Integrating and Exchanging XML Data using Ontologies ⋆

Huiyong Xiao and Isabel F. Cruz

Department of Computer Science
University of Illinois at Chicago
{hxiao | ifc}@cs.uic.edu

Abstract. While providing a uniform syntax and a semistructured data model, XML does not express semantics but only structure such as nesting information. In this paper, we consider the problem of data integration and interoperation of heterogeneous XML sources and use an ontology-based framework to address this problem at a semantic level. Ontologies are extensively used for domain knowledge representation, by virtue of their conceptualization of the domain, which carries explicit semantics. In our approach, the global ontology is expressed in RDF Schema (RDFS) and constructed using the global-as-view approach by merging individual local ontologies, which represent XML source schemas. We provide a formal model for the mappings between XML schemas and local RDFS ontologies and those between local ontologies and the global RDFS ontology. We consider two cases of query processing, specifically for data integration and for data interoperation. In the first case, the user poses an RDF query on the global ontology, which is answered using all the mapped XML sources. In the second case, a query is posed on a single source and then is mapped to the XML sources that are connected to that source. For each case, we discuss the problem of query containment and present an equivalent query rewriting algorithm for queries expressed in two languages: conjunctive RDQL and conjunctive XQuery.

1 Introduction

1.1 Problem description

Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data [25]. It is relevant to a number of applications including data warehousing, enterprise information integration, geographic information systems, and e-commerce applications. Data integration systems are usually characterized by an architecture based on a global schema, which provides a reconciled and integrated view of the underlying sources. These systems are called *central*

⋆ A preliminary version of this paper was presented at the 8th International Database Engineering & Applications Symposium (Isabel F. Cruz, Huiyong Xiao, Feihong Hsu: An Ontology-Based Framework for XML Semantic Integration. IDEAS 2004: 217-226). This research was partially supported by the National Science Foundation under Awards ITR IIS-0326284 and IIS-0513553.

data integration systems, and a large number of such systems have been proposed [3, 5, 11, 14, 24, 27, 30, 34, 36].

There are two key issues in central data integration, namely system modeling and query processing. For modeling the relation between the sources and the global schema, two basic approaches have been proposed [10, 25, 36]. The first approach, called Global-as-View (GaV), expresses the global schema in terms of the data sources. The second approach, called Local-as-View (LaV), requires the global schema to be specified independently from the sources, and the relationships between the global schema and the sources are established by defining every source as a view over the global schema.

Query processing in central data integration may require a query reformulation step: the query over the global schema has to be reformulated in terms of a set of queries over the sources. In the GaV approach, every entity in the global schema is associated with a view over the source local schema, therefore query processing in this case uses a simple "unfolding" strategy [25]. In contrast, query processing in LaV can be complex, since the local sources may contain incomplete information. In this sense, query processing in LaV, called *view-based query processing* [1, 12, 18], is similar to query answering with incomplete information [37]. It can also be the case that two data sources communicate in a peer-to-peer (P2P) way either through the global schema or directly. Data exchange or query processing may occur in this case, which requires data translation or query rewriting when heterogeneities are present between the communicating sources [16, 23, 27, 30, 32].

The heterogeneities between distributed data sources can be classified as *syntactic*, *schematic*, and *semantic* heterogeneities [6]. Syntactic heterogeneity is caused by the use of different models or languages (e.g., relational and XML). Schematic heterogeneity results from the different data organizations (e.g., aggregation or generalization hierarchies). Semantic heterogeneity is caused by different meanings or interpretations of data. All these heterogeneities have to be resolved, to achieve the goal of integration or interoperation. In this paper, we consider the semantic integration of XML data and data exchange between heterogeneous XML sources, using ontologies.

XML documents that represent data with similar semantics may conform to different schemas. Therefore, a user must construct queries in accordance to the different XML document's structures even if to retrieve fragments of information that have the same meaning. This fact makes the formulation of queries over heterogeneous XML sources a nontrivial burden to the user. Furthermore, this shortcoming of XML impedes the interoperation between XML sources since the reformulation of XML queries from one source to another has to eliminate the structural differences of the queries while presenting the same semantics. Let us illustrate this problem using a running example.

Example 1. Figure 1 shows two XML schemas ($\mathcal{S}_1$ and $\mathcal{S}_2$) with their instances (i.e., XML documents $\mathcal{D}_1$ and $\mathcal{D}_2$), which are represented as trees. It is obvious that $\mathcal{S}_1$ and $\mathcal{S}_2$ both represent a many-to-many relationship between two concepts: `book` and `author` (equivalently denoted `article` and `writer` in $\mathcal{S}_2$). However, structurally speaking, they are different: $\mathcal{S}_1$, which is a book-centric schema, has the `author` element nested under the `book` element, whereas $\mathcal{S}_2$, which is an author-centric schema, has the `article` element nested under the `writer` element. Suppose our query target is "Find all the authors of the publication b_2." The XML path expressions that are used

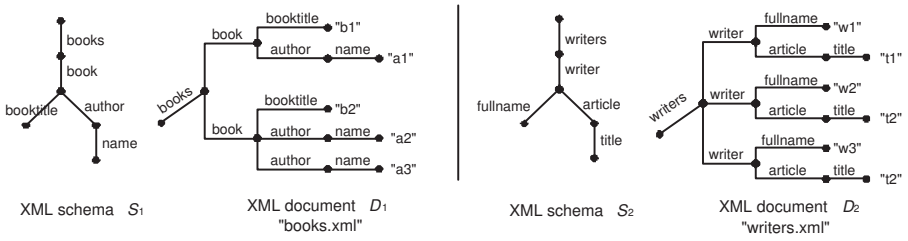

Fig. 1. Two XML sources with structural heterogeneities.

to define the search patterns in the two schema trees can be respectively written as `/books/book[booktitle.text()="b2"]/author/name` and `/writers /writer[article/title.text()="b2"]/fullname`, where the contents in the square brackets specify the constraints for the search patterns. We notice that although the above two search patterns refer to semantically equivalent concepts, they follow two distinct XML paths.

1.2 Semantic integration of XML documents

The structural diversity of conceptually equivalent XML schemas leads to the fact that XML queries over different schemas may represent the same semantics even though they are formulated using two different alphabets and structures. In comparison, the schema languages used for conceptual modeling are *structurally flat* so that the user can formulate a determined conceptual query without worrying about the structure of the source. RDF Schema (RDFS) [26], DAML+OIL, and OWL are examples of languages used to create ontologies, which represent a shared, formal conceptualization of the domain of knowledge [17]. There are currently many attempts to use conceptual schemas (or ontologies) [3, 4, 16] or conceptual queries [14, 15] to overcome the problem of structural heterogeneities among XML sources.

In this paper, we propose an ontology-based approach for the integration of XML sources. We use the GaV approach to model the mappings between the source schemas and the global ontology, which is, therefore, an integrated view of the source schemas. The global ontology is expressed in terms of RDFS, which is at the core of several ontology languages (e.g., OWL and DAML+OIL). In order to facilitate the mappings between the XML source schemas and the global RDFS ontology, their syntactic disparity needs to be reconciled. To this end, we first transform the heterogeneous XML sources into local RDFS ontologies (defined using the RDFS space [9]), which are then merged into the global ontology. This transformation process encodes the mapping information between each concept in the local ontology and the corresponding element in the XML source. The ontology merging process can be semi-automatically performed (e.g., by using the PROMPT algorithm [29]). In addition to the global ontology, the merging process also produces a *mapping table*, which contains the mapping information between concepts in the global ontology and concepts in the local ontologies.

In our approach, we can translate a query posed against the global ontology into subqueries over the sources. We can also translate a query posed against an XML source to an equivalent query against any other XML source. We call the query rewriting in the first case *global-to-local query rewriting* and that in the second case *local-to-local query rewriting*. Given that we choose a GaV approach, the global ontology is a view over the local ontologies, therefore the process of mapping a query over the global ontology to queries over the local ontologies is straightforward.

1.3 Contributions

We make the following contributions in this paper:

- We propose an ontology-based approach to the integration of heterogeneous XML sources. The global ontology takes into account both the XML nesting structure and the domain structure, which are expressed in RDFS, so as to enable semantic interoperation between the XML sources. This integration process is *lossless* with respect to the nesting structure of the XML sources, so that XML structural queries can be correctly rewritten.
- We extend the RDFS space by defining additional metadata, which enables the encoding of the nesting structure of the XML Schema in the RDF schema. We convert each of the XML source schemas into a local RDFS ontology while preserving their structure, so that they share a uniform representation with the global ontology.
- Finally, we refine the concepts of *certain answers* and of *query containment*, in two querying modes: global-to-local query rewriting and local-to-local query rewriting. Furthermore, a query rewriting algorithm that guarantees equivalence is provided for each case of query rewriting.

The paper is organized as follows. Section 2 describes related work. Section 3 describes the framework for the integration of XML sources. Data integration and query processing, which are the two key points in our approach, are discussed respectively in Sections 4 and 5. We draw conclusions and discuss future work in Section 6.

2 Related Work

There are a number of approaches addressing the problem of data integration or interoperation among XML sources. We classify those approaches into three categories, depending on their main focus, namely *semantic integration*, *query languages*, and *query rewriting*.

2.1 Semantic integration

High-level Mediator Amann *et al*. propose an ontology-based approach to the integration of heterogeneous XML Web resources in the C-Web project [3, 4]. The proposed approach is very similar to our approach except for the following differences. The first difference is that they use a local-as-view (LaV) approach [10] with a hypothetical global ontology that may be incomplete. The second difference is that

they do not retain the XML documents' structures in their conceptual mediator so they cannot deal with the reverse query translation (from the XML sources to the mediator). Our previous work involved a layered approach for the interoperation of heterogeneous web sources, but the nesting structure associated with XML was lost in the mapping from XML data to RDF data [16].

Direct Translation Klein proposes a procedure to transform XML data directly into RDF data by annotating the XML documents via external RDFS specifications [22]. The procedure makes the data in XML documents available for the Semantic Web. However, since the proposed approach does not consider the document structure of XML sources, it can not propagate queries from one XML source to another XML source.

Semantics Encoding The Yin/Yang Web approach proposed by Patel-Schneider and Siméon address the problem of incorporating the XML and RDF paradigms [31]. They develop an integrated model for XML and RDF by integrating the semantics and inferencing rules of RDF into XML, so that XML querying can benefit from their RDF *reasoner*. But the Yin/Yang Web approach does not solve the problem of query answering across heterogeneous sources, that is, sources with different syntax or data models. It also cannot process higher-level queries such as RDQL. Lakshmanan and Sadri also propose an infrastructure for interoperating over XML data sources by semantically marking up the information contents of data sources using application-specific common vocabularies [23]. However, the proposed approach relies on the availability of an application-specific standard ontology that serves as the global schema. This global schema contains information necessary for interoperation, such as key and cardinality information for predicates. This approach has the same problem as the Yin/Yang Web approach, that is, higher-level queries can not be processed downward to XML queries.

2.2 Query languages

CXQuery is a new XML query language proposed by Chen and Revesz, which borrows features from both SQL and other XML query languages [15]. It overcomes the limitations of the XQuery language by allowing the user to define views, explicitly specify the schema of the query answers, and query through multiple XML documents. However, CXQuery does not solve the issue of structural heterogeneities among XML sources. The user has to be familiar with the document structure of each XML source to formulate queries. Heuser *et al.* also present a new language (CXPath) based on *XPath* for querying XML sources at the conceptual level [14]. CXPath is used to write queries over a conceptual schema that abstracts the semantic content of several XML sources. However, they do not consider the situation of query translation from the XML sources to the global conceptual schema.

2.3 Query rewriting

Query rewriting is often a key issue for both mediator-based integration systems and peer-to-peer systems. The Clio approach, which provides an example for the former case, mainly addresses schema mapping and data transformation between nested schemas

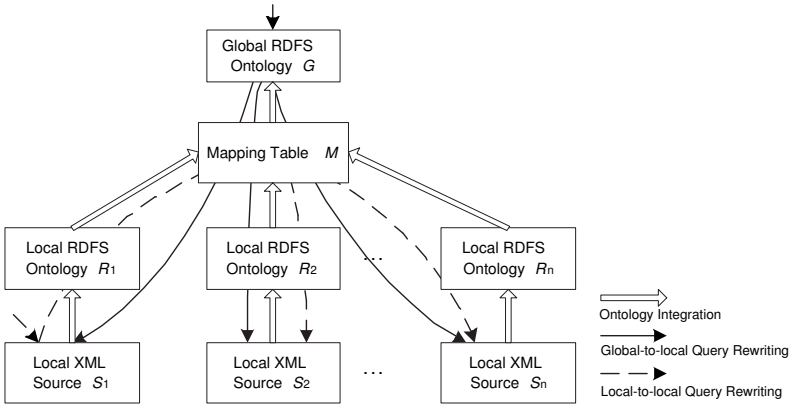

Fig. 2. The ontology-based framework for the integration of XML sources.

and/or relational databases [32]. It focuses on how to take advantage of schema semantics to generate the consistent translations from source to target by considering the constraints and structure of the target schema. It uses queries to express the mappings from the data to the target schema. The Piazza system is a peer-to-peer system that aims to solve the problem of data interoperation between XML and RDF [19]. The system achieves its interoperation in a low-level (syntactic) way, i.e., through the interoperation of XML and the XML serialization of RDF, whereas we aim to achieve the same objective at the semantic level. For example, our approach supports a conceptual view of XML sources (to facilitate the formulation of queries) and allows for conceptual queries (e.g., RDF queries).

3 Framework

In this section, we present the framework for the integration of XML data sources and in particular we describe the integration of XML source schemas and query processing in the integrated system.

As shown in Figure 2, we generate for each local XML source a local RDFS ontology, which represents the source schema. These local RDFS ontologies are then merged into the global RDFS ontology, which provides an overview of all the local ontologies and a mediation between each pair of XML sources. In this merging process, a mapping table is also produced to contain all the mappings, which are correspondences between the global ontology and local ontologies.

The ontology-based XML data integration framework $\mathcal{I}$ can be formalized as a quadruple $\langle \mathcal{G}, \mathcal{S}, \mu, \mathcal{M} \rangle$, where

- $\mathcal{G}$ is the global ontology expressed in RDFS over the alphabet $\mathcal{A}_\mathcal{G}$. The alphabet comprises the name of the classes and properties of $\mathcal{G}$.
- $\mathcal{S}$ is the XML source schema expressed in a language $\mathcal{L}_\mathcal{S}$ over the alphabet $\mathcal{A}_\mathcal{S}$, which comprises the XML element names in $\mathcal{S}$.

- μ is a schema transformation function, which generates a local RDFS ontology $\mathcal{R}$ for $\mathcal{S}$, such that $\mathcal{R}$ encodes the nesting structure specified by $\mathcal{S}$.
- $\mathcal{M}$ is the mapping table consisting of a set of mappings between the global ontology $\mathcal{G}$ and a set of n XML sources $\mathcal{S}_i$, where $i \in [1..n]$. Each entry in $\mathcal{M}$ is of the form $(g, s_1, ..., s_n)$, where $g \in \mathcal{A}_\mathcal{G}$ and $s_i \in \mathcal{A}_{\mathcal{S}_i} \cup \{\epsilon\}$ for $i \in [1..n]$. Note that ϵ is used when a source schema has no corresponding elements to an element of $\mathcal{G}$.

3.1 Integration of XML source schemas

The first task of the framework is the integration of the distributed and heterogeneous XML sources. Here, we are mainly concerned with the issue of schematic heterogeneity, that is, with the different schema structures among the sources. The process of data integration contains two steps: *schema transformation* and *ontology merging*.

In the first step, we use a local RDFS ontology to represent each XML source schema so as to achieve a uniform representation for the next step. In other words, the schema transformation function μ takes as input the source schema $\mathcal{S}$, and the output is the local ontology $\mathcal{R}$. The key operation in this schema transformation is the preservation of the nesting structure of $\mathcal{S}$. To this end, we have to extend the RDFS space since it does not have a property to encode the nesting structure between elements. In particular, we add a new RDF property, `contained`, in the namespace of "http://www.example.org/rdf-extension" (abbreviated as `rdfx`), The RDF/XML syntax for this property is described below.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdfx="http://www.example.org/rdf-extension#">
<rdf:Property rdf:about=
        "http://www.example.org/rdf-extension#contained">
  <rdfs:isDefinedBy rdf:resource=
        "http://www.example.org/rdf-extension#"/>
  <rdfs:label>contained</rdfs:label>
  <rdfs:comment> The containment between two classes.
  </rdfs:comment>
  <rdfs:range rdf:resource=
        "http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:domain rdf:resource=
        "http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Property>
```

The second step is the merging (or integration) of all local ontologies, which generates the global ontology as well as the mapping table. The merging is performed based on the semantics of classes and properties from each of the local ontologies. In particular, the classes or properties that have similar or same (equivalent) semantics are merged into a class or a property of the global ontology. Then, each of these correspondences are recorded as an entry in the mapping table. Different kinds of mappings can

be established between two schemas or ontologies [38]. For this paper, however, we consider only the *equivalence* type of mapping. We also do not consider the different degrees to which two concepts may be equivalent. For instance, we simply take `book` and `article` as equivalent concepts, although we could further refine such equivalence. Additional domain-related knowledge (e.g., inheritance) may be considered. We discuss these issues in more detail in Section 4.

It is worth mentioning that the global ontology in our system has two roles: (1) It provides the user with access to the data with a uniform query interface to facilitate the formulation of a query on all the XML sources; (2) It serves as the mediation mechanism for accessing the distributed data through any of the XML sources.

3.2 Query processing

Our framework handles user queries using a query rewriting strategy. More specifically, query processing in our framework may occur in the following two directions, as shown in Figure 2:

Global-to-local query rewriting. When the user poses a query q on the global ontology, the system rewrites q into the union q' of subqueries, one for each XML source. The subqueries are then executed over the XML sources to get the answers, which are then integrated (by using union) to produce the answer to q.

Local-to-local query rewriting. Given a query q posed on a local source, its answers then include not only those retrieved from the local source, but also those from all the other sources in the system. For the purpose of getting answers from the other sources, it requires that q be rewritten (through the global ontology) into a union q' of queries, one on each of the other sources. Query rewriting in this direction is performed similarly to that in peer-to-peer systems [33].

Query rewriting in both directions is based on the mapping information contained in the mapping table. Each entry contains a element (RDF class or property) of the global ontology and its corresponding elements in the local source schemas. Given that query rewriting is from a query over one alphabet to that over another alphabet, the mapping table provides a convenient way to finding the mapping between alphabets, in both rewriting directions. In addition, the query languages used to formulate the queries have to be taken into consideration, since they may have different expressiveness. We consider a subset of XQuery [7], called *conjunctive XQuery* (*c-XQuery*), for queries over the XML sources and a subset of RDQL [20], namely *conjunctive RDQL* (*c-RDQL*), for queries over the global RDFS ontology. We discuss in detail query processing and related issues in Section 5.

4 Integrating Structure and Semantics

4.1 Local XML schemas and local RDFS ontologies

To integrate heterogeneous XML data sources, we first transform the local XML schema into a local RDFS ontology while preserving the XML document structure. By *document structure*, we mean the structural relationship of objects specified in *data-centric*

documents [8] by a schema language (such as DTD, XML Schema, or RelaxNG[1]). In this paper, we only focus on the nesting structure (i.e., hierarchy). Other structural properties include order. A consequence of not including order in our framework is that we cannot consider a query that involves the order of the subelements of an element. However, this kind of query is of little interest in a framework where we are mostly concerned with the semantics of the data.

Elements and attributes are the two basic building blocks of XML documents. Elements can be defined as *simple types*, which cannot have element content and cannot carry attributes, or *complex types*, which allow elements in their content and/or contain attributes. On the other hand, all attribute declarations must reference simple types since attributes cannot contain other elements or other attributes. From the perspective of XML Schema, these nesting relationships are defined in terms of *datatypes* (simple or complex). An XML schema can be formalized as an edge-labeled tree, namely an *XML schema tree*, as depicted in Figure 1. We overlook the distinction between XML elements and attributes by considering both of them as vertices in the XML schema tree.

Definition 1. *An XML schema $\mathcal{S}$ over alphabet $\mathcal{A}_\mathcal{S}$ is an edge-labeled tree $\mathcal{S} = (V, E, \lambda)$, where V is a set of vertices, $E = \{(v_i, v_j)|v_i, v_j \in V\}$ is a set of edges, and λ is a labeling function $\lambda : E \mapsto \mathcal{A}_\mathcal{S}$.*

Before we discuss schema transformation, let us look at the formalization of ontologies. Both the global ontology and local ontologies are actually RDF schemas defined in the RDFS space, which is extended with the RDF property "`rdfx:contained`". An RDF schema can be formalized as a labeled graph, called *RDF schema graph*, as defined in Definition 2. We do not elaborate on the data types of RDF properties and assume that they are all of type *literal*. Also, we do not take into account the notion of namespace in the definition of both XML and RDF schemas.

Definition 2. *An RDF schema graph $\mathcal{R}$ over alphabet $\mathcal{A}_\mathcal{R}$ is a directed labeled graph $\mathcal{R} = (V, E, \lambda)$, where V is a set of labeled vertices consisting of classes C, properties P, and data types L, $E = \{(v_i, v_j)|v_i, v_j \in V\}$ is a set of labeled edges, and λ is a labeling function $\lambda : V \cup E \mapsto \mathcal{A}_\mathcal{R}$, such that*

- *$\forall\, v \in P$, we have $domain(v) \in C$, $range(v) \in C \cup L$, and $\lambda((v, domain(v)))$ = "rdfs:domain" and $\lambda((v, range(v)))$="rdfs:range";*
- *$\forall\, e = (v_i, v_j) \in E$, we have $\lambda(e)$="rdfs:subClassOf" (or "rdfx:contained") if v_i and $v_j \in C$, or $\lambda(e)$ = "rdfs:subPropertyOf" if v_i and $v_j \in P$.*

Now we are able to define the schema transformation function μ. Formally speaking, the *schema transformation function* μ is a function $\mu : \mathcal{S} \mapsto \mathcal{R}$, where $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S}, \lambda_\mathcal{S})$, $\mathcal{R} = (V_\mathcal{R}, E_\mathcal{R}, \lambda_\mathcal{R})$, and $V_\mathcal{R} = C \cup P$, such that $\forall\, e_{ij} = (v_i, v_j) \in E_\mathcal{S}$, we have $\mu(v_j) \in V_\mathcal{R}$, $\lambda_\mathcal{R}(\mu(v_j)) = \lambda_\mathcal{S}(e_{ij})$, and furthermore:

(1) if $\exists(v_j, v_k) \in E_\mathcal{S}$, then $\mu(v_j) \in C$, $(\mu(v_j), \mu(v_i)) \in E_\mathcal{R}$, and $\lambda_\mathcal{R}(\mu(v_j), \mu(v_i))$ = "rdfx:contained";

[1] http://relaxng.sourceforge.net

(2) if $\nexists(v_j, v_k) \in E_{\mathcal{S}}$, then $\mu(v_j) \in P$, $(\mu(v_j), \mu(v_i)) \in E_{\mathcal{R}}$, and $\lambda_{\mathcal{R}}(\mu(v_j), \mu(v_i))$ = *"rdfs:domain"*.

The transformations thus defined fall into two categories:

Element-level transformation The element-level transformation converts from XML complex-type elements to RDF classes and from XML simple-type elements to properties. For example, for $\mathcal{S}_1$ in Example 1, we define the RDF classes `Books`, `Book`, and `Author`, while taking `booktitle` and `name` as RDF properties of `Book` and `Author`, respectively, as depicted in the resulting local RDFS ontology of Figure 3.

Structure-level transformation The structure-level transformation encodes the nesting structure of the XML schema into the local RDFS ontology. In particular, the nesting may occur between two complex-type elements or between a complex-type element and its child (simple) element. Following the element-level transformation, the nesting structure in the former case corresponds to a *class-to-class* relationship between two RDFS classes, which are connected by the property `rdfx:contained`, The first item that defines μ formalizes this case. In the latter case, the XML nesting structure corresponds to the *class-to-literal* relationship in the local ontology, with the class and the literal connected by the corresponding property. The second item that defines μ formalizes this case.

By applying the schema transformation function to the two XML schemas in Figure 1, we can get the resulting local ontologies as shown in Figure 3. We see that `rdfx:contained` enables the representation of the nesting relationship. Specifically, by following the edges of `rdfx:contained` from `Books` to `Author` in $\mathcal{R}_1$, we actually get the corresponding path `/books/book/author` in $\mathcal{S}_1$. In terms of the alphabets, the schema transformation function specifies a mapping between the alphabet of the source schema and that of the local ontology. Table 1 lists the mapping between the XML schema $\mathcal{S}_1$ and the local RDFS ontology $\mathcal{R}_1$. For simplicity, we use XPath to specify the XML elements. Also, the properties in the mapping table are in the form of an RDF expression $c.p$, where c is the class associated with p.

Table 1. Mappings between $\mathcal{S}_1$ and $\mathcal{R}_1$

XPath expressions in $\mathcal{S}_1$	**RDF expressions in $\mathcal{R}_1$**
/books	Books
/books/book	Book
/books/book/booktitle	Book.booktitle
/books/book/author	Author
/books/book/author/name	Author.name

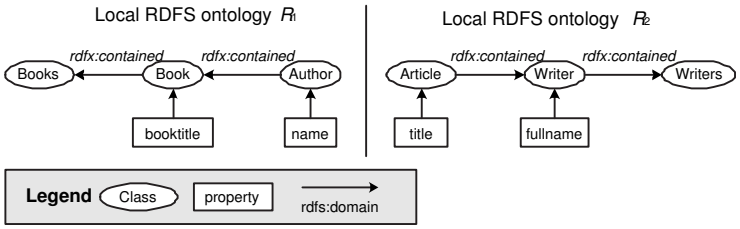

Fig. 3. Local RDFS ontologies transformed from $\mathcal{S}_1$ and $\mathcal{S}_2$.

4.2 Global RDFS ontology

Now that the source schemas are represented by local RDFS ontologies, we are able to merge them to construct the global RDFS ontology. In other words, the process of ontology merging takes as input the multiple local ontologies and returns a merged ontology as the output [35].

Ontology merging and ontology alignment, which require the mapping of ontologies, are widely pursued research topics. Readers can be referred to a thorough survey of the state-of-the-art of ontology mapping [21]. In this paper we do not intend to introduce a new technique for ontology merging. Instead, we utilize existing techniques to generate the integrated ontology from the local ontologies. In particular, we use an approach (such as PROMPT [29]) that provides the following functionalities:

- *Merging of classes*: Multiple conceptually equivalent classes of the local ontologies are combined into one class in the global ontology.
- *Merging of properties*: Multiple conceptually equivalent properties of the equivalent classes in the local ontologies are combined as one property of the combined class in the global ontology.
- *Merging relationships between classes*: Given two conceptually equivalent relationships, e.g., p_1 from a class c_1 to another class c_1' and p_2 from c_2 to c_2', we combine p_1 and p_2 into one relationship p between the combined class c (of c_1 and c_2) and c' (of c_1' and c_2').
- *Copying a class or a property*: If there does not exist a conceptually equivalent class or property for a class c (or a property p of c), we simply copy c (or p, as a property of the target class of c) into the global ontology.
- *Generalizing semantically related classes into a superclass:* The superclass can be obtained by searching an existing knowledge domain (e.g., the DAML Ontology Library) or reasoning over a thesaurus such as WordNet.[2] For example, we can find in the semantic network of terms (consisting of terms and their semantic relations) that two classes (`Author` and `Writer`) have the same hypernym (`Person`), which is then taken as a superclass of both classes.

Figure 4 shows the global ontology that results from merging the two local RDF ontologies of Figure 3. The greyed classes and properties are merged classes and properties from the original ontologies. For instance, `Book` in $\mathcal{R}_1$ and `Article` in $\mathcal{R}_2$

[2] http://wordnet.princeton.edu

are merged into `Book`, whereas `booktitle` in $\mathcal{R}_1$ and `title` in $\mathcal{R}_2$ are merged into `title`. The classes `Book` and `Author` are also respectively extended with the superclasses `Publication` and `Person`.

Besides the global ontology, the process of ontology merging also yields as an output the mapping table that contains the mappings between the local RDFS ontologies and the global RDFS ontology. In general, if a class, property, or relationship between classes p in the global ontology is the result of merging p_i and p_j from different local ontologies, then a tuple of the form (p, p_i, p_j) is generated. If a class or property p in the global ontology is only copied from p_i in a local ontology, then a tuple (p, p_i) is produced. For instance, for the class `Book.title` (in the global ontology), which is merged from `Book.booktitle` in $\mathcal{R}_1$ and `Article.title` in $\mathcal{R}_2$, we generate a tuple in the mapping table: (`Book.title`, `Book.booktitle`, `Article.title`). Table 2 lists all the mappings in our example.

Table 2. Mapping table between the global ontology and local RDF ontologies

RDF expressions in the global ontology	**RDF expressions in $\mathcal{R}_1$**	**RDF expressions in $\mathcal{R}_2$**
Books	Books	-
Book	Book	Article
Book.title	Book.booktitle	Article.title
Authors	-	Writers
Author	Author	Writer
Author.name	Author.name	Writer.fullname

Now that we have the one-to-one mappings $\mathcal{M}_1$ between the XML source schemas and their local ontologies and the one-to-one mappings $\mathcal{M}_2$ between the local ontologies and the global ontology, we can compose $\mathcal{M}_1$ and $\mathcal{M}_2$ to get the mappings $\mathcal{M}$ between the source schemas and the global ontology. Table 3 shows the results.

4.3 Data integration semantics

In this subsection, we discuss the semantics of the data integration in our proposed framework including the semantics of the XML (local) databases, the mapping table,

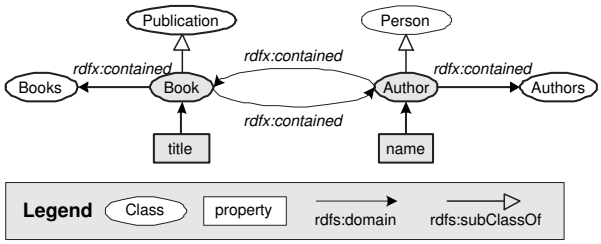

Fig. 4. The global ontology $\mathcal{G}$ that results from merging $\mathcal{R}_1$ and $\mathcal{R}_2$.

Table 3. Mapping table between the global ontology and XML source schemas

RDF expressions in the global ontology	**XPath expressions in $\mathcal{S}_1$**	**XPath expressions in $\mathcal{S}_2$**
Books	/books	-
Book	/books/book	/writers/writer/article
Book.title	/books/book/booktitle	/writers/writer/article/title
Authors	-	/writers
Author	/books/book/author	/writers/writer
Author.name	/books/book/author/name	/writers/writer/article/fullname

and the RDFS (global) database. The discussion of the syntax and semantics of queries is postponed until Section 5. In what follows, we refer to a fixed, finite set Γ of constants, which is shared by all data sources. We also refer to a finite set U of URIs.

There are two types of databases in the framework, i.e., the local XML databases and the global RDF database. An XML database is an *XML instance tree*, and an RDF database is an *RDF instance graph*.

Definition 3 (XML instance tree). *Given an XML schema $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S}, \lambda_\mathcal{S})$, an instance of $\mathcal{S}$ is an XML instance tree $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G}, \tau, \lambda_\mathcal{G})$, where $V_\mathcal{G}$ is a set of vertices, $E_\mathcal{G}$ is a set of edges, and*

(1) τ is a typing function $\tau : V_\mathcal{G} \mapsto V_\mathcal{S}$, such that (a) $\forall v \in V_\mathcal{G}$, $\tau(v) \in V_\mathcal{S}$, and (b) $\forall (v_i, v_j) \in E_\mathcal{G}$, $(\tau(v_i), \tau(v_j)) \in E_\mathcal{S}$.
(2) $\lambda_\mathcal{G}$ is a labeling function, such that (a) $\forall v \in V_\mathcal{G}$, $\lambda_\mathcal{G}(v) \in \Gamma \cup \{\epsilon\}$, and (b) $\forall (v_i, v_j) \in E_\mathcal{G}$, $\lambda_\mathcal{G}((v_i, v_j)) = \lambda_\mathcal{S}((\tau(v_i), \tau(v_j)))$.

Definition 4 (RDF instance graph). *Given an RDF schema $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S}, \lambda_\mathcal{S})$, where $V_\mathcal{S} = C \cup P$, an instance of $\mathcal{S}$ is an RDF instance graph $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G}, \tau, \lambda_\mathcal{G})$, where $V_\mathcal{G}$ is a set of vertices, $E_\mathcal{G}$ is a set of edges, $\lambda_\mathcal{G}$ is a labeling function $\lambda_\mathcal{G} : V_\mathcal{G} \cup E_\mathcal{G} \mapsto \mathcal{A}_\mathcal{S} \cup U \cup \Gamma$, and τ is a typing function $\tau : V_\mathcal{G} \cup E_\mathcal{G} \mapsto V_\mathcal{S} \cup \{$"rdf:Property"$\} \cup \{$"rdfs:literal"$\}$, such that $\forall e = (v_i, v_j) \in E_\mathcal{G}$, we have*

(1) if $\tau(e)$="rdf:Property", then $\lambda_\mathcal{G}(e)$="rdfx:contained" or "rdfs:subClassOf", $\lambda_\mathcal{G}(v_i)$ and $\lambda_\mathcal{G}(v_j) \in U$, $\tau(v_i)$ and $\tau(v_j) \in C$, and $(\tau(v_i), \tau(v_j)) \in E_\mathcal{S}$;
(2) if $\tau(e) \in P$, then $\lambda_\mathcal{G}(e) = \lambda_\mathcal{S}(\tau(e))$, $\lambda_\mathcal{G}(v_i) \in U$, $\tau(v_i) \in C$, $\lambda_\mathcal{S}((\tau(e), \tau(v_i)))$ = "rdfs:domain", $\lambda_\mathcal{S}((\tau(e), \tau(v_j)))$="rdfs:range", and
- *$\lambda_\mathcal{G}(v_j) \in U$, when $\tau(v_j) \in C$;*
- *$\lambda_\mathcal{G}(v_j) \in \Gamma$, when $\tau(v_j)$="rdfs:literal";*

The semantics of the mappings depends on the assumptions adopted. In the view-based approach, there are three assumptions for the inter-schema mappings, namely *soundness*, *completeness*, and *exactness* [25]. In particular, given a database $\mathcal{D}$, a set of view definitions $\mathcal{V}$ over $\mathcal{D}$, and view extensions $\mathcal{E}$ of $\mathcal{V}$, we say the views $\mathcal{V}$ are *sound* if $\mathcal{V}^\mathcal{D} \supseteq \mathcal{E}$, *complete* if $\mathcal{V}^\mathcal{D} \subseteq \mathcal{E}$, and *exact* if $\mathcal{V}^\mathcal{D} = \mathcal{E}$. It is common to use the soundness assumption for view-based data integration [25]. Given that our framework adopts a GaV approach, it is natural to assume an exact semantics, that is, the sources

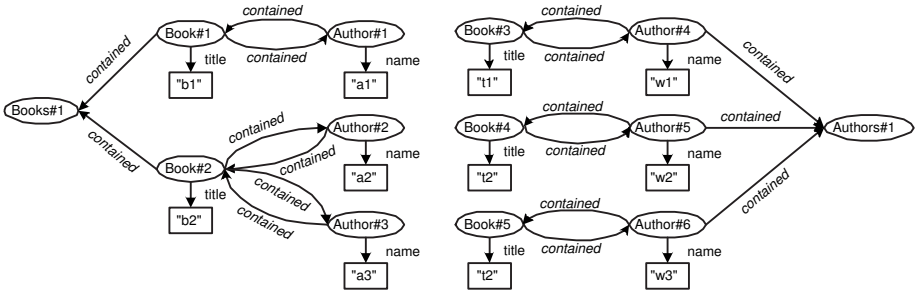

Fig. 5. The global database of $\mathcal{G}$.

are complete with respect to the global database. However, the definition for these assumptions differs from our framework, where mappings are represented by element correspondences in the mapping table.

Given an entry $t_i = (g_i, s_{i,1}, ..., s_{i,n})$ in the mapping table $\mathcal{M}(\mathcal{G}, \mathcal{S}_1, ..., \mathcal{S}_n)$, where $g_i \in \mathcal{G}$ and $s_{i,j} \in \mathcal{S}_j$ $(1 \leq j \leq n)$, the semantics of the mappings can be captured by the concept of valuation. Given the global database $\mathcal{B}$ of $\mathcal{G}$ and local databases $\mathcal{D}_j$ of $\mathcal{S}_j$ $(1 \leq j \leq n)$, a *valuation* of t_i is a function σ, which maps t_i to a tuple $(v_i, v_{i,1}, ..., v_{i,n})$, where $v_i \in \mathcal{B}$, and $v_{i,j} \in \mathcal{D}_j$ $(1 \leq j \leq n)$, such that $\tau_{\mathcal{B}}(v_i) = g_i$ and $\tau_{\mathcal{D}_j}(v_{i,j}) = s_{i,j}$ for $j \in [1..n]$. Under the exact assumption, the semantics of the mapping table $\mathcal{M} = \{t_1, ..., t_m\}$ is captured by a conjunction of all the equalities (between the valuation of each global element and the union of the valuations of its mapped local elements), that is:

$\bigwedge_{1 \leq i \leq m}[\sigma(g_i) = \sigma(s_{i,1}) \cup ... \cup \sigma(s_{i,n})]$, such that for $1 \leq k, l \leq m$,
(1) $(g_k, g_l) \in E_{\mathcal{G}} \Leftrightarrow (\sigma(g_k), \sigma(g_l)) \in E_{\mathcal{B}}$, and
(2) $(s_{k,j}, s_{k,l}) \in E_{\mathcal{S}_k} \Leftrightarrow (\sigma(s_{k,j}), \sigma(s_{k,l})) \in E_{\mathcal{D}_k}$, for each $j \in [1..n]$.

The definition of the semantics of sound (or complete) mappings is the same as the above definition, except for the substitution of $=$ by $\supseteq$ (or $\subseteq$). For simplicity, we abbreviate the preceding assertion to $\sigma(\mathcal{G}) = \sigma(\mathcal{S}_1) \cup ... \cup \sigma(\mathcal{S}_n)$. The *global database* $\mathcal{B}$ is then any database such that $\sigma(\mathcal{G}) = \sigma(\mathcal{S}_1) \cup ... \cup \sigma(\mathcal{S}_n)$ holds for the local databases $\mathcal{D}_1, ..., \mathcal{D}_n$. Figure 5 shows the global database (instances) for the data sources of Example 1.

5 Query Processing

5.1 Query languages

RDQL (RDF Data Query Language) uses an SQL-like syntax. More specifically, the `Select` clause identifies the variables to be returned to the application. The `From` clause specifies the RDF model using an URI. The `Where` clause specifies the graph pattern as a list of triple patterns. The `And` clause specifies the Boolean expressions.

Finally, the `Using` clause provides a way to shorten the length of the URIs. By overlooking the notion of namespace (i.e., URI) and the `And` clause, we get a *conjunctive RDQL* (c-RDQL) expression, which can be expressed in a conjunctive formula:

$$ans(\boldsymbol{X}) \text{ :- } p_1(\boldsymbol{X_1}), ..., p_n(\boldsymbol{X_n}).$$

where $\boldsymbol{X_i} = (x_i, x_i')$ and p_i is an RDF property of x_i having the value x_i'.

XQuery is a typed functional language that has an FLWR (i.e., `For`, `Let`, `Where`, `Return`) syntax. For simplification, we assume that the XML query posed by the user is formulated only in the form of *FLWR expressions* [7]. In other words, we do not consider nesting FLWR expressions, although they are allowed in XQuery. In particular, a *conjunctive XQuery* (c-XQuery) is of the form:

$$ans(\boldsymbol{X}) \text{ :- } p_1(\boldsymbol{X_1}), ..., p_n(\boldsymbol{X_n}).$$

where $\boldsymbol{X_i} = (x_i, x_i')$ and p_i is an XPath $/e_1/.../e_n$ connecting x_i to x_i'. That is, each predicate represents an expression $x_i/e_1/.../e_n/x_i'$, where $e_i (1 \leq i \leq n)$ is an edge label along the path from x_i to x_i'.

In both query definitions, $ans(\boldsymbol{X})$ is the *head* of the query, denoted $head_q$, and the remaining part is the *body* of the query, denoted $body_q$. We say that the query is *safe* if $\boldsymbol{X} \subseteq \boldsymbol{X_1} \cup ... \cup \boldsymbol{X_n}$.

The answer $q^{\mathcal{D}}$ to a query q over a database $\mathcal{D}$ is the result of evaluating q over $\mathcal{D}$. The query evaluation is based on the concept of *valuation* and depends on the data model and the query language used. Informally, a *valuation* ρ over the variables $var(q)$ of a query q is a total function from $var(q)$ to constants (or URIs for RDF queries) in the domain Γ of the database, where q is evaluated [2], as follows:

- In the XML model: given a c-XQuery q of the form $ans(\boldsymbol{X})$:- $p_1(\boldsymbol{X_1}), ..., p_n(\boldsymbol{X_n})$ over an XML instance graph $\mathcal{D}$, we have

$$q^{\mathcal{D}} = \{\rho(\boldsymbol{X}) | \rho \text{ is a valuation over } var(q) \text{ and } p_i = (\rho(x_i), \rho(x_i')) \text{ is a fact in } \mathcal{D}, \text{ for each } \boldsymbol{X_i} = (x_i, x_i'), \text{ where } i \in [1..n]\}.$$

- In the RDF model: given a c-RQL query q of the form $ans(\boldsymbol{X})$:- $p_1(\boldsymbol{X_1}), ..., p_n(\boldsymbol{X_n})$ over an RDF instance graph $\mathcal{D}$, we have

$$q^{\mathcal{D}} = \{\rho(\boldsymbol{X}) | \rho \text{ is a valuation over } var(q) \text{ and } p_i \text{ is a path connecting } \rho(x_i) \text{ and } \rho(x_i') \text{ in } \mathcal{D}, \text{ for each } \boldsymbol{X_i} = (x_i, x_i'), \text{ where } i \in [1..n]\}.$$

Example 2. Consider two queries q_1 and q_2. In particular, q_1 is expressed over the global ontology $\mathcal{G}$ in c-RDQL, to retrieve all the (`Author`, `Book`) pairs. The c-XQuery query q_2 is issued on local XML source $\mathcal{S}_1$, to retrieve all (`Author`, `Book`) pairs.

q_1: $ans(x, y)$:- $name(u, x)$, $title(v, y)$, $contained(u, v)$.
q_2: $ans(x, y)$:- $/name(u, x)$, $/booktitle(v, y)$, $/author(v, u)$.

By evaluating q_1 over the global database $\mathcal{B}$ (shown in Figure 5) and q_2 over $\mathcal{D}_1$ (shown in Figure 1), we obtain the following answer sets to both queries.

$q_1^{\mathcal{B}} = \{(a1, b1), (a2, b2), (a3, b2), (w1, t1), (w2, t2), (w3, t2)\}$,
$q_2^{\mathcal{D}_1} = \{(a1, b1), (a2, b2), (a3, b2)\}$.

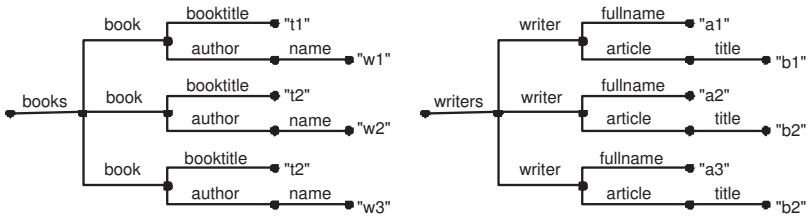

Fig. 6. The retrieved database on $\mathcal{S}_1$ w.r.t. $\mathcal{S}_2$ and that on $\mathcal{S}_2$ w.r.t. $\mathcal{S}_1$.

We finally assume that all the concepts in the local ontologies are mapped to the concepts in the global ontology during the ontology integration process. That is, the mappings are total, one-to-one mappings from the local RDF ontologies to the global ontology. However, it is possible that some concept c or property p in the global ontology gets mapped to a local ontology but not to another local ontology. This may lead to null values when a query involves c or p. However, we do not consider this case in our discussion.

5.2 Certain answers and query containment

The concept of *certain answers* has been introduced in view-based query processing to represent the results of answering a global query (the query over the global schema) using view extensions [1]. In our framework, where the mappings are correspondences between elements of the global ontology and elements of the source schemas, the concept of *certain answers* is redefined. We call the query posed on the global ontology a *global query*, and the query posed over a local data source a *local query*. As previously discussed, these two queries are processed in two different directions, i.e., the global-to-local direction and the local-to-local direction. The certain answers to a global query are called *global certain answers*, while those to a local query are called *local certain answers*.

Before we discuss the formalism for these two types of certain answers, we revisit the concept of *global database*, from which we retrieve the global certain answers, and we introduce the concept of *retrieved database*, where the local certain answers are computed.

Given the local data sources $\mathcal{D}_1, ..., \mathcal{D}_n$ and the mapping table $\mathcal{M}(\mathcal{G}, \mathcal{S}_1, ..., \mathcal{S}_n)$ between the global ontology $\mathcal{G}$ and local source schemas $\mathcal{S}_1, ..., \mathcal{S}_n$. The *global database* $\mathcal{B}$ is such that $\sigma(\mathcal{G}) = \bigcup_{(1 \leq i \leq n)} \sigma(\mathcal{S}_i)$ holds on $\mathcal{D}_1, ..., \mathcal{D}_n$. Likely, the *retrieved database* $\mathcal{B}_k$ on a local source $\mathcal{S}_k$ w.r.t. all the other local sources is the one satisfying $\sigma(\mathcal{S}_k) = \bigcup_{(1 \leq i \leq n, i \neq k)} \sigma(\mathcal{S}_i)$, whereas, the retrieved database $\mathcal{B}_{k,l}$ on $\mathcal{S}_k$ w.r.t. a particular local source $\mathcal{S}_l$ is the one satisfying $\sigma(\mathcal{S}_k) = \sigma(\mathcal{S}_l)$ (refer to Section 4 for the semantics of σ). Figure 6 shows an example of the retrieved database on $\mathcal{S}_1$ w.r.t. $\mathcal{S}_2$ (on the left side) and the one on $\mathcal{S}_2$ w.r.t. $\mathcal{S}_1$ (on the right side), for $\mathcal{S}_1$ and $\mathcal{S}_2$ as presented in Figure 1.

Based on the concept of global database and that of retrieved database, we formally define both types of certain answers next.

Definition 5 (Certain answers). *Let $\mathcal{G}$ be the global ontology of n XML source schemas $\mathcal{S}_1, ..., \mathcal{S}_n$ respectively with databases $\mathcal{D}_1, ..., \mathcal{D}_n$, $\mathcal{M}$ be the mapping table, q be a global query posed over $\mathcal{G}$, and q_k be a local query on $\mathcal{S}_k$. The **global certain answers** to q with respect to $\mathcal{D}_1, ..., \mathcal{D}_n$ based on $\mathcal{M}$ are the results of evaluating q over the global database $\mathcal{B}$, denoted $cert_{\mathcal{M}}(q) = q^{\mathcal{B}}$. The **local certain answers** to q_k with respect to $\mathcal{D}_1, ..., \mathcal{D}_{k-1}, \mathcal{D}_{k+1}, ..., \mathcal{D}_n$ based on $\mathcal{M}$ are computed by evaluating q_k over the retrieved database $\mathcal{B}_k$ on $\mathcal{S}_k$, denoted $cert_{\mathcal{M},k}(q_k) = q^{\mathcal{B}_k}$.*

While the global certain answers constitute the answer to a global query, the answer to a local query q_k contains both the local certain answers and those retrieved from the local database $\mathcal{D}_k$, that is, $ans(q_k) = cert_{\mathcal{M},k}(q_k) \cup q^{\mathcal{D}_k}$.

Query containment is a fundamental problem in database research. In general, query containment checks whether two queries are contained in each other. This problem has been studied in the following three cases.

The first case is query containment in a single database $\mathcal{D}$, over which the two queries are posed, that is, $\mathcal{D}_1 = \mathcal{D}_2 = \mathcal{D}$. Given a single database schema $\mathcal{S}$ over which q_1 and q_2 are posed, we say q_1 is *contained* in q_2, denoted $q_1 \subseteq q_2$, if they have the same output schema and $q_1^{\mathcal{D}} \subseteq q_2^{\mathcal{D}}$ for every database $\mathcal{D}$ of $\mathcal{S}$. The two queries q_1 and q_2 are said to be *equivalent*, denoted $q_1 \equiv q_2$, if $q_1^{\mathcal{D}} \subseteq q_2^{\mathcal{D}}$ and $q_2^{\mathcal{D}} \subseteq q_1^{\mathcal{D}}$ [2].

The second case is query containment in data integration systems, where both queries are posed over the global database. The data sources are usually *homogeneous* in the sense that the same syntax is used. Given that the sources are expressed as views over the global database, two queries are said to be equivalent relative to the same set of data sources, if for any source databases they have the same set of certain answers. The query containment problem in this case is called *relative query containment* [28].

The third case is also in homogeneous data integration systems, where data sources are defined as views of the global schema, but the two queries are formulated in terms of different alphabets. In particular, there are two kinds of queries, i.e., the queries q^{Σ} over the alphabet Σ of the global schema and the queries $q^{\mathcal{V}}$ over the alphabet $\mathcal{V}$ of the view definitions. The query containment in this case is called *view-based containment* and is discussed for different situations such as containment between q_1^{Σ} and q_2^{Σ}, between q_1^{Σ} and $q_2^{\mathcal{V}}$, between $q_1^{\mathcal{V}}$ and q_2^{Σ}, and between $q_1^{\mathcal{V}}$ and $q_2^{\mathcal{V}}$ [13].

In our case, we are interested in two kinds of containment, specifically the containment between a global query q and a union of local queries $q_1, ..., q_n$, and the containment between two local queries q_k and q_l. The first kind of containment, which we call *global query containment*, is the same as the containment between q_1^{Σ} and $q_2^{\mathcal{V}}$. Whereas the second kind differs from the containment between $q_1^{\mathcal{V}}$ and $q_2^{\mathcal{V}}$, in the sense that q_k and q_l refer to different alphabets but $q_1^{\mathcal{V}}$ and $q_2^{\mathcal{V}}$ are expressed over the same alphabet. We call the containment between q_k and q_l *P2P query containment*, because of its likeness to query processing in a P2P system. Next we give the formal definitions for these two containments in our framework.

Definition 6 (Global query containment). *Let $\mathcal{G}$ be the global ontology over n XML source schemas $\mathcal{S}_1, ..., \mathcal{S}_n$, $\mathcal{M}$ be the mapping table, q be a global query posed over $\mathcal{G}$, and q' be a union of local queries $q_1, ..., q_n$ respectively over $\mathcal{S}_1, ..., \mathcal{S}_n$. We say q is **globally contained** in q', denoted $q \subseteq_{\mathcal{M}} q'$, if for any databases $\mathcal{D}_1, ..., \mathcal{D}_n$, we have*

$cert_{\mathcal{M}}(q) \subseteq q_1^{\mathcal{D}_1} \cup ... \cup q_n^{\mathcal{D}_n}$. *We say* q *and* q' *are* ***globally equivalent****, denoted* $q \equiv_{\mathcal{M}} q'$*, if* $q \subseteq_{\mathcal{M}} q'$ *and* $q \supseteq_{\mathcal{M}} q'$.

Definition 7 (P2P query containment). *Let* $\mathcal{G}$ *be the global ontology over* n *XML source schemas* $\mathcal{S}_1, ..., \mathcal{S}_n$*,* $\mathcal{M}$ *be the mapping table,* q_i *be a local query posed over* $\mathcal{S}_i$*, and* q_j *be a local query over* $\mathcal{S}_j$*. We say* q_i *is* ***P2P contained*** *in* q_j*, denoted* $q_i \subseteq_{\mathcal{M}} q_j$*, if for any databases* $\mathcal{D}_1, ..., \mathcal{D}_n$*, we have* $cert_{\mathcal{M},i}(q_i) \cup q_i^{\mathcal{D}_i} \subseteq cert_{\mathcal{M},j}(q_j) \cup q_j^{\mathcal{D}_j}$*. We say* q *and* q' *are* ***P2P equivalent****, denoted* $q_i \equiv_{\mathcal{M}} q_j$*, if* $q_i \subseteq_{\mathcal{M}} q_j$ *and* $q_i \supseteq_{\mathcal{M}} q_j$.

Example 3. Consider the following three queries q, q_1, and q_2 respectively on the global ontology $\mathcal{G}$, local XML source $\mathcal{S}_1$, and local XML source $\mathcal{S}_2$. Also consider the mapping table $\mathcal{M}$ shown in Table 3.

q: $ans(x, y)$:- $name(u, x)$, $title(v, y)$, $contained(u, v)$.
q_1: $ans(x, y)$:- $/name(u, x)$, $/booktitle(v, y)$, $/author(v, u)$.
q_2: $ans(x, y)$:- $/fullname(u, x)$, $/title(v, y)$, $/article(u, v)$.

By executing q on the global database $\mathcal{B}$, q_1 on $\mathcal{D}_1$ and on the retrieved database $\mathcal{B}_1$, and q_2 on $\mathcal{D}_2$ and on the retrieved database $\mathcal{B}_2$, we obtain the following answers to the three queries.

$cert_{\mathcal{M}}(q) = q^{\mathcal{B}}$: $\{(a1, b1), (a2, b2), (a3, b2), (w1, t1), (w2, t2), (w3, t2)\}$
$q_1^{\mathcal{D}_1}$: $\{(a1, b1), (a2, b2), (a3, b2)\}$
$cert_{\mathcal{M},1}(q_1) = q_1^{\mathcal{B}_1}$: $\{(w1, t1), (w2, t2), (w3, t2)\}$
$q_2^{\mathcal{D}_2}$: $\{(w1, t1), (w2, t2), (w3, t2)\}$
$cert_{\mathcal{M},2}(q_2) = q_2^{\mathcal{B}_2}$: $\{(a1, b1), (a2, b2), (a3, b2)\}$

Therefore, by Definition 6 and Definition 7, we have $q \equiv_{\mathcal{M}} (q_1 \cup q_2)$ and $q_1 \equiv_{\mathcal{M}} q_2$.

5.3 Query rewriting

In a data integration system where the sources are described as views over the global schema, query processing is called *view-based query processing*, which has two approaches, i.e., *view-based query answering* and *view-based query rewriting* [12, 18]. Likewise, there are two approaches to answering a query in our framework, where mappings are expressed by correspondences. The first approach utilizes the notion of (global or local) certain answers, as previously discussed.

The alternative approach is by query rewriting. Specifically, to answer a global (or local) query q, the query is rewritten into a union of the queries over all the sources, using the mappings. The integration of the answers retrieved from each source constitutes the answer to q.

As mentioned before, there are two directions of query processing in our framework. We expect that query rewriting in both directions is equivalent, in the sense that the rewriting is *globally* (or *P2P*) *equivalent* to the original query. We present next two query rewriting algorithms, i.e., GLREWRITING for global-to-local query rewriting and LLREWRITING for local-to-local rewriting, which will ensure the equivalence of the rewritten queries.

Algorithm GLREWRITING
Input:1. q_1 over the global ontology $\mathcal{G}$: $ans(\boldsymbol{X})$:- $p_1(\boldsymbol{X_1}), ..., p_m(\boldsymbol{X_m})$;
2. $\mathcal{M}$ between the global ontology $\mathcal{G}$ and local XML schemas $\mathcal{S}_1, ..., \mathcal{S}_n$.
Output: q_2: Union of the c-XQueries over $\mathcal{S}_1, ..., \mathcal{S}_n$.
1. $q_2 = null$;
2. **For** $i = 1$ **to** n **do**
3. $head_q = head_{q_1}$; $body_q = null$;
4. **For** $j = 1$ **to** m **do**
5. (c_1, c_2) = name of the class/property bound to (x_1, x_2), for $\boldsymbol{X_j} = (x_1, x_2)$;
6. Search $\mathcal{M}$ to find (d_1, d_2) such that $\{(c_1, d_1), (c_2, d_2)\} \subseteq \pi_{\mathcal{G},\mathcal{S}_j}(\mathcal{M})$;
7. **If** a path p exists from d_1 to d_2 in $\mathcal{S}_j$ **then**
8. add $p(x_1, x_2)$ to $body_q$;
9. **Else if** a path p exists from d_2 to d_1 in $\mathcal{S}_j$ **then**
10. add $p(x_2, x_1)$ to $body_q$;
11. **Else** add $p(\hat{x}, x_1)$ and $p'(\hat{x}, x_2)$ to $body_q$, where $\hat{x}$ is a new variable bound to the lowest ancestor d of d_1 and d_2, and p (p') is the path from d to $d_1(d_2)$;
12. $q_2 = q_2 \cup q$;

We see that the algorithm GLREWRITING adopts a strategy similar to the "unfolding" strategy used by query processing in a GaV-based relational data integration system [25]. However, instead of substituting the predicates in a query q with the corresponding views, the substitution of predicates in GLREWRITING is guided by the correspondences in the mapping table $\mathcal{M}$, as stated in Lines 5 to 11. The calculation of the class or property (Line 5) bound to different variables in q_1 is as follows. For each predicate $p(x_1, x_2)$: (1) if p is a property connecting two classes c_1 and c_2, we say that x_1 is bound to c_1 and that x_2 is bound to c_2; (2) if p connects a class c to a value (or literal) v, we say that x_1 is bound to c and that x_2 is bound to p. Also, we note that the algorithm uses the relational algebra *projection* operator π (Line 6).

Example 4. Given a global query

$$q : ans(x, y) \text{ :- } name(u, x), title(v, y), contained(u, v).$$

we use GLREWRITING to rewrite q into a union of subqueries, each on a local XML source (refer to the mapping table $\mathcal{M}$ of Table 3). For illustration, we only look at the rewriting of q into a subquery q_1 over the local source $\mathcal{S}_1$.

In particular, Line 5 computes the bound classes or properties of the variables (u, v, x, y) as (`Author`, `Book`, `Author.name`, `Book.title`). By looking into $\mathcal{M}$, we find the corresponding element sequence of (`Author`, `Book`, `Author.name`, `Book.title`) in $\mathcal{S}_1$ to be (`/books/book/author`, `/books/book`, `/books/book/author/name`, `/books/book/booktitle`). From Lines 7 to 11, we compute the predicates in the body of q_1 as follows.

$$q_1\text{: } ans(x, y) \text{ :- } /name(u, x), /booktitle(v, y)\ /author(v, u).$$

Note that for the predicate $contained(u, v)$ in q, we generate in q_1 a predicate $/author(v, u)$, where the order of the two variables is switched. This results from the computation performed by Lines 9 and 10. In particular, u and v are respectively bound

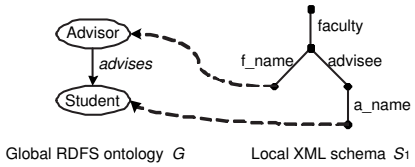

Fig. 7. A part of XML data integration setting.

to `Author` and `Book`, which respectively correspond to XML paths `/books/book/author` and `/books/book`. From $\mathcal{S}_1$, we find that `/author` is the path from v to u, not the path from u to v.

Example 5. We give one more example to illustrate query rewriting when Line 11 is used. Consider the following setting, where a local XML schema $\mathcal{S}_1$ (on the right side) is mapped to the global RDFS ontology $\mathcal{G}$ (on the left side), as indicated by the dashed lines. The two classes `Advisor` and `Student` are respectively instantiated with the name of `faculty` and the name of `advisee`, that is, the mapping table contains two correspondences:

(`Advisor`, `/faculty/f_name`)
(`Student`, `/faculty/advisee/a_name`).

Now we consider rewriting a global c-RDQL query q: $ans(x, y)$:- $advises(x, y)$. into a local c-XQuery query q' over $\mathcal{S}_1$. It is apparent that x and y are bound to `Advisor` and `Student`, thus corresponding to `/faculty/f_name` and `/faculty/advisee/a_name`, respectively. Because `/faculty/f_name` and `/faculty/advisee/a_name` share the same ancestor `/faculty`, by using Line 11 we add two predicates $/f\_name(u, x)$ and $/advisee/a\_name(u, y)$ to the body of q', generating the following local c-XQuery query q':

$$ans(x, y) \text{ :- } /f\_name(u, x),\ /advisee/a\_name(u, y).$$

Algorithm LLREWRITING
Input:1. q_1 over a local XML schema $\mathcal{S}_1$: $ans(\boldsymbol{X})$:- $p_1(\boldsymbol{X_1}), ..., p_m(\boldsymbol{X_m})$;
2. $\mathcal{M}$ between the global ontology $\mathcal{G}$ and local XML schemas $\mathcal{S}_1, ..., \mathcal{S}_n$.
Output: q: A query over local XML schema $\mathcal{S}_2$.
1. $head_q = ans(\boldsymbol{X})$; $body_q = null$;
2. **For** $j = 1$ **to** m **do**
3. (c_1, c_2) = name of the element bound to (x_1, x_2), for $\boldsymbol{X_j} = (x_1, x_2)$;
4. Search $\mathcal{M}$ to find (d_1, d_2) such that $\{(c_1, d_1), (c_2, d_2)\} \subseteq \pi_{\mathcal{S}_1,\mathcal{S}_2}(\mathcal{M})$;
5. **If** a path p exists from d_1 to d_2 in $\mathcal{S}_2$ **then**
6. add $p(x_1, x_2)$ to $body_q$;
7. **Else if** a path p exists from d_2 to d_1 in $\mathcal{S}_2$ **then**
8. add $p(x_2, x_1)$ to $body_q$;
9. **Else** add $p(\hat{x}, x_1)$ and $p'(\hat{x}, x_2)$ to $body_q$, where $\hat{x}$ is a new variable bound to the lowest ancestor d of d_1 and d_2, and p (p') is the path from d to $d_1(d_2)$;

Algorithm LLREWRITING differs from GLREWRITING only in finding the elements bound to the variables (Line 3) and in finding the corresponding elements from the mapping table (Line 4). Unlike in global-to-local rewriting, the result of using LLREWRITING is a single c-XQuery.

Taking into account the definitions of *global* and *P2P query containment*, we prove below that the algorithms GLREWRITING and LLREWRITING yield equivalent queries.

Theorem 1. *Given a global query q over the global ontology $\mathcal{G}$, its rewriting q' as computed by* GLREWRITING *is globally equivalent to q, that is, $q \equiv_{\mathcal{M}} q'$.*

PROOF SKETCH. To prove $q \equiv_{\mathcal{M}} q'$, where $q' = q_1 \cup ... \cup q_n$, we will check whether $cert_{\mathcal{M}}(q) = q_1^{\mathcal{D}_1} \cup ... \cup q_n^{\mathcal{D}_n}$, given the mapping table $\mathcal{M}(\mathcal{G}, \mathcal{S}_1, ..., \mathcal{S}_n)$. Taking into account the semantics of $\mathcal{M}$, given any sequence u of values from the global database $\mathcal{B}$, which makes $body_q$ true, we can always have a sequence v of values from $\mathcal{D}_1, ..., \mathcal{D}_n$, since $\sigma(\mathcal{G}) = \sigma(\mathcal{S}_1) \cup ... \cup \sigma(\mathcal{S}_n)$. By GLREWRITING, the sequence v is exactly the one that makes $body_{q_i}$ true, where $i \in [1..n]$. Therefore, we have $q^{\mathcal{B}} \subseteq q_1^{\mathcal{D}_1} \cup ... \cup q_n^{\mathcal{D}_n}$. Similarly, we can show that $q^{\mathcal{B}} \supseteq q_1^{\mathcal{D}_1} \cup ... \cup q_n^{\mathcal{D}_n}$. By the definition of certain answers, we conclude that $cert_{\mathcal{M}}(q) = q_1^{\mathcal{D}_1} \cup ... \cup q_n^{\mathcal{D}_n}$. □

Similarly, we have:

Theorem 2. *Given a local query q_1 over a local XML source $\mathcal{S}_1$, its rewriting q_2 over the local XML source $\mathcal{S}_2$ computed by* LLREWRITING *is P2P equivalent to q_1, that is, $q_1 \equiv_{\mathcal{M}} q_2$.*

We discuss here an interesting property, namely *reversibility*, of the local-to-local query rewriting. Informally, consider a local query q_1, which is rewritten into another local query q_2. If q_2 can be rewritten back to a query q_1' (on the same source as q_1) such that $q_1 \equiv q_1'$, we say q_1' is a *reverse* query of q_1. In the case that q_2 and q_1' are computed using the same rewriting algorithm, we say that the algorithm is *reversible*, if every query that is rewritable by the algorithm has a reverse rewriting.

More generally, we consider a P2P data integration system with a cyclic path of P2P mappings, informally annotated as $p_1, \mathcal{M}_{12}, p_2, ..., \mathcal{M}_{(n-1)(n)}, p_n, \mathcal{M}_{n1}, p_1$, and an equivalent query rewriting algorithm translating a query q_1 (over p_1) along this path until it comes back to p_1 with the resulting query q_1'. In the spirit of equivalent query rewriting, we expect that it is the case that $q_1 \equiv q_1'$, and furthermore, $(q_1 \equiv_{\mathcal{M}} q_2), ..., (q_n \equiv_{\mathcal{M}} q_1') \Rightarrow q_1 \equiv q_1'$ and $q_1 \equiv q_1' \Rightarrow (q_1 \equiv_{\mathcal{M}} q_2), ..., (q_n \equiv_{\mathcal{M}} q_1')$. In other words, we expect that there exists a logical relationship between P2P query containment/equivalence and a reversible rewriting algorithm.

6 Conclusions and Future Work

XML and its schema languages do not express semantics but rather the document structure, such as information about nesting. Therefore, semantically-equivalent documents often present different document structures when they originate from different applications. In this paper, we provide an ontology-based framework that aims to make XML documents interoperate at the semantic level while retaining their nesting structure. The framework consists of two key aspects: data integration and query processing.

For data integration, a global RDFS ontology is generated by merging the local RDFS ontologies that are generated from each of the XML documents. At the same time, the mappings between the global ontology and local XML schemas are manually established. We extend RDFS by defining additional metadata that can encode the nesting structure of an XML document. For query processing, we propose two query rewriting algorithms: one algorithm translates an RDF query (posed on the global ontology) to an XML query; the other algorithm translates an XML query (posed on one of the individual XML data sources) to another XML query (posed on a different XML data source). In doing so, we discuss the problem of query containment for two query languages, namely conjunctive RDQL (c-RDQL) and conjunctive XQuery (c-XQuery). It is shown that both query rewriting algorithms are equivalent, in terms of both global and P2P query equivalence.

In the future, we will extend query processing in our framework, by taking into account other data models, such as relational and RDF data sources. We will further study query containment in the case of more expressive query languages, e.g., the complete RDQL and XQuery. The concept of reversibility of query rewriting, especially in P2P data integration systems, is also a direction for future research.

References

1. S. Abiteboul and O. M. Duschka. Complexity of Answering Queries Using Materialized Views. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 1998)*, pages 254–263, 1998.
2. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
3. B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Ontology-Based Integration of XML Web Resources. In *Proceedings of the 1st International Semantic Web Conference (ISWC 2002)*, pages 117–131, 2002.
4. B. Amann, I. Fundulaki, M. Scholl, C. Beeri, and A.-M. Vercoustre. Mapping XML Fragments to Community Web Ontologies. In *Proceedings of the 4th International Workshop on the Web and Databases (WebDB 2001)*, pages 97–102, 2001.
5. Y. Arens, C. A. Knoblock, and C. Hsu. Query Processing in the SIMS Information Mediator. In *The AAAI Press*, May 1996.
6. Y. A. Bishr. Overcoming the semantic and other barriers to GIS interoperability. *International Journal of Geographical Information Science*, 12(4):229–314, 1998.
7. S. Boag, D. Chamberlin, M. F. Fernández, J. R. Daniela Florescu, and J. Siméon. XQuery 1.0: An XML Query Language. http://www.w3.org/TR/xquery, W3C Working Draft, April 2005.
8. R. Bourret. XML and Databases. http://www.rpbourret.com/xml/XMLAndDatabases.htm, December 2004.
9. D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema, W3C Working Draft, February 2004.
10. A. Calì, D. Calvanese, G. D. Giacomo, and M. Lenzerini. On the Expressive Power of Data Integration Systems. In *Proceedings of the 21st International Conference on Conceptual Modeling (ER 2002)*, pages 338–350, 2002.
11. A. Calì, D. Calvanese, G. D. Giacomo, M. Lenzerini, P. Naggar, and F. Vernacotola. IBIS: Semantic Data Integration at Work. In *Proceedings of the 15th Conference on Advanced Information Systems Engineering (CAiSE 2003)*, pages 79–94, 2003.

12. D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. View-Based Query Processing and Constraint Satisfaction. In *The 15th Annual IEEE Symposium on Logic in Computer Science (LICS 2000)*, pages 361–371, 2000.
13. D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. View-based Query Containment. In *Proceedings of the 22rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2003)*, pages 56–67, 2003.
14. S. D. Camillo, C. A. Heuser, and R. dos Santos Mello. Querying Heterogeneous XML Sources through a Conceptual Schema. In *Proceedings of the 22nd International Conference on Conceptual Modeling (ER 2003)*, pages 186–199, 2003.
15. Y. Chen and P. Revesz. CXQuery: A Novel XML Query Language. In *Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Science, and Medicine on the Internet (SSGRR 2002w)*, 2002.
16. I. F. Cruz and H. Xiao. Using a Layered Approach for Interoperability on the Semantic Web. In *Proceedings of the 4th International Conference on Web Information Systems Engineering (WISE 2003)*, pages 221–232, Rome, Italy, December 2003.
17. T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
18. A. Y. Halevy. Answering Queries Using Views: A Survey. *VLDB Jounal*, 10(4):270–294, 2001.
19. A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: Data Management Infrastructure for Semantic Web Applications. In *Proceedings of the 12th International World Wide Web Conference (WWW 2003)*, pages 556–567, 2003.
20. HP Labs. RDQL - RDF Data Query Language. http://www.hpl.hp.com/semweb/rdql.htm, 2005.
21. Y. Kalfoglou and M. Schorlemmer. Ontology Mapping: the State of the Art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.
22. M. C. A. Klein. Interpreting XML Documents via an RDF Schema Ontology. In *Proceedings of the 13th International Workshop on Database and Expert Systems Applications (DEXA 2002)*, pages 889–894, 2002.
23. L. V. S. Lakshmanan and F. Sadri. Interoperability on XML Data. In *Proceedings of the 2nd International Semantic Web Conference (ICSW 2003)*, pages 146–163, 2003.
24. P. Lehti and P. Fankhauser. XML Data Integration with OWL: Experiences and Challenges. In *2004 Symposium on Applications and the Internet (SAINT 2004)*, pages 160–170, 2004.
25. M. Lenzerini. Data Integration: A Theoretical Perspective. In *Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2002)*, pages 233–246, Madison, Wisconsin, June 2002. ACM.
26. F. Manola and E. Miller. RDF Primer. http://www.w3.org/TR/rdf-primer, W3C Working Draft, February 2004.
27. E. Mena, V. Kashyap, A. P. Sheth, and A. Illarramendi. OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. In *Proceedings of the 1st IFCIS International Conference on Cooperative Information Systems (CoopIS 1996)*, pages 14–25, 1996.
28. T. D. Millstein, A. Y. Halevy, and M. Friedman. Query Containment for Data Integration Systems. *Journal of Computer and System Sciences*, 66(1):20–39, 2003.
29. N. F. Noy and M. A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI 2000)*, pages 450–455, 2000.
30. Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object Exchange Across Heterogeneous Information Sources. In *Proceedings of the 11th International Conference on Data Engineering (ICDE 1995)*, pages 251–260, 1995.

31. P. F. Patel-Schneider and J. Siméon. The Yin/Yang Web: XML Syntax and RDF Semantics. In *Proceedings of the 11th International World Wide Web Conference (WWW 2002)*, pages 443–453, July 2002.
32. L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, and R. Fagin. Translating Web Data. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB 2002)*, pages 598–609, 2002.
33. O. D. Sahin, A. Gupta, D. Agrawal, and A. E. Abbadi. Query Processing Over Peer-To-Peer Data Sharing Systems. Technical Report CSD-2002-28, University of California at Santa Barbara, 2002.
34. L. A. Shklar, A. P. Sheth, V. Kashyap, and K. Shah. InfoHarness: Use of Automatically Generated Metadata for Search and Retrieval of Heterogeneous Information. In *Proceedings of the 7th Conference on Advanced Information Systems Engineering (CAiSE 1995)*, pages 217–230, 1995.
35. G. Stumme and A. Maedche. Ontology Merging for Federated Ontologies for the Semantic Web. In *Proceedings of the International Workshop on Foundations of Models for Information Integration (FMII 2001)*, pages 16–18, 2001.
36. J. D. Ullman. Information Integration Using Logical Views. In *Proceedings of the 6th International Conference on Database Theory (ICDT 1997)*, pages 19–40, 1997.
37. R. van der Meyden. Logical Approaches to Incomplete Information: A Survey. In *Logics for Databases and Information Systems*, pages 307–356, 1998.
38. H. Xiao, I. F. Cruz, and F. Hsu. Semantic Mappings for the Integration of XML and RDF Sources. In *Proceedings of the VLDB Workshop on Information Integration on the Web (VLDB-IIWeb 2004)*, 2004.