Knowledge Representation Approach to 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

In recent years there has been a growing interest in accessing, relating, and combining 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. 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 [37, 26, 35, 20].

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 sub-schemata, each one produced independently from the others [3]. 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 [34]. 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 [33, 22], whereas in the second case, the system maintains a reconciled, replicated view of the data at the sources [17, 23].

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 [10, 34], SQUIRREL [39, 21], and WHIPS [18, 38] follow this idea. They do not require an explicit notion of integrated data schema, and rely on *wrappers* to encapsulate sources and *mediators* to merge and reconcile data coming from wrappers and other 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. This is the idea underlying systems such as *Carnot* [11, 19], SIMS [1, 2] and *Information Manifold* [30, 25, 28]. The declarative approach provides a crucial advantage over the procedural one: although building a unified representation may be costly, it represents a reusable component of the Information Integration system.

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 [16, 4] based framework for Information Integration. In particular, our work provides the following main contributions:

1. We present a novel architecture for an integration system, which allows one to explicitly represent data and information needs at various levels.

2. At the conceptual level we use Description Logics for *modeling* 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.

   Additionally, 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 all data at the logical level 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. We provide *inference procedures* for the fundamental reasoning services, namely concept and relation subsumption, and query containment. Indeed, we present the first decidability result on query containment for a Description Logic with n-ary relations [6]. 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.

Compared with the procedural approaches, which have been designed to cope in a more flexible way with 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 architecture for Information Integration. In Section 3 we present the particular Description Logic we use to realize the architecture. 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. Section 5 concludes the paper.

2 Architecture of integration systems

In this section we describe the architecture of an integration system resulting from the introduction of a conceptual layer. In particular, we illustrate both the various components that are maintained and used by the system, and the tasks that the system has to carry out for performing its job. The proposed architecture serves as a general setting where different approaches to integration can be evaluated and compared. Indeed, we illustrate how existing integration systems can be obtained as specializations of this general architecture.

2.1 Components

The data structures managed by an integration system are shown in Figure 1, where four levels are singled out: *conceptual*, *logical*[1], *physical*, and *meta*. Furthermore, Figure 1 includes the following elements, which are outside the boundary of the integration system:

- The *Interface*, which is the module that allows the communication with both the user (i.e. anyone interested in retrieving information) and the designer (i.e. the one in charge of the building and the functioning of the system).

- The *External Sources*, which represent the independent systems managing the actual data that the system is supposed to integrate.

[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 conceptual level

The conceptual level contains a formal description of the concepts, the relationships between concepts, and the information requirements that the integration application has to deal with. The key feature of this level is that such a description is independent from any system consideration, and is oriented towards the goal of expressing the semantics of the application. In particular, we distinguish among the following elements in the conceptual level:

- The *Enterprise Model*[2] 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.

- For an information source S, the *Source Model* of S is a conceptual representation of the data residing in S.

- The term *Domain Model* is used to denote the union of both the Enterprise Model and the various Source Models, plus possible intermodel relationships, i.e. relationships holding between concepts belonging to different models (i.e. between one concept in source S and one concept in the Enterprise Model, or between one concept in one source and one concept in another source).

- A *Query Model* is a conceptual representation of an information need. An example of Query Model is a relational query over the Domain Model.

We point out that the Domain Model contains *intermodel relationships*, i.e. 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 architecture. Since the sources are of interest in the system, integration does not simply mean producing the Enterprise Model, but rather to be able to establish the correct relationships both between the Source Models and the Enterprise Model, and between the various Source Models.

The logical level

The *logical level* contains the description of the data and the queries of interest to the system, expressed in terms of typical logical structures managed by DBMSs. In particular, the *Source Schema* of a source S describes the logical content of S and the *Materialized View Schema* describes the logical content of the materialized views maintained by the system. Collectively, the Source Schemas and the Materialized View Schema form what we call the *Data Schema*. Obviously, the Materialized View Schema is meaningful only in the case where the integrated data (or portions thereof) are materialized, whereas it is meaningless in the case of fully virtual integration. Finally, the *Query Schemas* express the information needs at the logical level, for example as a set of relational queries over the Data Schema.

The physical level

In our architecture, the physical level refers to the actual data managed by the system. Therefore, the physical level is the one where the extensional information of the system is taken into account. In particular, the *Materialized View Store* contains the data that the system maintains materialized. Figure 1 shows also wrappers and mediators at this level. A *wrapper* is a software module that is able

[2]Here the term "model" is used to denote a formal description in a given representation language. Note the difference with the usual meaning in databases, where it denotes the formalism itself.

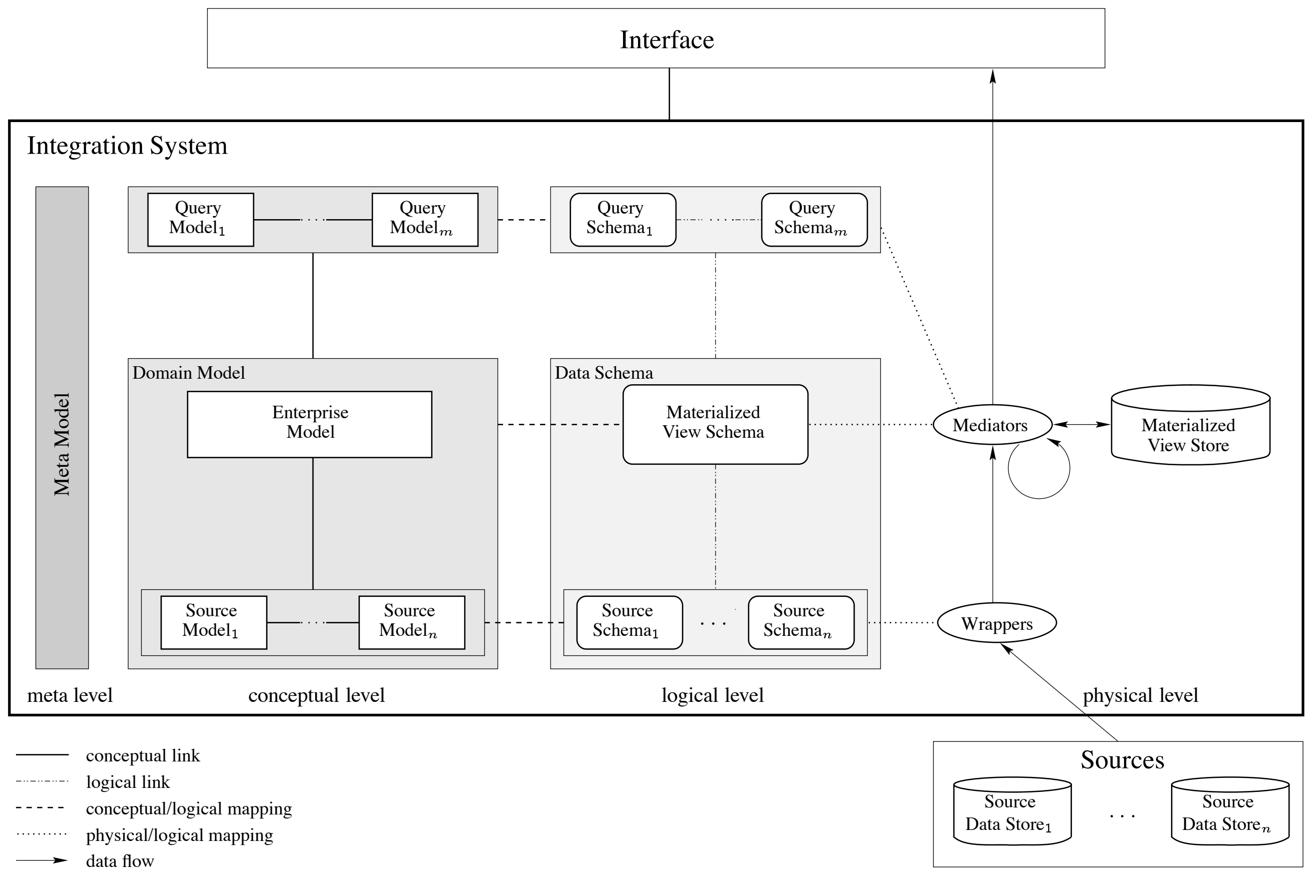

Figure 1: Architecture for Data Integration

to access a source and retrieve the data therein in a form that is coherent with the logical specification of the source.

A *mediator* [36] is a software module that takes as input sets of data produced by either wrappers or other mediators, refines this information by integrating and resolving conflicts, and produces as output another set of data, namely the one corresponding to the result of a given query. In other words, a mediator is always associated to a particular query at the logical level. The result of a mediator can be either materialized, transferred to the interface, or transferred to another mediator.

The meta level

The meta level comprises the *Meta Model*, which is the repository with all meta information about the various system components, and is used by both the user and the designer. A more detailed discussion of the meta level is outside the scope of this paper, and can be found, for example, in [24].

Mappings

Figure 1 also shows the mappings between the conceptual and the logical level, and between the logical and the physical level.

Regarding the first aspect, the mapping between Source Models and Source Schemas represents the fact that the correspondence between the logical representation of data in the sources and concepts in the Source Models should be explicit. The same holds for information needs expressed at the conceptual level and queries expressed at the logical level. Finally, the correspondence between elements of the Domain Model and the Materialized View Schema represents the information on which are the concepts and relationships that are materialized in the views maintained by the system.

Regarding the second aspect, the mapping between mediators and Query Schemas and/or the Materialized Views Schema makes explicit the fact that each mediator is supposed to compute the extension of a logical object, which can be either materialized or not. A wrapper is always associated to an element of a Source Schema, namely, the one whose data are extracted and retrieved by the wrapper. The mapping with Source Schemas represents exactly the correspondence between a wrapper and the logical element whose extensional data are extracted from the source by the wrapper.

2.2 Tasks

In this section we briefly discuss the tasks that should be carried out during the use of an integration system conforming to our architecture.

The first class of tasks comprises all the activities regarding the definition of the different elements of the architecture. Such activities mainly pertain to the design of the integration system. For example, the specification of the various Conceptual Models and the intermodel links belongs to this phase. We note that the architecture does not prescribe to build the conceptual level in one shot, but rather supports an incremental definition of both the Domain and the Query Models. Indeed, such models are subject to changes and additions as the analysis of the information sources proceeds.

Observe that in the (partially) materialized approach to integration, one of the most critical tasks is the decision of what and how to materialize. Moreover, in both the materialized and the virtual approach, the task of wrapper and mediator design is extremely important. Designing a wrapper means to decide how to access the source in order to retrieve data, and designing a mediator means to decide how to use wrappers in order to answer a particular query or to materialize a particular view. Note that the design of a mediator comprises the resolution of conflicts and/or heterogeneity of data residing in different sources.

The second kind of tasks includes all the design activities to be performed when a new information need arises. In this case, the new query has to be compared with those computed by the available mediators. The most important problem here is the one of *query rewriting*, i.e. checking if and how the new query can be reformulated in terms of those computed by the existing mediators. In virtual integration, this may lead the new mediator to simply call for the existing mediators. In materialized integration, reformulating the query in terms of the materialized views means avoiding to access the sources. Conversely, if the query (or part thereof) cannot be answered by simply relying on the existing materialized views, a new view (or new views) should be materialized, and the problem of query rewriting arises in a different form: the new view to materialize is seen as a query that has to be formulated in terms of the Source Schemas.

Finally, the third class of tasks concerns the activities that are routinely carried out during the operational phase of the systems, namely data extraction, query computation, and view materialization and refreshment.

2.3 Comparison with existing systems

The architecture outlined above can be instantiated to different information integration settings.

Schema integration [3] In the schema integration setting, integration starts by providing a conceptual representation of the sources (Source Models), and proceeds by generating the global database schema (Enterprise Model). Such a schema is then used for the design of the implemented database (Materialized View Schema, Materialized View Store). Once such database has been created, the sources are discarded and the conceptual level is not used anymore.

Multidatabases [33, 22] The setting of multidatabases deals with different sources, which are considered as internal components of the Integration System. Based on a logical representation of the sources, mediators are designed in order to satisfy information needs also expressed at the logical level (Query Schemas). Typically, mediators do not materialize data in the system. Also, the conceptual level is generally not taken into account.

Global information systems [34] In this setting the goal is to provide tools for the integrated access to multiple and diverse autonomous information sources and repositories, such as databases, HTML documents, unstructured files. Among the systems proposed in this framework, Information Manifold [30, 25, 28] uses a representation at the conceptual level of a reconciled view (called World View) of the information sources and no data is materialized. Also TSIMMIS [10, 34] deals with a virtual scenario, but does not provide a conceptual representation of data. One difference between the above two systems is that in the former, data at the sources are described as views over the World View, whereas in the latter, each mediator computes a view over the sources. Both these strategies have disadvantages: in the first case intersource relationships are not expressible, and in the second case general concepts cannot be characterized independently from the sources.

A declarative approach is taken also in Carnot and SIMS, in which reasoning is based on formalisms (Cyc [27] and LOOM [31], respectively) that are undecidable. Information Manifold uses the Classic [32] 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 the relationships among the various classes of data in the domain, nor the intermodel assertions.

Notably, our approach, described in the next section, is based on a decidable Description Logic, but does not impose any predefined direction for expressing links between the sources and the Enterprise Model.

Data warehouses [23] In this setting views are materialized, as e.g. in the WHIPS system [18, 38], in which information is not represented at the conceptual level. The lack of a conceptual level is shared by the SQUIRREL system [41, 40, 39, 21]. However, within SQUIRREL it is also possible to take into account the case of virtual views.

3 Representation and reasoning

While Section 2 illustrates the general architecture of the integration system, the goal of this section is to describe a formalism that can be used both at the conceptual and the logical level, and the associated reasoning techniques. In this paper we do not deal with representation and reasoning on Query Models and Query Schemas.

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*[3]. $\mathcal{DLR}$ is inspired by the languages introduced in [5, 13, 12, 9], and is a natural extension of Description Logics [16, 8, 4] 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*, respectively built according to the following syntax (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)[4]:

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

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

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

[4]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.

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

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

concepts in $\mathcal{M}$, and have the form

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

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* [9], 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, the 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 $\top_{1j}$). For example, an interpretation $\mathcal{I}$ satisfies the intermodel assertion $\mathbf{R}_i \sqsubseteq_{int} \mathbf{R}'_j$ if $\top_{ni}^{\mathcal{I}} \cap \top_{nj}^{\mathcal{I}} \cap \mathbf{R}_i^{\mathcal{I}} \subseteq \top_{ni}^{\mathcal{I}} \cap \top_{nj}^{\mathcal{I}} \cap \mathbf{R}_j'^{\mathcal{I}}$.

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 $\mathcal{W}$.

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 related 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 $\vec{\mathbf{x}}, \vec{\mathbf{y}}_i$)[5], with $\mathbf{R}$, C relations and concepts over the DM. The *arity* of q is equal to the number of variables of $\vec{\mathbf{x}}$.

We observe that, by means of assertions on both relations and concepts expressed in the DM, additional constraints than those directly present in the query can be imposed. This distinguishes our approach with respect to [15, 29], where n-ary relations appearing in queries are not part of the conceptual model.

[5]Our approach works also when constants are used 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 $\mathcal{W}$.

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 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 $\mathcal{CIQ}$ [14]; 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 [6, 7] to deal with both intramodel and intermodel assertions.

Example Figure 3 shows a DM, $\mathcal{W} = \langle \mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \mathcal{G} \rangle$, 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

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

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

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

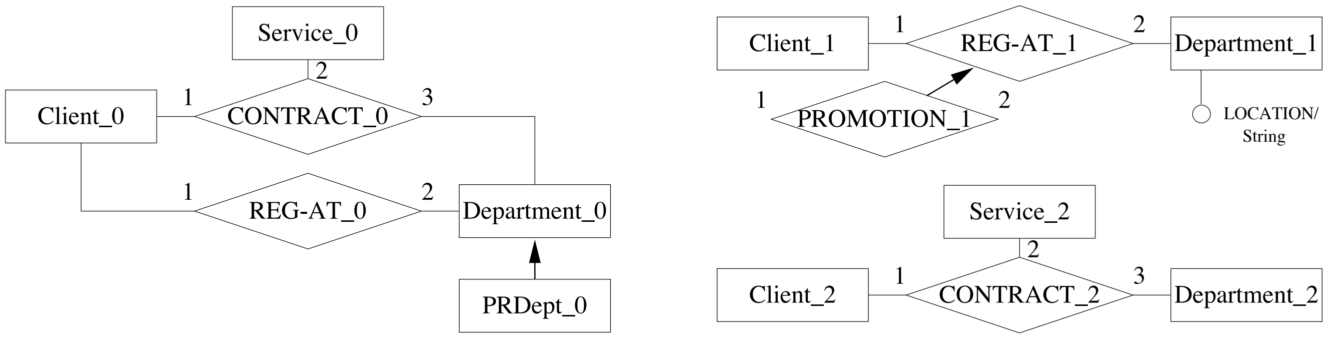

Figure 4: Enterprise and source models in Entity-Relationship diagrams

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 $\texttt{Table}_1$ 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:

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

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 $\texttt{PROMOTION}_1 \sqsubseteq_{ext} \texttt{REG-AT}_0 \sqcap (\$2\colon \texttt{PrDept}_0)$ is a logical consequence of $\mathcal{W}$. Observe that, although $\mathcal{M}_0$ does not contain a relation $\texttt{PROMOTION}$, the above assertion relates $\texttt{PROMOTION}_1$ to $\mathcal{M}_0$ in a precise way.

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

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

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 methodology deals with two scenarios, called *source-driven* and *client-driven*.

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.

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. The specification of intermodel assertions and the derivation of implicit relationships by exploiting the reasoning techniques, represent the novel part of the methodology. Notably, not only assertions relating elements in one Source Model with elements in the Enterprise Model, but 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.

3. *Quality analysis*. The Quality Factors of the resulting Domain Model are evaluated and a restructuring is accomplished to match the required criteria. This step requires the use of the reasoning techniques associated with our formalisms to check for quality factors such as consistency, redundancy, readability, accessibility, believability [7].

4. *Source Schema construction*. 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 produced. 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.

5. *Materialized View Schema restructuring*. This step is done only in Materialized Data Integration. On the basis of the new source, an analysis is carried out on whether the Materialized View Schema should be restructured and/or modified in order to better meet quality requirements. Again, the schema is constituted by a set of queries over the Domain Model, and for its restructuring the use of reasoning techniques is crucial. A restructuring of the Materialized View Schema may require the design of new mediators.

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 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.

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

1. By exploiting query containment checking, we verify if and how the answer can be computed from the materialized views.

2. In the case where the materialized views are not sufficient, we verify if the answer can be obtained by materializing new concepts represented in the Domain Model. In this case, query containment helps to identify the set of sub-queries to be issued on the sources and to extend and/or restructure the Materialized View Schema. Different choices can be identified, based on various preference criteria (e.g. minimization of the number of sources, as in [30]) which take into account the above mentioned quality 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, one has 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

We have presented the fundamental features of a declarative approach to Information Integration based on Description Logics. We are currently applying the presented framework to the problem of data warehouse design within the ESPRIT Project DWQ (Foundations of Data Warehouse Quality).

References

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

[2] Y. Arens, C. A. Knoblock, and W. Chen. Query reformulation for dynamic information integration. *J. of Intelligent Information Systems*, 6:99–130, 1996.

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

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

[5] D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured objects: Modeling and reasoning. In *Proc. of the 4th Int. Conf. on Deductive and Object-Oriented Databases (DOOD-95)*, number 1013 in Lecture Notes in Computer Science, pages 229–246. Springer-Verlag, 1995.

[6] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS-98)*, 1998.

[7] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Source integration in data warehousing. Technical Report DWQ-UNIROMA-002, DWQ Consortium, Oct. 1997.

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

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

[10] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of IPSI Conf. (IPSI'94)*, Tokyo (Japan), 1994.

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

[12] G. De Giacomo and M. Lenzerini. 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)*, volume 838 of *Lecture Notes In Artificial Intelligence*, pages 332–346. Springer-Verlag, 1994.

[13] G. De Giacomo and M. Lenzerini. 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)*, pages 801–807, 1995.

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

[15] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. A hybrid system integrating Datalog and concept languages. In *Proc. of the 2nd Conf. of the Italian Association for Artificial Intelligence (AI*IA-91)*, number 549 in Lecture Notes In Artificial Intelligence. Springer-Verlag, 1991. An extended version appeared also in the Working Notes of the AAAI Fall Symposium "Principles of Hybrid Reasoning".

[16] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

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

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

[19] M. N. Huhns, N. Jacobs, T. Ksiezyk, W.-M. S. an Munindar P. Singh, and P. E. Cannata. Integrating enterprise information models in Carnot. In *Proc. of the Int. Conf. on Cooperative Information Systems (CoopIS-93)*, pages 32–42, 1993.

[20] R. Hull. 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)*, 1997.

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

[22] A. Hurson, M. Bright, and S. Pakzad, editors. *Multidatabase Systems: An Advanced Solution for Global Information Sharing*. IEEE Computer Society Press, 1994.

[23] W. H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, second edition, 1996.

[24] M. Jarke, M. A. Jeusfeld, C. Quix, and P. Vassiliadis. Architecture and quality in data warehouses. In *Proc. of the 10th Conf. on Advanced Information Systems Engineering (CAiSE'98)*, 1998.

[25] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments*, pages 85–91, 1995.

[26] C. Knoblock and A. Levy, editors. *AAAI Symposium on Information Gathering from Heterogeneous, Distributed Environments*, number SS-95-08 in AAAI Spring Symposium Series. AAAI Press/The MIT Press, 1995.

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

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

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

[30] A. Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *J. of Intelligent Information Systems*, 5:121–143, 1995.

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

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

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

[34] J. D. Ullman. Information integration using logical views. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT-97)*, number 1186 in Lecture Notes in Computer Science, pages 19–40. Springer-Verlag, 1997.

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

[36] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.

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

[38] J. L. Wiener, H. Gupta, W. J. Labio, Y. Zhuge, H. Garcia-Molina, and J. Widom. A system prototype for warehouse view maintenance. Technical report, Stanford University, 1996. Available at http://www-db-stanford.edu/warehousing/warehouse.html.

[39] G. Zhou, R. Hull, and R. King. Generating data integration mediators that use materializations. *J. of Intelligent Information Systems*, 6:199–221, 1996.

[40] G. Zhou, R. Hull, R. King, and J.-C. Franchitti. Data integration and warehousing using H20. *IEEE Bulletin of the Technical Committee on Data Engineering*, 18(2):29–40, 1995.

[41] G. Zhou, R. Hull, R. King, and J.-C. Franchitti. Using object matching and materialization to integrate heterogeneous databases. In *Proc. of the 3rd Int. Conf. on Cooperative Information Systems (CoopIS-95)*, pages 4–18, 1995.