) Our representation framework is equipped with *inference procedures* for the fundamental reasoning services, namely concept and relation subsumption, and query containment. Indeed, we make use of the first decidability result on query containment for a Description Logic with *n*-ary relations (Calvanese, De Giacomo, & Lenzerini, 1998). Based on these reasoning methods, we present a methodological framework for Information Integration, which can be applied both in the virtual and in the materialized approach.

In comparing our framework with other declarative approaches, we observe that in both Carnot and SIMS, reasoning is based on formalisms, Cyc (Lenat

& Guha, 1990) and LOOM (MacGregor, 1991) respectively, that are undecidable. Information Manifold uses the Classic (Patel-Schneider, McGuiness, Brachman, Resnick, & Borgida, 1991) Description Logic at the conceptual level, and extends it with conjunctive queries at the logical level. While this Description Logic is polynomially decidable, it cannot fully capture neither n-ary relationships among the various classes of data in the domain, nor the intermodel assertions, nor many interesting inferences on such assertions.

Compared with the procedural approaches, which have been designed to cope in a more flexible way with the heterogeinity and the dynamics of the sources, our methodology for incremental schema integration based on intermodel assertions combines the advantages of a conceptual representation with the necessary flexibility to deal with changes in the domain. In particular, the ability of reasoning over both the conceptual representation and the relational structures can be profitably used in designing mediators with verifiable specifications.

The paper is organized as follows. In Section 2 we describe in more detail our framework for Information Integration based on Description Logics. In Section 3 we present the particular Description Logic we use in the framework. In Section 4 we illustrate how the reasoning techniques associated with our logic are used to improve the design and maintenance of the Information Integration system. Finally, Section 5 concludes the paper.

2 THE FRAMEWORK

In our approach to Information Integration, we refer to the architecture depicted in Figure 1, in which three layers can be identified:

- a *conceptual layer* called the Domain Model, which is constituted by an Enterprise Model and one Source Model for each data source;
- a *logical layer*[1], constituted by the *Source Schemas* and the *Materialized View Schema*, which describe the logical content of source data stores and of materialized view store, respectively;
- a *physical layer*, which consists of the data stores containing the actual data of the sources and the integrated materialized views.

[1]Here the term "logical" is used according to the database terminology, where it denotes a description of data in terms of structures managed by DBMSs (e.g., relational tables), which are at a more abstract level with respect to the physical organization of data.

The methodology for Information Integration described in Section 4, and the reasoning techniques illustrated in Section 3, support the incremental building of the conceptual and the logical representations. The designer is provided with information on various aspects, including the global concepts relevant for new information requirements, the sources from which a new view can be defined, the correspondences between sources and/or views, and a trace of the integration steps.

We describe now the structure of the conceptual and logical layers, which constitute the core of the proposed integration framework. The actual formalisms we adopt, and the associated reasoning techniques are described in the next section.

2.1 THE CONCEPTUAL LEVEL

The *Enterprise Model* is a conceptual representation of the global concepts and relationships that are of interest to the application. It corresponds roughly to the notion of integrated conceptual schema in the traditional approaches to schema integration. However, since we propose an incremental approach to integration, the Enterprise Model is not necessarily a complete representation of all the data of the sources but it provides a consolidated and reconciled description of the concepts and the relationships that are important to the enterprise, and have already been analyzed. Such a description is subject to changes and additions as the analysis of the information sources proceeds. The *Source Model* of an information source is a conceptual representation of the data residing in it, or at least of the portion of data currently taken into account. Again, our approach does not require a source to be fully analyzed and conceptualized.

Both the Enterprise Model and the Source Models are expressed by means of a logic-based formalism (see Section 3) which is general and powerful enough to express the usual database models, such as the Entity-Relationship Model, the Relational Model, or the Object-Oriented Data Model (for the static part). The inference techniques associated with the formalism allow for carrying out several reasoning services on the representation.

Besides the Enterprise Model and the various Source Models, the *Domain Model* contains the specification of the interdependencies between elements of different Source Models and between Source Models and the Enterprise Model. The notion of interdependency is a central one in our approach. Since the sources are of interest in the overall architecture, integration does not simply mean producing the Enterprise Model, but

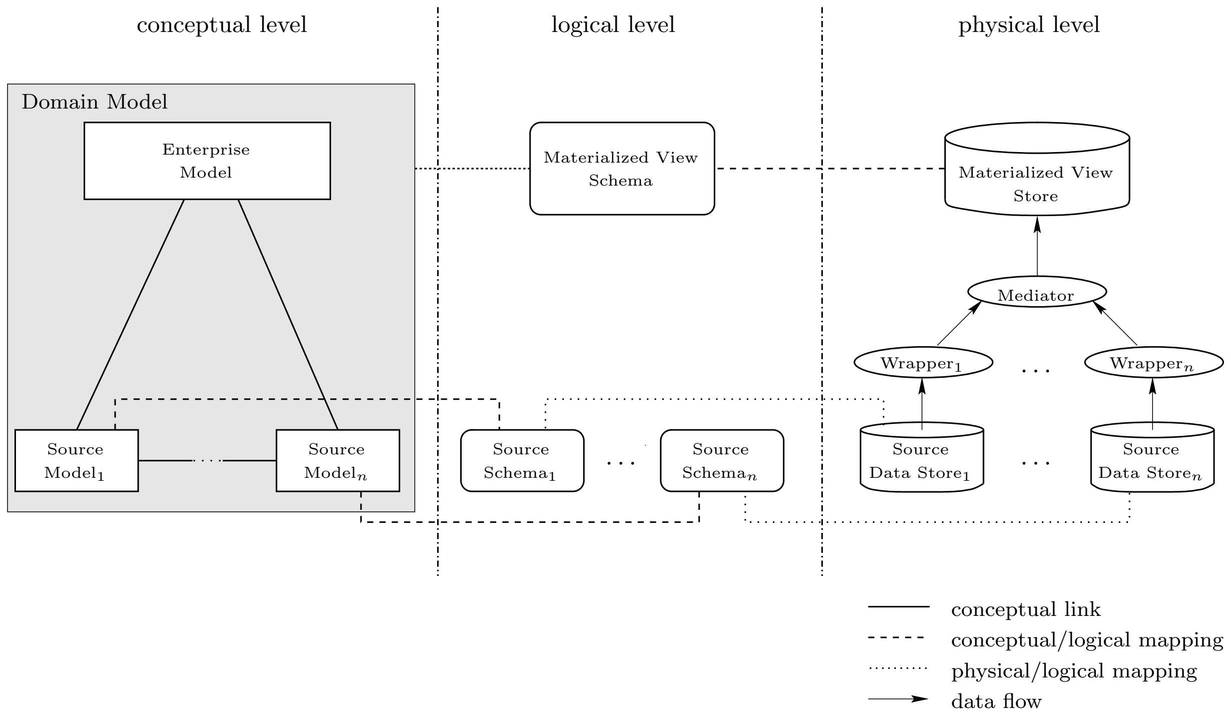

Figure 1: Architecture for Data Integration

rather to be able to establish the correct relationships both between the Source Models and the Enterprise Model, and between the various Source Models. We formalize the notion of interdependency by means of so called *intermodel assertions* (Catarci & Lenzerini, 1993), which provide a simple and effective declarative mechanism to express the dependencies that hold between entities (i.e. classes and relationships) in different models (Hull, 1997). We use again a logic-based formalism to express intermodel assertions, and the associated inference techniques provide a means to reason about interdependencies among models.

2.2 THE LOGICAL LEVEL

Our approach requires that each source, besides being conceptualized, is also described in the *Source Schema* in terms of a logical data model (in our case the Relational Model) which allows for representing the structure of the stored data. Such a structure is specified in terms of a set of relation definitions, each one expressed by means of a view (i.e. a query) over the conceptual representation of the source (i.e. the Source Model). Suitable software components, called *wrappers*, implement the mapping of physical structures to logical structures (see Figure 1). More precisey, a wrapper is able to access a source and transform the data therein into a form that is coherent with the logical specification of the source.

In the case where the integrated data (or portions thereof) are materialized, the *Materialized View Schema* provides a description of the logical content of the materialized views constituting the *Materialized View Store*. Similarly to the case of the sources, each portion of the Materialized Views Schema is described in terms of a set of definitions of relations, each one expressed in terms of a query over the Domain Model. A view is actually materialized starting from the data produced by wrappers by means of suitable software components, called *mediators* (see Figure 1). Again, a discussion on mediators is outside the scope of the present paper. In the case where a virtual approach is adopted there are no Materialized Views, and the data are provided by the mediators at query processing time.

A more detailed discussion on wrappers and mediators is outside the scope of this paper.

3 REPRESENTATION AND REASONING

In this section we present the formalism that we use for describing data both at the conceptual and the logical level, and we illustrate the basis of the reasoning techniques associated with the formalism.

3.1 REPRESENTATION AT THE CONCEPTUAL LEVEL

We use for the conceptual level a specific *Description Logic*, called $\mathcal{DLR}$, which includes *concepts* and *n-ary relations*[2]. $\mathcal{DLR}$ is inspired by the languages introduced in (Calvanese, De Giacomo, & Lenzerini, 1995; De Giacomo & Lenzerini, 1995, 1994; Catarci & Lenzerini, 1993), and is a natural extension of Description Logics (Donini et al., 1996; Calvanese, Lenzerini, & Nardi, 1994; Borgida, 1995) towards n-ary relations, which are extremely important in our context.

We assume to deal with a finite set of *atomic relations* and *concepts*, denoted by $\mathbf{P}$ and A respectively. We use $\mathbf{R}$ to denote arbitrary *relations* (of given arity between 2 and n_{max}), and C to denote arbitrary *concepts*. Concepts and relations are built according to the following syntax, where i and j denote components of relations, i.e. integers between 1 and n_{max}, n denotes the arity of a relation, i.e. an integer between 2 and n_{max}, and k denotes a nonnegative integer:

$$\begin{array}{lcl}\mathbf{R} & ::= & \top_n \mid \mathbf{P} \mid (\$i/n\colon C) \mid \neg\mathbf{R} \mid \mathbf{R}_1 \sqcap \mathbf{R}_2 \\ C & ::= & \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists[\$i]\mathbf{R} \mid (\leq k\, [\$i]\mathbf{R})\end{array}$$

Concepts and relations must be *well-typed*, which means that (i) only relations of the same arity n can be combined to form expressions of type $\mathbf{R}_1 \sqcap \mathbf{R}_2$ (which inherit the arity n), and (ii) $i \leq n$ whenever i denotes a component of a relation of arity n.

The semantics of the $\mathcal{DLR}$ constructs is specified through the usual notion of interpretation. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is constituted by an *interpretation domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and to each relation $\mathbf{R}$ of arity n a subset $\mathbf{R}^{\mathcal{I}}$ of $(\Delta^{\mathcal{I}})^n$, such that the conditions in Figure 2 are satisfied. We observe that $\top_1$ denotes the interpretation domain, while $\top_n$, for $n > 1$, does *not* denote the n-cartesian product of the domain, but only a subset of it, that covers all relations of arity n. As a consequence, the "$\neg$" construct on relations is used to express difference of relations, rather than complement.

A $\mathcal{DLR}$ *conceptual model* $\mathcal{M}$ (i.e., either the Enterprise Model or one of the Source Models) is constituted by a finite set of *intramodel assertions*, which express knowledge on the relations and concepts in $\mathcal{M}$, and have the form

$$L \sqsubseteq L' \qquad L \not\sqsubseteq L' \qquad L \equiv L' \qquad L \not\equiv L'$$

[2]Domains, i.e. sets of values such as integer, string, etc., can be easily included in $\mathcal{DLR}$.

with L, L' either two relations of the same arity or two concepts.

An interpretation $\mathcal{I}$ *satisfies* an intramodel assertion $L \sqsubseteq L'$ (resp. $L \equiv L'$) if $L^{\mathcal{I}} \subseteq L'^{\mathcal{I}}$ (resp. $L^{\mathcal{I}} = L'^{\mathcal{I}}$), and it satisfies $L \not\sqsubseteq L'$ (resp. $L \not\equiv L'$) if $\mathcal{I}$ does not satisfy $L \sqsubseteq L'$ (resp. $L \equiv L'$). An interpretation *satisfies* $\mathcal{M}$, if it satisfies all assertions in $\mathcal{M}$.

To specify knowledge on the conceptual interrelationships among the sources and/or the enterprise, we use *intermodel assertions* (Catarci & Lenzerini, 1993), which have essentially the form of intramodel assertions, although the two relations (concepts) L and L' belong to two different conceptual models $\mathcal{M}_i$, $\mathcal{M}_j$. Intermodel assertions can be either *extensional*, which express relationships between the extensions of the relations (concepts) involved, or *intensional*, which express conceptual relationships that are not necessarily reflected at the instance level. Formally, an *intermodel assertion* over two conceptual models $\mathcal{M}_i$, $\mathcal{M}_j$ $(i \neq j)$ is an assertion of one of the following forms

$$\begin{array}{lll@{\qquad\qquad}lll} L & \sqsubseteq_{ext} & L' & L & \not\sqsubseteq_{ext} & L' \\ L & \equiv_{ext} & L' & L & \not\equiv_{ext} & L' \\ \\ L & \sqsubseteq_{int} & L' & L & \not\sqsubseteq_{int} & L' \\ L & \equiv_{int} & L' & L & \not\equiv_{int} & L' \end{array}$$

in which L and L' are either two relations with compatible signatures or two concepts belonging to $\mathcal{M}_i$ and $\mathcal{M}_j$ respectively.

An interpretation $\mathcal{I}$ *satisfies* an extensional intermodel assertion $L \sqsubseteq_{ext} L'$ (resp. $L \equiv_{ext} L'$) if $L^{\mathcal{I}} \subseteq L'^{\mathcal{I}}$ (resp. $L^{\mathcal{I}} = L'^{\mathcal{I}}$), and it satisfies $L \not\sqsubseteq_{ext} L'$ (resp. $L \not\equiv_{ext} L'$) if $\mathcal{I}$ does not satisfy $L \sqsubseteq_{ext} L'$ (resp. $L \equiv_{ext} L'$). Hence, interpretation of extensional intermodel assertions is analogous to the one of intramodel assertions.

Instead, intensional intermodel assertions are interpreted by first taking the intersection of the relations (concepts) L, L' with both $\top_{ni}$ and $\top_{nj}$ ($\top_{1i}$ and $\top_{1j}$). Formally:

1. Let C, C' be concepts belonging respectively to $\mathcal{M}_i, \mathcal{M}_j$. Then, an interpretation $\mathcal{I}$ satisfies the intensional intermodel assertion $C \sqsubseteq_{int} C'$ (resp. $C \equiv_{int} C'$) if

$$C^{\mathcal{I}} \cap \top_{1i} \cap \top_{1j} \subseteq C'^{\mathcal{I}} \cap \top_{1i} \cap \top_{1j}$$

(resp. $C^{\mathcal{I}} \cap \top_{1i} \cap \top_{1j} = C'^{\mathcal{I}} \cap \top_{1i} \cap \top_{1j}$). Moreover, $\mathcal{I}$ satisfies $C \not\sqsubseteq_{int} C'$ (resp. $C \not\equiv_{int} C'$) if $\mathcal{I}$ does not satisfy $C \sqsubseteq_{int} C'$ (resp. $C \equiv_{int} C'$).

$$
\begin{array}{rcl}
\top_n^{\mathcal{I}} & \subseteq & (\Delta^{\mathcal{I}})^n \\
\mathbf{P}^{\mathcal{I}} & \subseteq & \top_n^{\mathcal{I}} \\
(\neg \mathbf{R})^{\mathcal{I}} & = & \top_n^{\mathcal{I}} \setminus \mathbf{R}^{\mathcal{I}} \\
(\mathbf{R}_1 \sqcap \mathbf{R}_2)^{\mathcal{I}} & = & \mathbf{R}_1^{\mathcal{I}} \cap \mathbf{R}_2^{\mathcal{I}} \\
(\$i/n\colon C)^{\mathcal{I}} & = & \{(d_1,\ldots,d_n) \in \top_n^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\} \\
\\
\top_1^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
A^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} & = & C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\exists[\$i]\mathbf{R})^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \exists (d_1,\ldots,d_n) \in \mathbf{R}^{\mathcal{I}}.\, d_i = d\} \\
(\leq k\,[\$i]\mathbf{R})^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid |\{(d_1,\ldots,d_n) \in \mathbf{R}_1^{\mathcal{I}} \mid d_i = d\}| \leq k\}
\end{array}
$$

Figure 2: Semantic rules for $\mathcal{DLR}$ ($\mathbf{P}$, $\mathbf{R}$, $\mathbf{R}_1$, and $\mathbf{R}_2$ have arity n)

2. Let $\mathbf{R}, \mathbf{R}'$ be relations (of the same arity n) belonging respectively to $\mathcal{M}_i, \mathcal{M}_j$. Then, an interpretation $\mathcal{I}$ satisfies the intensional intermodel assertion $\mathbf{R} \sqsubseteq_{int} \mathbf{R}'$ (resp. $\mathbf{R} \equiv_{int} \mathbf{R}'$) if

$$\mathbf{R}^{\mathcal{I}} \cap \top_{ni} \cap \top_{nj} \subseteq \mathbf{R}'^{\mathcal{I}} \cap \top_{ni} \cap \top_{nj}$$

(resp. $\mathbf{R}^{\mathcal{I}} \cap \top_{ni} \cap \top_{nj} = \mathbf{R}'^{\mathcal{I}} \cap \top_{ni} \cap \top_{nj}$). Moreover, $\mathcal{I}$ satisfies $\mathbf{R} \not\sqsubseteq_{int} \mathbf{R}'$ (resp. $\mathbf{R} \not\equiv_{int} \mathbf{R}'$) if $\mathcal{I}$ does not satisfy $\mathbf{R} \sqsubseteq_{int} \mathbf{R}'$ (resp. $\mathbf{R} \equiv_{int} \mathbf{R}'$).

A *Domain Model (DM)* $\mathcal{W}$ is an $(m+2)$-tuple $\langle \mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_m, \mathcal{G}\rangle$ such that: (i) $\mathcal{M}_0$ is the Enterprise Model; (ii) each $\mathcal{M}_i$, for $i \in \{1,\ldots,m\}$, is a Source Model; (iii) $\mathcal{G}$ (for "glue") is a finite set of intermodel assertions. We assume that $\mathcal{G}$ always includes for each $i \in \{1,\ldots,m\}$ the following assertions: $\top_{1i} \sqsubseteq_{ext} \top_{10}$, and $\top_{ni} \sqsubseteq_{ext} \top_{n0}$ for each n such that a relation $\mathbf{R}$ of arity n appears in $\mathcal{M}_i$. An interpretation $\mathcal{I}$ *satisfies* $\mathcal{W}$ if it satisfies all the intramodel and intermodel assertions in $\mathcal{W}$.

3.2 REPRESENTATION AT THE LOGICAL LEVEL

We express the logical level in terms of a set of relation schemas, each describing either a relation of a Source Schema, or a relation of the Materialized View Schema. Such relations are connected to the DM by characterizing each relation schema in terms of a non-recursive Datalog query over the elements of the DM, i.e. a query of the form:

$$q(\vec{\mathbf{x}}) \;\leftarrow\; \mathit{body}_1(\vec{\mathbf{x}},\vec{\mathbf{y}}_1) \vee \cdots \vee \mathit{body}_m(\vec{\mathbf{x}},\vec{\mathbf{y}}_m)$$

where each $\mathit{body}_i(\vec{\mathbf{x}},\vec{\mathbf{y}}_i)$ is a conjunction of *atoms*, either $\mathbf{R}(\vec{\mathbf{t}})$ or $C(t)$ (where $\vec{\mathbf{t}}$ and t are variables in $\vec{\mathbf{x}}$,$\vec{\mathbf{y}}_i$)[3], with $\mathbf{R}$, C relations and concepts over the DM. The *arity* of q is equal to the number of variables of $\vec{\mathbf{x}}$.

[3]Our approach is applicable also when constants are used in the queries.

We observe that the atoms in the queries are arbitrary $\mathcal{DLR}$ relations and concepts, freely used in the assertions of the schema. This distinguishes our approach with respect to (Donini, Lenzerini, Nardi, & Schaerf, 1991, 1998; Levy & Rousset, 1996), where no constraints can be expressed in the schema on the relations that appear in the queries.

Given an interpretation $\mathcal{I}$ of a DM $\mathcal{W}$, a query q for $\mathcal{W}$ of arity n is interpreted as the set $q^{\mathcal{I}}$ of n-tuples $(o_1,\ldots,o_n)$, with each $o_i \in \Delta^{\mathcal{I}}$, such that, when substituting $(o_1,\ldots,o_n)$ for $(x_1,\ldots,x_n)$, the formula

$$\exists \vec{\mathbf{y}}_1 . \mathit{body}_1(\vec{\mathbf{x}},\vec{\mathbf{y}}_1) \vee \cdots \vee \exists \vec{\mathbf{y}}_m . \mathit{body}_m(\vec{\mathbf{x}},\vec{\mathbf{y}}_m)$$

evaluates to true in $\mathcal{I}$. If q and q' are two queries (of the same arity) for $\mathcal{W}$, we say that q is *contained in* q' wrt $\mathcal{W}$, if $q^{\mathcal{I}} \subseteq q'^{\mathcal{I}}$ for every $\mathcal{I}$ satisfying $\mathcal{W}$.

3.3 REASONING

The typical kinds of reasoning services needed at the conceptual level in order to support the designer in applying the integration methodology presented in Section 4 (e.g., checking whether the DM is consistent, checking whether a relation or a concept is satisfiable in the DM, checking subsumption between relations or concepts in the DM) can be reduced to checking satisfiability of the DM. The reasoning tasks can in particular be exploited for computing and incrementally maintaining the concept and relation lattice of the DM, or more generally the lattice of all concept and relation expressions.

The expressiveness of $\mathcal{DLR}$, required for capturing meaningful properties in the DM, makes reasoning a

$$\begin{array}{rcl}
\mathtt{CONTRACT}_0 & \sqsubseteq & (\$1{:}\,\mathtt{Client}_0) \sqcap (\$2{:}\,\mathtt{Dept}_0) \sqcap \\
 & & (\$3{:}\,\mathtt{Service}_0) \\
\mathtt{REG\text{-}AT}_0 & \sqsubseteq & (\$1{:}\,\mathtt{Client}_0) \sqcap (\$2{:}\,\mathtt{Dept}_0) \\
\mathtt{PrDept}_0 & \sqsubseteq & \mathtt{Dept}_0 \\
 & & \\
\mathtt{REG\text{-}AT}_1 & \sqsubseteq & (\$1{:}\,\mathtt{Client}_1) \sqcap (\$2{:}\,\mathtt{Dept}_1) \\
\mathtt{PROMOTION}_1 & \sqsubseteq & \mathtt{REG\text{-}AT}_1 \\
\mathtt{LOCATION}_1 & \sqsubseteq & (\$1{:}\,\mathtt{Dept}_1) \sqcap (\$2{:}\,\mathtt{String}) \\
\mathtt{Dept}_1 & \sqsubseteq & \exists^{\leq 1}\mathtt{LOCATION}_1[\$1].\top_2 \\
 & & \\
\mathtt{CONTRACT}_2 & \sqsubseteq & (\$1{:}\,\mathtt{Client}_2) \sqcap (\$2{:}\,\mathtt{Dept}_2) \sqcap \\
 & & (\$3{:}\,\mathtt{Service}_2)
\end{array}$$

$$\begin{array}{rcl}
\mathtt{Dept}_1 & \equiv_{ext} & \mathtt{PrDept}_0 \\
\mathtt{REG\text{-}AT}_1 & \sqsubseteq_{ext} & \mathtt{REG\text{-}AT}_0 \\
\mathtt{Client}_1 & \equiv_{ext} & \mathtt{Client}_0 \sqcap \exists^{\geq 1}\mathtt{REG\text{-}AT}_0[\$1].\mathtt{PrDept}_0 \\
\multicolumn{3}{l}{\mathtt{Client}_0 \sqcap \exists^{\geq 1}\mathtt{CONTRACT}_0[\$1].\top_2} \\
 & \sqsubseteq_{ext} & \exists^{\geq 1}\mathtt{PROMOTION}_1[\$1].\top_2 \\
 & & \\
\mathtt{Client}_2 & \sqsubseteq_{ext} & \mathtt{Client}_0 \sqcap \exists^{\geq 1}\mathtt{CONTRACT}_0[\$1].\top_2 \\
\mathtt{Dept}_2 & \sqsubseteq_{ext} & \mathtt{Dept}_0 \\
\mathtt{Service}_2 & \equiv_{ext} & \mathtt{Service}_0 \\
 & & \\
\mathtt{Client}_1 & \equiv_{int} & \mathtt{Client}_2 \\
\mathtt{Dept}_1 & \equiv_{int} & \mathtt{Dept}_2
\end{array}$$

Figure 3: Domain model ($(\$i/n{:}\,C)$ is abbreviated by $(\$i{:}\,C)$)

complex task. We have devised a sound and complete procedure to decide the satisfiability of a DM which works in worst-case deterministic exponential time in the size of the DM. Indeed, this worst-case complexity is inherent to the problem, therefore reasoning with respect to a DM is EXPTIME complete. The inference method works in two steps: first, reasoning on the DM is reduced to reasoning on a knowledge base expressed in the Description Logic $\mathcal{CIQ}$ (De Giacomo & Lenzerini, 1996); then reasoning procedures for $\mathcal{CIQ}$, based on the correspondence with Propositional Dynamic Logics, are exploited.

For reasoning at the logical level, we provide suitable techniques for query containment. In particular, we have developed an algorithm for deciding query containment with respect to a DM, which exploits a reduction to unsatisfiability in $\mathcal{CIQ}$, and which extends the one in (Calvanese, De Giacomo, & Lenzerini, 1997; Calvanese et al., 1998) to deal with both intramodel and intermodel assertions.

3.4 EXAMPLE

Figure 3 shows a DM, $\mathcal{W} = (\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \mathcal{G})$, that represents an enterprise and two sources containing information about contracts between clients and departments for services, and about registration of clients at departments. Symbols subscripted by i refer to model $\mathcal{M}_i$. The intramodel assertions in $\mathcal{M}_0$, $\mathcal{M}_1$, $\mathcal{M}_2$ are visualized in Figure 4, using Entity-Relationship diagrams, which are typical of conceptual modeling in Databases and are fully compatible with $\mathcal{DLR}$. Source 1 contains information about clients registered at public-relations departments. Source 2 contains information about contracts and complete information about services. The Enterprise Model provides a reconciled conceptual description of the two sources. Note that, in this example, such reconciled description is not complete yet: e.g., the relation PROMOTION is not modeled in $\mathcal{M}_0$ (recall that our approach to integration is incremental). The various interdependencies among relations and concepts in the Enterprise Model and the two Sources Models are represented by the intermodel assertions on the right-hand side of Figure 3.

As for the logical level representation, suppose, for example, that the actual data in Source 1 are described by a relational table $\mathtt{Table}_1$ having three columns, one for the client, one for the department which the client is registered at, and one for the location of the department. Such a table is specified in terms of the DM by means of the query:

$$\mathtt{Table}_1(x,y,z) \;\leftarrow\; \mathtt{REG\text{-}AT}_1(x,y) \wedge \mathtt{LOCATION}_1(y,z)$$

Using the reasoning services associated with $\mathcal{DLR}$, we can automatically derive logical consequences of the DM. For instance, we can prove that the assertion $\mathtt{PROMOTION}_1 \sqsubseteq_{ext} \mathtt{REG\text{-}AT}_0 \sqcap (\$2{:}\,\mathtt{PrDept}_0)$ is a logical consequence of $\mathcal{W}$. Observe that, although $\mathcal{M}_0$ does not contain a relation PROMOTION, the above assertion relates $\mathtt{PROMOTION}_1$ to $\mathcal{M}_0$ in a precise way.

Next, consider, for instance, the following queries posed to $\mathcal{M}_0$:

$$\begin{array}{rcl}
q_1(x,y) & \leftarrow & \mathtt{Client}_0(x) \wedge \mathtt{CONTRACT}_0(x,y,z) \\
q_2(x,y) & \leftarrow & \mathtt{Client}_0(x) \wedge \mathtt{CONTRACT}_0(x,y,z) \wedge \\
 & & \mathtt{REG\text{-}AT}_0(x,w) \wedge \mathtt{PrDept}_0(w)
\end{array}$$

q_2 is obviously contained in q_1. However, taking into account the assertions in $\mathcal{W}$, we can also derive that q_1 is contained in q_2 wrt $\mathcal{W}$.

4 THE METHODOLOGY

We outline a methodology for Information Integration, based on the techniques previously described, which can be applied in the context of both virtual and materialized data integration. The proposed methodology

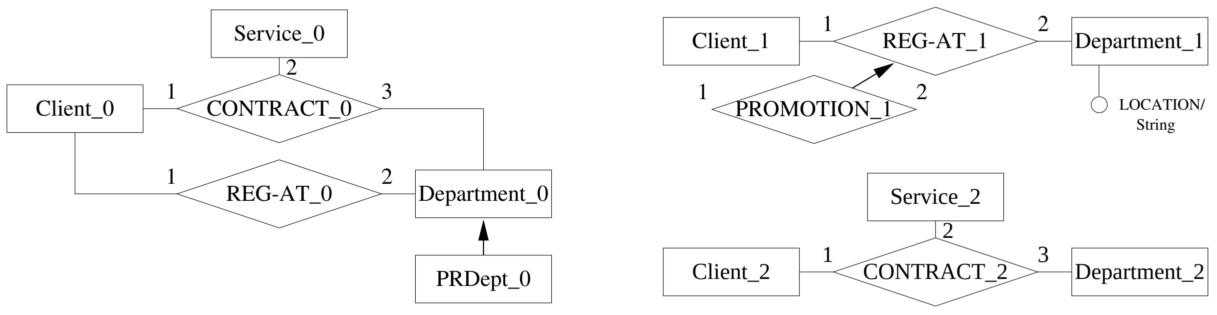

Figure 4: Enterprise and source models in Entity-Relationship diagrams

focusses on the conceptual layer of the system. Once the knowledge about this layer is available, one can exploit reasoning to support various aspects related to the other layers. We shall discuss various kinds of information that the designer can obtain through the reasoning services on the knowledge base, but the problems arising in the design of the logical and physical levels of the system are outside of the scope of the present paper. The methodology deals with two scenarios, called *source-driven* and *client-driven*. The former applies whenever the design of the system is accomplished in a top-down fashion by incrementally adding new sources of data; the latter arises when the design is developed bottom-up to satisfy the requests for data by the user applications.

4.1 SOURCE-DRIVEN INTEGRATION

Source-driven integration is triggered when a new source or a new portion of a source is taken into account for integration. The steps to be accomplished in this case are:

(1) *Source Model construction.* The Source Model capturing the concepts and the relationships of the new source that are critical for the enterprise is produced. Since in a typical setting, sources already exist, this task may be accomplished through a reverse engineering activity (Batini, Ceri, & Navathe, 1992). However, it is worth stressing that the Source Model is really meant to capture the semantics of the domain, independently of the organization of the data recorded in the physical structures. To this end, our approach provides a very expressive modeling language, which, as already pointed out, embodies the features of the most popular data models. In addition, in our formalism, it is possible to reason about the Source Model. Once a formalization of the model in terms of $\mathcal{DLR}$ is provided, checking several interesting properties of the model becomes possible, and can be used to help correctness and optimality of the design. We refer to (Calvanese et al., 1994) for a discussion on using the inference techniques associated to Description Logics during Source Model construction.

(2) *Source Model integration.* The Source Model is *integrated into the Domain Model.* This can lead to changes both to the Source Models, and to the Enterprise Model. It is worth recalling that our approach to integration is mainly declarative: the most important efforts during source integration are thus devoted to single out and to specify the intermodel assertions relating the Enterprise Model and the Source Models, rather than producing a unified conceptual representation. More precisely, the step of Source Model integration is characterized by the following activities:

- Structural and semantic conflicts involving the Source Model under analysis are detected and solved.
- Intermodel assertions between the Source Model and the Enterprise Model, and between different sources, are added to the Domain Model.

The activity of conflict resolution in our framework can be carried out by relying on the large body of work developed in database integration. More specifically, in our framework, the basic structural and semantic conflicts are very similar to those arising in the Entity-Relationship Data Model. An example of structural conflict is represented by the situation where the same concept is represented as a class in one model and as a relation in another model. The principles for resolving such conflicts are now well established (Batini et al., 1992). Other types of conflicts are dealt with in the Quality Analysis step.

The specification of intermodel assertions and the derivation of implicit relationships by exploiting the reasoning techniques, represent the novel part of the methodology. The most common intermodel assertions are those specifying the relation between elements in one Source Model with elements in the Enterprise Model. However, also assertions relating elements in different Source Models are of importance. For example, inferring that the set of instances of a relation in source S_i is always a subset of those in source S_j can be important in order to infer that accessing source S_j for retrieving instances of the relation is useless. We point out that the possibility of expressing relationships between concepts in different sources is a distinguished feature of our approach.

Intermodel assertions can be roughly classified as follows:

- *Subsetting assertions*, that are used to state that a certain concept or relation in the Source Model is a subset of another concept or relation in the Enterprise Model (or in another source). These assertion have the form $L_i \sqsubseteq_{ext} L'_j$.
- *Definition assertions*, that are used to completely characterize the set of instances of one concept in a model in terms of the set of instances of a concept in another model.
- *Completeness assertions*, that are used to state that the set of instances of a concept or relation in the Enterprise Model can be obtained as the union of different concepts or relations in the various sources. A special case of this type of assertions is the one stating that a certain concept in the Enterprise Model is fully captured by a concept in one Source Model.
- *Synonym assertions*, that are used to state that different symbols in two models denote in fact the same concept. These assertions have the form $L_i \equiv_{int} L_j$.
- *Homonym assertions*, that are used to state that the same symbol is used to denote different concepts in different models. These assertions have the form $L_i \not\equiv_{int} L_j$.

It is important to observe that the possibility of using complex concept and relation expressions in the context of intermodel assertions greatly enhances the expressive power of such assertions, and is another distinguished feature of our approach. Moreover, the possibility of reasoning about intermodel assertions provides support and guidelines to the designer of the Domain Model, as pointed out in the discussion on the step of quality analysis. Finally, we note that the usage of intermodel assertions is required also to reason about queries which is addressed in client-driven integration.

(3) *Quality analysis.* The goal of this step is to verify that the quality requirements are met by the Domain Model. In particular, the reasoning capabilities of our approach allow for dealing with several quality factors, such as:

- *Consistency* of the Source Model in isolation.
- *Redundancy*, by identifying equivalent concepts.
- *Readability*, by pointing out relationships that are implicit in the model.
- *Accessibility*, which amounts to verifying which data are available in the Materialized View Store, which data can be extracted from the sources, and which are indeed needed from external sources.
- *Believability*, which amounts to verifying whether the data available in the materialized views or provided by a source are consistent and complete.

It is worth noticing that, depending on the result of the evaluation of the quality factors, a restructuring of both the Source Model and the Enterprise Model may be required.

(4) *Source Schema specification.* The Source Schema, i.e. the logical view of the new source or a new portion of the source (expressed as a collection of queries over the corresponding Source Model) is specified. The source schemas are used in order to determine the sources relevant for computing answers to queries, by exploiting the ability to reason about queries. Notice that, the actual logical design of the sources is outside the scope of the integration system. Therefore, the focus here is on the specification of the sources at the logical level.

(5) *Materialized View Schema restructuring.* This step is done only in Materialized Data Integration. As we said before, the Materialized View Schema is specified in terms of a set of relational tables, each one described as a query over the Domain Model. On the basis of the description of the new source, an analysis can be carried out on whether the Materialized View Schema should be restructured and/or modified in order to better satisfy quality criteria. Again several quality factors can be evaluated by exploiting reasoning, which, in this case, essentially amounts to query containment. Although the design of the Materialized View Schema is outside the scope of the present work, we point out that this task can be effectively supported by the reasoning services about the representation of the logical schemata in terms of queries.

4.2 CLIENT-DRIVEN INTEGRATION

The client-driven design strategy refers to the case when a new query (or a set of queries) posed by a client is considered. The query is expressed in terms of the domain model, and the reasoning facilities are exploited to analyze and systematically decompose the query and check whether its components are subsumed by the views defined in the various schemas. Therefore, the central reasoning service for query analysis is query containment checking.

In Materialized Data Integration, the analysis is carried out as follows:

(1) We verify whether and how the answer can be computed from the materialized views. This problem is known as the *query rewriting* problem, which amounts to find a way to rewrite the original query in terms of the relations in the Materialized View Schema. Although we do not have a method for automatically rewriting the query, we can exploit query containment checking in order to support the designer in this task.

(2) In the case where the materialized views are not sufficient to compute the answer to a query, the idea is to verify whether the answer can be obtained by materializing new concepts represented in the Domain Model. It is interesting to observe that this is again an instance of the query rewritinng problem, where one aims at expressing the query in terms of the relations in the Sources. In this case, query containment helps to identify the set of subqueries to be issued on the sources and to extend and/or restructure the Materialized View Schema (see step 5 of source-driven integration). Different choices can be identified, based on various preference criteria. E.g., in (Levy et al., 1995) minimization in terms of the number of sources is proposed, based on the observation that accessing a source is the most expensive part of the process. In fact, by exploiting the information available through the intermodel assertions, we can accommodate different kinds of constraints, that are related to the above mentioned quality factors. For example, we can optimize with respect to believability, or interpretability, possibly combining different factors.

(3) In the case where neither the materialized data nor the concepts in the Domain Model are sufficient, the necessary data should be searched for in new sources, or in new portions of already analyzed sources. The new (portions of the) sources are then added to the Domain Model using the source-driven approach, and the process of analyzing the query is iterated.

In Virtual Data Integration, the basic problem is to determine whether and how the answer can be computed from the data in the analyzed sources, falling into case (2) or (3).

5 CONCLUSIONS

In this paper we have presented the fundamental features of a declarative approach to Information Integration based on Description Logics. As pointed out in the previous sections, there are a number of issues that deserve further investigation, and in particular:

- How to exploit the knowledge about the conceptual level in the design of wrappers and mediators.
- Designing automatic methods and techniques for the query rewriting problem, arising in the client-driven integration.

We are currently studying the above issues within the ESPRIT Project DWQ (Foundations of Data Warehouse Quality) (Calvanese, De Giacomo, Lenzerini, Nardi, & Rosati, 1997), where we are using the presented framework in the context of data warehouse design.

Acknowledgments

This work was partly supported by MURST, ESPRIT LTR Project No. 22469 DWQ (Foundations of Data Warehouse Quality), the Italian Research Council (CNR) under Progetto Strategico "Informatica nella Pubblica Amministrazione", sottoprogetto PROGRESS (Reingegnerizzazione dei Processi e dei Dati nella Pubblica Amministrazione), and the Italian Space Agency (ASI) under project "Integrazione ed Accesso a Basi di Dati Eterogenee".

References

Arens, Y., Chee, C. Y., Hsu, C., & Knoblock, C. A. (1993). Retrieving and integrating data from multiple information sources. *Journal of Intelligent and Cooperative Information Systems*, *2*(2), 127–158.

Arens, Y., Knoblock, C. A., & Chen, W. (1996). Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, *6*, 99–130.

Batini, C., Ceri, S., & Navathe, S. B. (1992). *Conceptual Database Design, an Entity-Relationship Approach.* Benjamin and Cummings Publ. Co., Menlo Park, California.

Batini, C., Lenzerini, M., & Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, *18*(4), 323–364.

Borgida, A. (1995). Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering*, *7*(5), 671–682.

Calvanese, D., De Giacomo, G., & Lenzerini, M. (1995). Structured objects: Modeling and reasoning. In *Proc. of the 4th Int. Conf. on Deductive and Object-Oriented Databases (DOOD-95)*, No. 1013 in Lecture Notes in Computer Science, pp. 229–246. Springer-Verlag.

Calvanese, D., De Giacomo, G., & Lenzerini, M. (1997). Conjunctive query containment in Description Logics with n-ary relations. In *Proc. of the 1997 Description Logic Workshop (DL-97)*, pp. 5–9.

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)*. To appear.

Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., & Rosati, R. (1997). Source integration in data warehousing. Tech. rep. DWQ-UNIROMA-002, DWQ Consortium.

Calvanese, D., Lenzerini, M., & Nardi, D. (1994). A unified framework for class based representation formalisms. In Doyle, J., Sandewall, E., & Torasso, P. (Eds.), *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-94)*, pp. 109–120 Bonn. Morgan Kaufmann, Los Altos.

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

Collet, C., Huhns, M. N., & Shen, W.-M. (1991). Resource integration using a large knowledge base in Carnot. *IEEE Computer*, *24*(12), 55–62.

De Giacomo, G. & Lenzerini, M. (1994). Description logics with inverse roles, functional restrictions, and n-ary relations. In *Proc. of the 4th European Workshop on Logics in Artificial Intelligence (JELIA-94)*, Vol. 838 of *Lecture Notes in Artificial Intelligence*, pp. 332–346. Springer-Verlag.

De Giacomo, G. & Lenzerini, M. (1995). What's in an aggregate: Foundations for description logics with tuples and sets. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI-95)*, pp. 801–807.

De Giacomo, G. & Lenzerini, M. (1996). TBox and ABox reasoning in expressive description logics. In Aiello, L. C., Doyle, J., & Shapiro, S. C. (Eds.), *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-96)*, pp. 316–327. Morgan Kaufmann, Los Altos.

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1991). A hybrid system integrating Datalog and concept languages. In *Proc. of the 2nd Conf. of the Italian Association for Artificial Intelligence (AI*IA-91)*, No. 549 in Lecture Notes in Artificial Intelligence. Springer-Verlag. An extended version appeared also in the Working Notes of the AAAI Fall Symposium "Principles of Hybrid Reasoning".

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1996). Reasoning in description logics. In Brewka, G. (Ed.), *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pp. 193–238. CSLI Publications.

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1998). AL-log: Integrating datalog and description logics. *Journal of Intelligent Information Systems*. To appear.

Patel-Schneider, P., McGuiness, D., Brachman, R. J., Resnick, L., & Borgida, A. (1991). The CLASSIC knowledge representation system: Guiding principles and implementation rational. *SIGART Bulletin*, *2*(3), 108–113.

Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J., & Widom, J. (1994). The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of IPSI Conference (IPSI'94)*.

Gupta, A. & Mumick, I. S. (1995). Maintenance of materialized views: Problems, techniques, and applications. *IEEE Bulletin of the Technical Committee on Data Engineering*, *18*(2), 3–18.

Hammer, J., Garcia-Molina, H., Widom, J., Labio, W., & Zhuge, Y. (1995). The Stanford data warehousing project. *IEEE Bulletin of the Technical Committee on Data Engineering*, *18*(2), 41–48.

Huhns, M. N., Jacobs, N., Ksiezyk, T., Shen, W.-M., Singh, M. P., & Cannata, P. E. (1993). Integrating enterprise information models in Carnot. In *Proc. of the Int. Conf. on Cooperative Information Systems (CoopIS-93)*, pp. 32–42.

Hull, R. (1997). Managing semantic heterogeneity in databases: A theoretical perspective. In *Proc. of the*

16th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS-97).

Hull, R. & Zhou, G. (1996). A framework for supporting data integration using the materialized and virtual approaches. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pp. 481–492.

Hurson, A., Bright, M., & Pakzad, S. (Eds.). (1994). *Multidatabase Systems: An Advanced Solution for Global Information Sharing*. IEEE Computer Society Press.

Inmon, W. H. (1996). *Building the Data Warehouse* (second edition). John Wiley & Sons.

Kirk, T., Levy, A. Y., Sagiv, Y., & Srivastava, D. (1995). The Information Manifold. In *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments*, pp. 85–91.

Knoblock, C. & Levy, A. (Eds.). (1995). *AAAI Symposium on Information Gathering from Heterogeneous, Distributed Environments*, No. SS-95-08 in AAAI Spring Symposium Series. AAAI Press/The MIT Press.

Lenat, D. & Guha, R. V. (1990). *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison Wesley Publ. Co., Reading, Massachussetts.

Levy, A. Y., Rajaraman, A., & Ordille, J. J. (1996). Query answering algorithms for information agents. In *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI-96)*, pp. 40–47.

Levy, A. Y. & Rousset, M.-C. (1996). CARIN: A representation language combining Horn rules and description logics. In *Proc. of the 12th European Conf. on Artificial Intelligence (ECAI-96)*, pp. 323–327.

Levy, A. Y., Srivastava, D., & Kirk, T. (1995). Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, *5*, 121–143.

MacGregor, R. (1991). Inside the LOOM description classifier. *SIGART Bulletin*, *2*(3), 88–92.

Sheth, A. & Larson, J. (1991). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, *22*(3).

Ullman, J. D. (1997). Information integration using logical views. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT-97)*, Lecture Notes in Computer Science, pp. 19–40. Springer-Verlag.

Widom, J. (1995). Special issue on materialized views and data warehousing. *IEEE Bulletin on Data Engineering*, *18*(2).

Wiederhold, G. (1992). Mediators in the architecture of future information systems. *IEEE Computer* (March), 38–49.

Wiederhold, G. (1996). Special issue: Intelligent integration of information. *Journal of Intelligent Information Systems*, *6*(2/3).

Wiener, J. L., Gupta, H., Labio, W. J., Zhuge, Y., Garcia-Molina, H., & Widom, J. (1996). A system prototype for warehouse view maintenance. Tech. rep., Stanford University. Available at http://www-db-stanford.edu/warehousing/warehouse.html.

Zhou, G., Hull, R., & King, R. (1996). Generating data integration mediators that use materializations. *Journal of Intelligent Information Systems*, *6*, 199–221.