Mediation of XML Data through Entity Relationship Models

Irini Fundulaki\textsuperscript{1⋆} and Maarten Marx\textsuperscript{2⋆⋆}

\textsuperscript{1} Bell Laboratories, Lucent Technologies, USA and INRIA-Rocquencourt, France.
\texttt{fundulaki@research.bell-labs.com}

\textsuperscript{2} Institute for Logic Language and Computation, University of Amsterdam, The Netherlands.
\texttt{marx@science.uva.nl}

\textbf{Abstract.} This paper describes an approach for the querying of heterogeneous XML resources using an ontology-based mediator. Here an ontology is an Entity-Relationship schema defined independently of the schemas of the data sources. The sources are described to the mediator by means of mapping rules as in the Local-As-View approach to data integration. User queries are conjunctive queries formulated in terms of the ontology, and answers to these queries are obtained by rewriting them to XQuery expressions and evaluating these on the data sources. A formal semantics for queries is defined by interpreting XML sources into ER models. As there can be many such interpretations, a certain answer to a query is one which is true in all of them. We describe the rewriting algorithm and we show its completeness and correctness with respect to the given semantics. We also give an algorithm for producing a canonical model of the ontology and the interpreted data sources. It is shown that the certain answers can also be obtained by evaluating the query to just this one model.

1 Introduction

XML [1] is becoming the de-facto standard for data representation and exchange of Web data and plays an essential role in the deployment of the Semantic Web whose goal is “to develop enabling technologies and standards to support richer discovery, data integration, navigation and automation of tasks” [31]. In such an environment where the sources are autonomous and heterogeneous, data integration is a critical issue. The goal there is to enable users to query the data of heterogeneous sources as if it resides in a single source, which is exactly what a data integration system does. The fact that the sources concern a \emph{restricted domain of interest} is crucial for the successful deployment of data integration systems. Their backbone is a \emph{global schema} which describes the basic notions in

\textsuperscript{⋆} This work is funded by a scholarship from the French National Institute for Research in Computer Science and Control (INRIA).

\textsuperscript{⋆⋆} Research funded by NWO grant 612.000.106.
the domain. Appropriate mappings between the global schema and the schemas of the sources are defined to describe the latter to the former. User queries are formulated in terms of the global schema and the answers are obtained by accessing the source data.

There are several questions that need to be considered when one is constructing a data integration system: (i) is the global schema defined out of the sources schemas (e.g. federated architecture [22]), or is independent thereof (e.g. defined by an authority in the domain); (ii) is query answering done on the fly by translating the user query into queries expressed in the sources’ schemas (query mediation paradigm [34, 26]), or by evaluating it against the database which has been constructed out of the source data (e.g. data warehouse paradigm [33]). In an evolving environment like the Web, defining the global schema out of the sources schemas may lead to significant overload of schema maintenance: every insertion/deletion of a new source or modification of an already integrated source’s schema, leads to modification(s) of the global schema. Besides that, the global schema can become quite unintelligible, due to the idiosyncrasies present in the different local schemas. An independently defined global schema (an ontology) seems most appropriate for the integration of web data, and this line is followed in the paper. In an environment where data is changing rapidly, keeping the data warehouse “up-to-date” is not the easiest of tasks, so we choose to study mediation.

Another dimension of a data integration system is the approach used to define the mappings between the global schema and the sources schemas. There are two prevailing approaches: Global-As-View (GAV) and Local-As-View (LAV) [23]. In the former, the global schema relations are defined as views of the sources relations. The mappings are conjunctive queries\(^3\) which specify how to obtain the tuples for the former: their head involves one global schema relation, and their body involves a conjunction of source relations. Queries are formulated in terms of the global schema relations and their translation is done by substituting the global schema relations by their definitions\(^4\). In the latter, a source is described as a (materialized) view of the global schema relations [24]. The mappings (also conjunctive queries) have the inverse direction of those in the GAV approach. Query translation in this context is known as \textit{querying rewriting using views} [24]. The drawback of the first approach is that the global schema depends on the sources schemas, while in the second, it is defined independently of them. But, query translation in LAV is more complicated than in GAV. In the simple case of conjunctive queries, it has been proved \textit{NP-hard} [24].

A number of algorithms have been proposed in the literature for the latter approach. Authors in [25, 29, 28, 15, 17, 21] consider query rewriting for conjunctive queries and relational views. Algorithms in [25, 29] are characterized as \textit{bucket-based} where the idea is to match each query subgoal with the body of the view definitions. Authors in [15, 17] follow a different approach where they produce

\(^3\) We use the term conjunctive query to refer to \textit{rule-based conjunctive query} as defined in [2].

\(^4\) This substitution is called query unfolding.
a set of rewriting rules out of the view definitions. Query rewriting is done by
matching the body of the rules with the body of the query. Rewriting in a dif-
ferent framework is done in [8, 20, 7] where regular path queries and views are
considered.

In this paper we consider query rewriting in the framework proposed in [4, 18].
A mediator-based architecture for the integration of XML resources is proposed
where the LAV approach is used to describe the sources to the global schema,
the latter defined independently of the schemas of the former. The global schema
is an ontology which incorporates the basic features of the Entity Relationship
(ER) model [9] (which can be easily represented as an object-oriented or as an
RDF schema [30]). The principal contributions of this work in the domain of
XML data integration are:

- the identification of a reasonable subset of the web data integration problem
  in which mediation is a feasible alternative to data warehousing;
- the use of a rich conceptual schema which considers inverse roles and global
  keys, instead of a relational one as the global schema of the mediator;
- an expressive mapping language to describe a source’s schema to the global
  schema. All previous approaches consider that a source is described as a
  conjunctive query over the global schema relations. In [4] mapping rules
  which associate XML fragments (expressed by XPath [10] expressions) to
  ontology paths are used.

This approach has several advantages. First ontologies are more expressive
than XML DTDs or Schemas due to the presence of (i) subsumption relations
and (ii) typed binary relationships between entities. In XML neither of them
exist, and the only type of relationships between nodes is the parent/child and
attribute relationships. The ID/IDREF attribute mechanism can be used to relate
two nodes but these relationships are untyped. Second, the use of XPath in
the mapping language allows one to describe arbitrary XML structures, and
in addition to associate specific semantics (expressed in the ontology) to the
relationships between XML nodes. We will not discuss here the benefits of using
an ER schema as the global schema. Detailed discussion on this issue can be
found in [3].

In this context, the result of the rewriting for a query \( q \) and for a set of sources \( S \)
is the union of all the rewritings of \( q \) for each source \( s \). The idea behind rewriting
\( q \) for \( s \) is simply to match the query variable binding paths to the ontology paths
of the rules. The proposed framework along with the query rewriting algorithms
has been implemented in the ST3X prototype [19].

A similar but simpler approach has been undertaken in [13]. The global
schema is a node-labeled tree (called abstract DTD) hence there is no notion of
subsumption relationship and parent/child is the only relationship between
nodes. XML sources are described by concrete DTDs (also node-labeled trees).
The mapping language is much simpler than in [4]: only the child axis of XPath
is used where absolute abstract paths are mapped to absolute concrete paths\(^5\).

\(^5\) Paths are called absolute since they are specified only from the root node of the
abstract/concrete DTDs.
Given a user query expressed in terms of the abstract DTD, their query rewriting algorithm tries to find the concrete paths that match the abstract paths.

In this paper we examine the completeness and soundness of the rewriting algorithm in [4] for a query $q$ and a source $s$ following a different approach for the manipulation of ontology paths in the models of the ontology. In this new setting we started by examining the proposed framework and in order to achieve our goal we had to consider the following restrictions: (i) the ontology contains no inverse roles, (ii) no global keys are considered and (iii) only the child and attribute axis of XPath are used in the mapping rules. Hence, we had to contemplate with a poorer mapping language than the original one to prove completeness. The conclusion that we can draw is that if one wants to go for expressive power, then one must pay the price of incompleteness.

The paper is organized as follows: in Section 2 we present the approach followed in [4] using a cultural example. Section 3 gives the formal definitions for the ontology, the mapping rules, queries and answers. In Section 4 we show how a model for the ontology is built in which the XML sources are interpreted using a Tableau System. We also show how this model can be used to yield all certain answers to queries. Section 5 presents the query rewriting algorithm and proves its completeness and correctness. Conclusions are given in Section 6. Proofs of all statements are provided in the Appendix.

2 Overview of the approach through a cultural example

We give in this section an overview of the approach in [4] using an example from the cultural domain. An XML resource is considered to contain a number of XML documents which conform to a unique XML DTD\(^6\). The upper side of Fig. 1 shows an XML DTD describing painters and their paintings for source \(http://www.paintings.com\). Each painter (element \texttt{Painter}, line 2) is associated with one or more paintings (element \texttt{Painting}). A painter has a name (attribute \texttt{name}, line 3). A painting has a title (attribute \texttt{title}, line 5). Fig. 1 presents an XML document that conforms to the DTD.

The backbone of the mediator is an ontology which incorporates the basic features of the Entity Relationship (ER) model [9]. The terms \textit{concept} and \textit{role} are used to name entities and relationships respectively. Concepts are related to each other with the \textit{isa} subsumption relation. We use the same ontology as the one used in [4] as a reference example, which is inspired from the ICOM/CIDOC Reference Model [14]\(^7\).

The ontology is depicted in the graphical notation of ER schemas, with the addition of \textit{isa} arcs and is shown in Fig. 2. It describes concepts such as \texttt{Actor},

\(^6\) Although this may look like a simplification from the setting with a number of heterogeneous sources, on our level of description it is not: using appropriate renaming every subset of XML documents with their own DTD can be equivalently transformed into one source with one DTD.

\(^7\) The ICOM/CIDOC model has been defined and used for the documentation of cultural information. Several cultural authorities have participated in this effort whose basic purpose was to provide a single schema to exchange their data.
1. `<!ELEMENT Collection (Painter+)>
2. `<!ELEMENT Painter (Painting+)>
3. `<!ATTLIST Painter name CDATA #REQUIRED>
4. `<!ELEMENT Painting EMPTY>
5. `<!ATTLIST Painting title CDATA #REQUIRED>

```xml
<Collection>
  <Painter name='Rembrandt'>
    <Painting title='de Nachtwacht'/>
    <Painting title='de Staalmeesters'/>
  </Painter>
  <Painter name='Vermeer'>
    <Painting title='de Brief'/>
    <Painting title='het Melkmeisje'/>
  </Painter>
</Collection>
```

Fig. 1. XML DTD and document for source http://www.paintings.com.

Fig. 2. An Ontology for Cultural Artifacts.

- **R1**: http://www.paintings.com/Collection/Painter as \( u_1 \) → Person
- **R2**: \( u_1/\text{name} \) → has_name
- **R3**: \( u_1/\text{Painting} \) as \( u_3 \) → carried_out/produced
- **R4**: \( u_3/\text{title} \) → has_title

Fig. 3. Set of Mapping Rules for source http://www.paintings.com.

its subconcept Person, Activity, Man_Made_Object, and Museum. The concepts are connected using binary roles such as carried_out, produced and located_at. The fact that an actor (instance of concept Actor) performs an activity (instance of concept Activity) to produce a man made object (instance of concept Man_Made_Object) is represented by roles carried_out and produced. Concepts may also have attributes: attribute has_name is defined in concept Person and takes its values in the atomic type String. In the ontology, each role has an inverse depicted within parenthesis.

To describe an XML resource to the ontology a set of mapping rules is defined. Each rule associates an XPath location path [10] to concepts, attributes and (composite) roles in the ontology (we use '/' to denote the composition of two roles).

The set of rules illustrated in Fig. 3 map fragments of XML documents which conform to the DTD of Fig. 1 into the ontology of Fig. 2. Each rule has a left
hand side and a right hand side. The right hand side is a concept, attribute or (composite) ontology role (referred to as ontology paths in the sequel). The left hand side is composed of (i) an XPath location path evaluated on some URL or on some variable and (ii) an optional variable declaration.

The first rule states that all elements returned when evaluating the XPath expression Collection/Painter on the root nodes of documents in URL http://www.paintings.com are instances of concept Person and this set of elements is bound to variable u_1. The statement ‘as u_1’ is a variable declaration. The second rule states that values of the attribute has\_name of Person are obtained from evaluating the XPath expression @name on instances of variable u_1 (Painter elements). The third rule states that x carried\_out/produced y holds whenever x is a Painter element (element of the variable u_1) and y is a Painting element, child of x. The obtained Painting elements are bound to variable u_3 which is used in the last rule to obtain the values for attribute has\_title.

The mapping rules allow one to define the semantics of XML fragments and their structural relationships in terms of the ontology. Thus, R_1 defines a subset of the extension of concept Person, while rule R_3 relates elements in this subset by the composite role carried\_out/produced to a subset of the extension of Man\_Made\_Object.

Queries are conjunctive queries expressed in the vocabulary of the ontology [2]. Such queries can be presented in several formats. In more theoretical work, conjunctive queries are presented as Datalog rules. Commercial systems often employ a select/from/where formulation (as in SQL). The query “Which objects are created by Vermeer” is expressed in the OQL syntax [11] and as a conjunctive query in Table 1. These two formulations differ only with respect to their notion of an answer. An answer to the conjunctive query is an instantiation of x_3 which makes the body of the rule true. The answer to the OQL query is the set of all these instantiations. In the sequel we use answer in the former sense.

Each variable in the query is bound to some ontology path (called in the sequel its binding path). For example, for the OQL query Q_1, Person is the binding path of x_1, has\_name is the binding path of x_2 and finally carried\_out/produced is x_3’s binding path.

Given a query, the rewriting algorithm proposed in [4] works as follows: given that each variable in the query is bound to some ontology path, the algorithm searches for mapping rules or concatenations thereof, that can be used to translate these paths to XPath location paths. This is done by matching the binding paths of the query variables against the ontology paths of the mapping rules. For example for Q_1 and for the set of mapping rules depicted in Fig. 3, we see that instances for variable x_1 are found by rule R_1, for variable x_2 by R_2 and for variable x_3 by rule R_3. By substituting the binding path of a query variable with the location path of the corresponding rule, we obtain query Q_1(a) which can be easily translated into the XQuery [12] expression Q_1(b) both shown in Table 2. For the document of Fig. 1, there are two answers to this query: <Painting title =" de Brief ">
and <Painting title =" het Melkmeisje ">.
\[ Q_1: \text{select } x_3 \\
\text{from Person } x_1, x_1.\text{has name } x_2, \\
x_1.\text{carried out/produced } x_3 \\
\text{where } x_2 = \text{“Vermeer”} \]

\[ Q_1(x_3) := \text{Person}(x_1), \text{has name}(x_1) = x_2, \text{carried out/produced}(x_1, x_3), x_2 = \text{“Vermeer”}. \]

Table 1. Query \( Q_1 \).

\[ Q_1(a): \text{select } x_3 \\
\text{from } \text{http://www.paintings.com/Collection/Painter } x_1, \\
x_1.\$\text{name } x_2, x_1.\text{/Painting } x_3, \\
\text{where } x_2 = \text{“Vermeer”} \]

\[ Q_1(b): \text{FOR } \$x_1 \text{ IN document(‘http://www.paintings.com’)/Collection/Painter,} \\
\$x_2 \text{ IN } \$x_1/\$\text{name,} \\
\$x_3 \text{ IN } \$x_1/\text{Painting} \\
\text{WHERE } \$x_2 = \text{“Vermeer”} \\
\text{RETURN } \$x_3 \]

Table 2. Queries \( Q_1(a) \) and \( Q_1(b) \).

3 Mapping XML resources to Ontologies

In the setting proposed in [4] several XML sources are integrated into one ER model. Whence, it is not evident what constitutes a correct answer to a query. In this section we develop a theory for interpreting XML sources into ER models and use that to define the notion of a certain answer. This notion originated in the context of incomplete databases and has been used for query answering in data integration [23].

**Ontologies** As aforementioned, ontologies incorporate the basic features of the Entity Relationship (ER) model [9]. The model considered in [4] differs from standard ER models in three respects: (i) only binary relationships (called roles) between entities are allowed, (ii) only two roles (called source and target assigning the domain and range to each relationship) are used, and (iii) attributes are partial rather than total functions.

Formally, an ontology is an 8-tuple \( O = (C, R, S, A, \text{source}, \text{target}, \text{isa}, \text{key}) \), where: (i) \( C \) is a set of concept symbols, (ii) \( R \) is a set of role symbols, (iii) \( S \) is the set of atomic types defined in the XML Schema Datatypes document [5], (iv) \( A \) is a set of attribute symbols, (v) source is a function that assigns to roles and attributes their domain in \( C \), (vi) target is a function that assigns to roles their range in \( C \) and to attributes their range in \( S \), (vii) isa is a subsumption relation between concepts in \( C \), (viii) \( \text{key}(-) \) is a function from \( C \) to \( \mathcal{P}(A) \), assigning to every concept a (possibly empty) set of its attributes. \text{key}(-) is such that for each pair of concepts \( c_1, c_2 \) with \( c_1 \ 	ext{isa} c_2 \) it holds that \( \text{key}(c_1) \subseteq \text{key}(c_2) \)\(^8\). We use \( \text{isa}^* \) to denote the reflexive and transitive closure of the isa relation.

\(^8\) In contrast to [4] we only deal with single valued keys and with at most one key per concept. In general, concepts may have several multivalued keys as in [6]. The general
Actor$^{35}$ = \{p$_1$, p$_2$\}  \quad \text{Person$^{35}$} = \{p$_1$, p$_2$\}
Activity$^{35}$ = \{a$_1$, a$_2$, a$_3$\}  \quad \text{Man Made Object$^{35}$} = \{o$_1$, o$_2$, o$_3$, o$_4$\}  \quad \text{Museum$^{35}$} = \emptyset

carried$_{\text{out}}^{35}$ = \{(p$_1$, a$_1$), (p$_1$, a$_2$), (p$_2$, a$_3$)\}
produced$_{\text{by}}^{35}$ = \{(a$_1$, o$_1$), (a$_2$, o$_2$), (a$_3$, o$_3$), (a$_4$, o$_4$)\}  \quad \text{located$_{\text{at}}^{35}$} = \emptyset
has$_{\text{title}}(o_1) = \text{"de Nachtwacht"}  \quad \text{has$_{\text{name}}(p_1) = \text{"Rembrandt"}}$
has$_{\text{title}}(o_2) = \text{"de Staalmeesters"}  \quad \text{has$_{\text{name}}(p_2) = \text{"Vermeer"}}$
has$_{\text{title}}(o_3) = \text{"de Brief"}  \quad \text{has$_{\text{title}}(o_4) = \text{"het Melkmeisje"}}$

| Table 3. A model $\mathfrak{B}$ for the ontology $O$. |

The semantics of an ontology can be given by specifying which database states are consistent with the information structure represented by the ontology. Formally, a database state $\mathfrak{B}$ for an ontology $O$ consists of a nonempty finite set $\mathcal{D}^3$ (which is disjoint from all basic domains) and a function $(\cdot)^3$ which interprets the symbols of the ontology: concepts $c$ as subsets $c^3$ of $\mathcal{D}^3$; roles $r$ as subsets $r^3$ of $\mathcal{D}^3 \times \mathcal{D}^3$; attributes $a$ as partial functions $a^3$ from $\mathcal{D}^3$ to the union of the basic domains; atomic types $S$ as the corresponding basic domains $S^3$.

A database state $\mathfrak{B}$ is a model for the ontology $O$, if (i) $\mathfrak{B}$ interprets the concept, role, attribute and atomic type symbols of the ontology, (ii) for each pair of concepts $c_1, c_2$ with $c_1$ isa $c_2$ it holds that $c_1^3 \subseteq c_2^3$; (iii) for each role $r$, the domain and range$^3$ of $r$ are subsets of source$(r)^3$ and target$(r)^3$, respectively, (iv) for each attribute $a$, for each $c \in \mathcal{D}^3$, if $a(c)$ is defined then $c \in \text{source}(a)^3$ and $a(c) \in \text{target}(a)^3$; and (v) for all $x, y \in \mathcal{D}^3$, whenever $x, y \in c^3$, for some concept $c$ and $\{a_1, \ldots, a_n\}$ is the key for $c$ it holds that: $x = y$ iff $a_1^3(x) = a_1^3(y)$ and $\ldots$ and $a_n^3(x) = a_n^3(y)$ (i.e. $\mathfrak{B}$ does not contain different objects with the same value for a key).

A model $\mathfrak{B}$ for the ICOM/CIDOC ontology is shown in Table 3. The domain $\mathcal{D}^3$ of $\mathfrak{B}$ consists of the following set of elements: \{p$_1$, p$_2$, a$_1$, a$_2$, o$_3$, o$_4$, o$_5$, o$_6$\}. There is only one basic domain of type String.

Mapping rule $R_3$ in Fig. 3 uses the composite role carried$_{\text{out}}$/produced. Composite roles have proven to be very useful for mapping XML elements into the ontology [3] and can be seen as describing paths in the ontology. A role path (rolepath) is a composition of roles, a concept path (conceptpath) is a composition of a concept and a role path and finally an ontology path (ontolgypath) is either a concept path, a role path, an attribute or a composition of a conceptpath and an attribute (the latter called attribute path).

\[
\begin{align*}
\text{rolepath} &::= r | \text{rolepath }/ r \\
\text{conceptpath} &::= c | c / \text{rolepath} \\
\text{ontolgypath} &::= \text{conceptpath }| \text{rolepath }| a | \text{conceptpath }/ a & \text{ for } a \in A
\end{align*}
\]

Case leads to no new conceptual difficulties; the only difference is that determining identity of objects is a longer process.

9 The domain of a role $r$ is the set $\{x | \exists y. (x, y) \in r^3 \}$. Its range is the set $\{x | \exists y. (y, x) \in r^3 \}$.  

364
The source of a role path is the source of its first element; the target of a role path is the target of its last element. A composition of two roles, or of a role and an attribute \( r/s \) is safe if \( \text{target}(r) \isa^* \text{source}(s) \). The composition of a concept and a role or attribute \( c/r \) is safe if \( c \isa^* \text{source}(r) \). An ontology path is safe if all its compositions are safe.

Examples of (safe) role paths in the ICOM ontology are \textit{carried out} and \textit{carried out/produced}. \textit{Person/carryed out/produced} is a safe concept path.

The interpretation of ontology paths in a model of an ontology is defined as follows: For \( p = r_1 / \ldots / r_n \), a role path, \( (r_1 / \ldots / r_n)^\equiv = r_1^\equiv \circ \ldots \circ r_n^\equiv \).

For \( p = c/r \) a concept path, \( (c/r)^\equiv = \{ x \mid \exists y \in c^\equiv \text{ and } (y, x) \in r^\equiv \} \). For \( p = c/a \) with \( c \) a concept path and \( a \) an attribute, \( (c/a)^\equiv = \{ (x, y) \mid x \in c^\equiv \text{ and } a(x) = y \} \).

\textbf{Mapping Rules} As presented in Section 2, an XML resource is described to the ontology by means of mapping rules which associate XPath location paths to ontology paths. A mapping rule is an expression of the form \( R: u/q \text{ as } v \rightarrow p \), or \( R: u/q \rightarrow p \) where: (i) \( R \) is the rule’s label, (ii) \( u \) is either a variable or a URL, (iii) \( q \) is an XPath location path, (iv) \( p \) is an ontology path, restricted as follows: \( p \) is an attribute only if \( q \) is of the form \( @q' \) where \( q' \) is an XML attribute and \( u \) is a variable; \( p \) is a role path only if \( u \) is a variable, and a concept path only if \( u \) is a url (iv) as \( v \), is a variable declaration, present only if \( p \) is a role path or a concept path. A set of mapping rules is called a mapping. Rules are distinguished between \textit{relative} and \textit{absolute}: a rule \( R \) is called absolute if it starts with a URL, relative otherwise. Concatenation of mapping rules is defined as follows: for \( R_1: a/q_1 \text{ as } v_1 \rightarrow p_1, R_2: v_1/q_2 \text{ as } v_2 \rightarrow p_2 \) two rules in a mapping \( M \), their concatenation is the new rule \( R_1/R_2: a/q_1/q_2 \text{ as } v_2 \rightarrow p_1/p_2 \). Given a mapping \( M \), its closure, denoted by \( M^* \), is the set of all rules that can be obtained from \( M \) by repeated concatenation. Its \textit{expansion}, denoted \( \dot{M} \), is the set of absolute rules in \( M^* \).

\textbf{Definition 1 (Well Presented Mapping).} We call a mapping well presented if it satisfies the following two conditions: (i) all ontology paths of rules in \( M \) are safe, and (ii) if \( R_1: u/q \text{ as } v_1 \rightarrow p_1, R_2: u/q \text{ as } v_2 \rightarrow p_2 \) are both in \( M \), then \( v_1 = v_2 \).

It is straightforward to check that the mapping rules in Fig. 3 are well presented.

From this point on, and w.l.o.g. we assume that a source is an XML document specified by a URL \( u \) and a mapping \( M \) in which \( u \) appears as the only URL.

As described in Section 2, a mapping \( M \) from an XML source \( s \) to an ontology \( O \) populates the concepts, roles and attributes of \( O \). This is done formally by an interpretation function which maps the elements and values in an XML document to a model \( \mathcal{B} \) of the ontology.

\textbf{Definition 2 (Interpretation Function).} Let \( (u, M) \) be a source, \( d \) the XML document specified by \( u \) and \( \mathcal{B} \) a model for an ontology \( O \). An interpretation of \( r_1^\equiv \circ r_2^\equiv \) is defined as follows: \( r_1^\equiv \circ r_2^\equiv = \{ (x, y) \mid \exists z. (x, z) \in r_1^\equiv \text{ and } (z, y) \in r_2^\equiv \} \).
(u, M) in B is a function f which: (i) sends elements of d to D; (ii) sends attribute values of d to the basic domains of B of the same type; and (iii) respects all rules in the expansion of M in the following sense\textsuperscript{11}: Let \( R_1 : u/q \rightarrow p \) be an absolute and \( R_2 : v/q' \rightarrow r \) a relative mapping rule. We say that f respects \( R_1 \) if \( \{ f(e) \mid e \in u/q \} \subseteq p \). We say that f respects \( R_1/R_2 \) if f respects \( R_1 \) and for all \( e \in u/q, \) for all \( e' \in e/q' \), \((f(e), f(e')) \in r \).

In the sequel it will be convenient to assume that the database state in which a source is interpreted has the set of elements of the XML document as constant symbols. Then we can use the XML element \( e \) to refer to the database object \( f(e) \). So given an interpretation \( f \) of a source \( (u, M) \) in a database state \( B \), we assume that \( B \) contains the set of elements of the XML document specified by \( u \) as constant symbols and the interpretation is such that \( f(e) = e \). Often we just equate \( e \) with \( e \).

**Example 1.** We will interpret the XML document of Fig. 1 into the model of the ICOM ontology given in Table 3, in such a way that it respects the mapping rules \( R_1 – R_4 \). The function \( f \) is shown in Fig. 4. The XML document has two painter elements, one for Rembrandt, and one for Vermeer. \( f \) maps the first to \( p_1 \), and the second to \( p_2 \). The four painting elements are mapped to the four objects \( o_1 – o_4 \). The values of the XML attributes name and title are mapped to the identical strings in the basic domain of String. It is not hard to check that \( f \) is indeed an interpretation.

\[
\begin{align*}
  f(<\text{Painter name} = 'Rembrandt'>)
  &<\text{Painting title} = 'de Nachtwacht'/> \\
  &<\text{Painting title} = 'de Staalmeesters'/> \\
  &</\text{Painter}> = p_1 \\
  f(<\text{Painter name} = 'Vermeer'>)
  &<\text{Painting title} = 'de Brief'/> \\
  &<\text{Painting title} = 'het Melkmeisje'/> \\
  &</\text{Painter}> = p_2 \\
  f(<\text{Painting title} = 'de Nachtwacht'/> = o_1 \\
  f(<\text{Painting title} = 'de Staalmeesters'/> = o_2 \\
  f(<\text{Painting title} = 'de Brief'/> = o_3 \\
  f(<\text{Painting title} = 'het Melkmeisje'/> = o_4
\end{align*}
\]

**Fig. 4.** Interpreting the XML document of Fig. 1 into the ontology model of Table 3.

**Queries and Answers** Queries are conjunctive queries expressed in the vocabulary of the ontology \[2\] extended with the ontology paths and some fixed set

\textsuperscript{11} In this paper we use the expression \( e' \in e/q' \) to denote that element \( e' \) is obtained when evaluating XPath location path \( q' \) on element \( e \). We use this expression instead of \( e' \in [q'](e) \) as defined in \[32\].
of comparison predicates defined over the basic domains. Formally, given an
ontology \( O \), a conjunctive query is an expression of the form:

\[
q(\bar{x}) : = \bigwedge c(x), \bigwedge r(x, x'), \bigwedge a(x) = x', \bigwedge x\theta n.
\]

where \( \bar{x} \) is a sequence of variables all occurring in the body of the query, every
c is a concept path, every r a role path, every a an attribute, and every \( x\theta n \) a comparison
predicate where \( x \) is a variable, \( n \) is a value and \( \theta \) is one of \( \{=, <, >\}
\).

Conjunctive queries in which no composite concepts or roles occur, are called
atomic. Conjunctive queries can be easily transformed into queries in an OQL-
like syntax \(^\text{12}\). We recall here the definition of an answer to a conjunctive query
to a model \( \mathcal{D}^{\mathcal{B}} \) [2]. Let \( q(x_1, \ldots, x_n) \) be a conjunctive query with variables
\( x_1, \ldots, x_n \) and \( y_1, \ldots, y_m \). Then \( a_1, \ldots, a_n \) is an answer to \( q(x_1, \ldots, x_n) \) if \( \mathcal{D}^{\mathcal{B}} \models \exists y_1 \ldots \exists y_m q(a_1, \ldots, a_n) \), where \( \models \) is the standard first order satisfaction relation.

Finally we come to the central definition in this paper. Let \( O \) be an ontology,
\( (u, M) \) a source mapping elements of the documents in \( u \) into \( O \), and \( q(\bar{x}) \)
a conjunctive query expressed in the vocabulary of the ontology. We want to
define an intuitively reasonable notion of a correct answer to \( q(\bar{x}) \) given \( O \) and
\( (u, M) \). This notion then will determine whether an algorithm which produces
the answers is both correct and complete. The definition below captures the
idea that something is an answer to a query if there is no countermodel. In the
literature on data integration, these are called the certain answers [23]. More
formally, a list of XML elements and attribute values \( \bar{a} \) of \( u \) is an answer to the
query \( q(\bar{x}) \) if there is no model \( \mathcal{B} \) for \( O \) in which the source is interpreted and
where \( \mathcal{B} \not\models q(\bar{a}) \). This is usually formulated positively as follows:

**Definition 3 (Certain answer).** Let \( (u, M) \) be a source, \( O \) an ontology and
\( q(\bar{x}) \) a conjunctive query in the language of \( O \). \( M \) maps the elements of \( u \) into \( O \).
Let \( \bar{a} \) be a list of XML elements and attribute values coming from the source. We
say that \( \bar{a} \) is a certain answer to the query \( q(\bar{x}) \) posed to \( (u, M) \), if for each model
\( \mathcal{B} \) for \( O \), for each interpretation \( f \) of \( (u, M) \) into \( \mathcal{B} \) it holds that
\( \mathcal{B} \models q(f(\bar{a})) \).

### 4 Data Warehousing

Given an ontology \( O \) and a source \( (u, M) \) which is mapped to \( O \) we can con-
struct a model of \( O \) using a tableau system. This model can be seen as a data
warehouse. In the previous section we have seen that there are many models of
\( O \) in which a source can be interpreted. A certain answer was an answer which is
true in all these interpretations. Rather surprisingly now, all certain answers to a

\(^{12}\) Let \( q(\bar{x}) \) be a conjunctive query. Query \( q(\bar{x}) \) can be transformed into query \( q' \) in
an OQL-like syntax as follows: \( q' \)'s select clause contains the variables in \( \bar{x} \); \( q' \)'s
from clause contains \( \bigwedge c(x), \bigwedge r(x, y), \bigwedge a(x, y) \) (\( a(x, y) \) for the expression of the
form \( a(x) = y \)) and \( q' \)'s where clause contains the conjunction of the comparison
predicates.
conjunctive query can be obtained by evaluating the query to just this one constructed model (Theorem 3). In the context of data exchange [16] such a model has been called the universal canonical solution. We use the same terminology. To have a constructive way of producing all certain answers turns out very useful in the next section: it will be straightforward to establish completeness of the rewriting algorithm on the basis of the tableau system.

The tableau rules are given in Table 4. They can be grouped according to their function. The mapping rule rules implement the meaning of the mapping rules. The domain, target, isa and global key rules derive facts from the structure of the ontology. The composition, inverse and equality rules specify the logic behind these operations. The comparison rule just lists facts true of the basic domains. Given that a source is finite, the tableau construction reaches a fixed point in which application of each rule only duplicates lines already on the tableau. Thus

**Theorem 1.** Constructing a tableau with the tableau rules in Table 4 terminates.

If we apply the tableau rules until no rule yields new information, we end up with a complete description of a model of the ontology in which the source is interpreted (Theorem 2). In the description of the model we use the XML elements of the source and parameters to denote elements of the domain of the model.

**Theorem 2.** Let $T$ be a complete tableau expansion for the ontology $O$ and the source $(u, M)$ mapped to $O$. Then there exists a model $\mathfrak{B}_T$ such that: (i) $\mathfrak{B}_T$ is a model for $O$, (ii) $(u, M)$ is interpreted in $\mathfrak{B}_T$, (iii) the domain of $\mathfrak{B}_T$ consists of the set of parameters and XML elements occurring in $T$ factored by the relation $\{(x, y) \mid x = y$ occurs in $T\}$, (iv) for all concepts $C$, relations $R$ and attributes $a$,

\[
\begin{align*}
\mathfrak{B}_T &\models C(\bar{e}_1) \text{ iff } C(e_1) \text{ on the tableau} \quad (1) \\
\mathfrak{B}_T &\models \bar{e}_1Re_2 \text{ iff } e_1Re_2 \text{ on the tableau} \quad (2) \\
\mathfrak{B}_T &\models a(\bar{e}_1) = n \text{ iff } a(e_1) = n \text{ on the tableau}, \quad (3)
\end{align*}
\]

where $\bar{e}$ denotes the equivalence class of $e$.

Proofs of all results are provided in the Appendix. The tableau rules not only give us a model, we can also use them as an algorithm to yield answers to queries. Here we present a primitive way of doing that. A more efficient calculus following the GAV query unfolding approach is described in the full version of this paper.

---

13 When constructing this model, the order of XML elements in the actual XML document is lost.

14 One has to restrict the application of the concept path and composition rules to exactly one time for each occurrence of the enumerator. Because the parameter is new, applying these rules more often yields no extra information.
Absolute Mapping rule
\[ R_i : u/q as v_i \to C \] is an absolute rule in \( \hat{M} \)
\[ C(e) \] for all \( e \in u/q \)

Relative Mapping rule I
\[ R_i : u/q as v_i \to C \] is an absolute rule in \( \hat{M} \)
\[ R_j : v_i/p as v_j \to R \] is a relative rule in \( M^* \)
\[ eRe' \] for all \( e \in u/q \) and \( e' \in e/p \)

Relative Mapping rule II
\[ R_i : u/q as v_i \to C \] is an absolute rule in \( \hat{M} \)
\[ R_j : v_i/n \to a \] (for \( a \) an attribute) is a relative rule in \( M \)
\[ a(e) = e/n \] for all \( e \in u/q \)

Isa rule
\[ C(e) \quad C \text{ isa } D \]
\[ D(e) \]

Domain rules
\[ eRe' \quad \text{source}(R) = C \]
\[ C(e) \]
\[ a(e) \text{ is defined } \quad \text{source}(a) = C \]
\[ C(e') \]

Target rules
\[ eRe' \quad \text{target}(R) = C \]
\[ a(e) \text{ is defined } \quad a(e) \notin \text{target}(a) \]
\[ C(e') \quad \text{ABORT CONSTRUCTION} \]

Concept path rule
\[ C/R(e) \]
\[ C(n) \]
\[ nRe \]
\[ n \text{ is a new parameter in the tableau expansion.} \]

Composition rule
\[ eR/n \]
\[ eRSe' \]
\[ n \text{ is a new parameter in the tableau expansion.} \]

Inverse rule
\[ eR^{-1}e' \]
\[ e'Re \]

Global key rule
\[ \text{key}(C) = \{a_1, \ldots, a_n\}, C(e), C(e') \]
\[ a_1(e) = a_1(e'), \ldots, a_n(e) = a_n(e') \]
\[ e = e' \]

Reflexivity rule
\[ e = e \]

Replacement rule
\[ \phi(e) \quad e = e' \]
\[ \phi(e') \]

Comparison predicate rule
\[ \text{for every true statement } n \theta m \ (n, m \text{ elements of some basic domain occurring on the tableau.} \]

Table 4. Tableau rules.
The following completeness theorem however is very convenient for proving completeness results later on.

Let \( \phi(\bar{x}) \) be an atomic query. We say that the tableau system gives \( \bar{e}_1 \) as an answer to \( \phi(\bar{x}) \) asked to ontology \( O \) and source \((u, M)\) if there exists parameters and XML elements \( \bar{e}_2 \) such that every conjunct \( \phi_i(\bar{x}, \bar{y}) \) occurs in the tableau with \( \bar{e}_1 \) and \( \bar{e}_2 \) substituted for \( \bar{x} \) and \( \bar{y} \), respectively.

**Theorem 3 (Soundness and completeness of the tableau system).** Given an ontology \( O \), source \((u, M)\) and an atomic conjunctive query \( \phi(\bar{x}) \), the tableau system gives \( \bar{e} \) as an answer to \( \phi(\bar{x}) \) if and only if \( \bar{e} \) is a certain answer to \( \phi(\bar{x}) \).

We end with two technical lemmas which will be used in the next section. The first lemma can be seen as a reason for asking that the paths in the mapping rules are safe. In that case the domain rule of the tableau system becomes redundant.

**Lemma 1.** Let \( T \) be a tableau for the ontology \( O \) and the source \((u, M)\) mapped to \( O \). Let the paths in the mapping rules of \( M \) be safe. Then every occurrence \( C(\bar{e}) \) in \( T \) can be obtained without an application of the domain rule.

The next lemma specifies a situation in which tableau proofs are particularly simple, and in which the universal canonical solution has the convenient property of being a tree.

**Lemma 2.** Let \( O \) be an ontology without inverse roles and without global keys. Let \((u, M)\) be a source in which the paths in the mapping rules of \( M \) are safe. Then

(i) for each conjunctive query \( q(\bar{x}) \), \( \bar{e} \) is a certain answer to \( q(\bar{x}) \) if and only if there is a tableau proof for \( q(\bar{e}) \) which only uses the mapping rule rules, isa, target, concept path and composition rules;

(ii) the universal canonical solution is a tree.

### 5 Lav Approach: Query Rewriting

In this section we discuss the completeness of the rewriting algorithm in [4] in our context. As in [4], we consider the rewriting of tree queries without joins between variables and we use an OQL-like syntax to express them\(^\text{15}\).

The algorithm in [4] needs only rewrite the conjuncts in the from clause of the query. We demonstrate the idea using query \( Q_1 \) shown in Table 1 and the mapping rules of Fig. 3.

In short, the algorithm examines the query variables and finds the mapping rules which provide answers for them. Query variables are arranged in preorder, and the algorithm considers a variable, after it has examined its parent. The root variable \( x_1 \) is examined first. The algorithm looks for absolute mapping rules

---

\(^{15}\) A tree query is a conjunctive query \( q(\bar{x}) : -c(x_0), \land r(x, y), \land a(x) = y, \land \text{ifm} \) satisfying the following restrictions: (i) \( x_0 \) is the root variable of the query, (ii) if \( r(x, y) \) or \( a(x) = y \) occurs in the query, \( x \) is called the parent of \( y \) and the variables with the parenthood relation form a tree with \( x_0 \) as the root.
1. input: tree query
2. repeat until no further reduction is possible
3. do
4. for each $x.R$ y with y a leaf and y not selected, replace $x.R$ y by $x.R/*$ y.
5. for each $x.R$ y with y not selected and y the only child of x then
6. if x is the root variable of the query and C x appears in the query
7. replace C x, $x.R$ y by C/R y (making y the new root).
8. else if z/S x is present in the query, then replace z/S x, $x.R$ y by z/S/R y
9. od
10. output: tree query

Fig. 5. Query preprocessing algorithm

which return instances of concept Person. Such a rule is $R_1$ which states that the elements obtained by evaluating XPath expression Collection/.Painter on the documents in URL http://www.paintings.com are instances of concept Person. These are bound in variable $u_1$. We create the binding $\{(x_1,u_1)\}$ which states that instances of $x_1$ are the instances of $u_1$. In this case the expression Person $x_1$ in $Q_1$ is replaced by http://www.paintings.com/ Collection/ Painter $x_1$. Then, variable $x_2$ is examined. The algorithm looks for a rule which (i) can be evaluated on instances of $x_1$ (i.e. instances of $u_1$) and (ii) whose ontology path is has_name (i.e. the binding path of $x_2$). Such a rule is $R_2$ since its root variable is $u_1$ and its ontology path is has_name. The expression $x_1$.has_name $x_2$ in $Q_1$ is replaced by $x_1./@name$ $x_2$. In a similar manner, expression $x_1.carried_out/produced$ $x_3$ is replaced by $x_1./Painting$ $x_3$ using rule $R_3$ and the binding is extended to $\{(x_1,u_1), (x_3,u_3)\}$. When all the variables have been considered, the obtained query is $Q_1(a)$ illustrated in Table 2. With a simple syntactical transformation\(^\text{16}\), it is transformed into the XQuery expression $Q_1(b)$ shown in Table 2.

There are cases where the algorithm briefly described above, does not return a rewriting even if one exists. The reason is the presence of composite concepts and roles in the query and the mapping rules.

To overcome this problem, we need to do a preprocessing of the query. This step is necessary since the algorithm tries to match all the query variables with some mapping rule. Here we distinguish between necessary and unnecessary variables. The latter appear neither in the select, nor in the where clause of the query. The former must be always instantiated by instances from the sources, the latter can be bound to objects in the model of the ontology which do not exist in the sources. Such objects appear in models of the ontology because of

\(^{16}\) The transformation of query $Q_1(a)$ to the XQuery expression $Q_1(b)$ is done as follows: $Q_1(b)$’s FOR clause is obtained by replacing each expression of the form $x_i/q$ $x_i$ in the from clause of $Q_1(a)$ by (i) $x_j$ IN $x_i/q$ if $x_i$ is a variable and (ii) $x_i$ IN document("u")/q if $x_i$ is of the form $u/q$ where $u$ is a URL. $Q_1(b)$’s WHERE clause contains all conditions in the where clause of $Q_1(a)$. Finally, the RETURN clause of $Q_1(b)$ contains all variables in the select clause of $Q_1(a)$. 

371
role paths and concept paths in the ontology paths of the mapping rules (this
is clearly visible from the tableau composition rules). The function of the query
preprocessing algorithm is to remove all unnecessary variables by rewriting the
initial query into an equivalent one. The query preprocessing algorithm is given
in Fig. 5. The special predicate $x.R/ \neq y$ is an abbreviation of the disjunction
$x.R' y$ for which there is a mapping rule $v_0/q$ as $v_0 \rightarrow R'$ and $R$ is a prefix of $R'$.

Lemma 3. (i) The algorithm in Fig. 5 yields an equivalent query.
(ii) Let $O$ be an ontology without inverse roles. Let $\exists y Q(x, y)$ be the output
from the algorithm in Fig. 5. Then the following are equivalent:
- $\bar{a}$ is a certain answer to $\exists y Q(x, y)$.
- $\bar{ab}$ is a certain answer to $Q(x, \bar{b})$, for some $\bar{b}$.

5.1 Query Rewriting Algorithm

Our aim is to create a simple, efficient and complete rewriting algorithm. For this,
we restrict the queries to tree queries and we do not allow inverse roles, attribute
paths in the mapping rules and the queries and global keys in the ontology. The
mapping rules must be well presented and obey that all XPath expressions $A.B$
in the mapping rules have the property that if there is a document on which $A$
and $B$ have a non empty intersection, then $A$ and $B$ are syntactically the same.
This can be implemented by allowing only the child and attribute axes in the
XPath expressions of the mapping rules. An elaborate motivation (apart from
their use in the completeness proof) of these restrictions is provided in the full
version of the paper [27].

In this section we present the query rewriting algorithm $query2xquery$ illustrated
in Fig. 6 which is a variation of the query rewriting algorithm in [4].
It accepts as input an OQL tree query (preprocessed by the algorithm shown in
Fig. 5) and a mapping $M$. As output it gives an OQL like “XQuery” expression
which has the same select and where clause as the tree query but whose from
part consists of the set $XPool$ in the algorithm.

The algorithm uses the following three sets: $OntPool$ which contains the
expressions in the query from clause, $XPool$ which contains expressions of the
form $x_1/Exp x_2$ and $Exp x_0$ where $Exp$ is an XPath expression, and the $x_1$’s
are query variables and finally $Bindings$ that contains pairs of the form $(x_1, Var)$
where $x_1$ is a query variable and $Var$ is a mapping rule variable. It makes use of
the (non-deterministic) DATALOG procedure $concept2x$ shown in Fig. 7.
Given a concept $c$ from the ontology and a mapping $M$, $concept2x$ returns (i)
the XPath location path and (ii) the bound variable of an absolute mapping rule
$R$ in $M$ which returns elements that are (according to the ontology path of $R$)
instances of $c$.

The algorithm examines first the query root variable $x_0$. Let $C_0 x_0$ be the
expression in $OntPool$ for $x_0$, where $C_0$ is the binding path of $x_0$. Procedure
$concept2x()$ is called with $C_0$ and returns the location path $Exp$ and the bound
variable $Var$ of an absolute mapping rule which returns instances of $C_0$ or of
subconcepts thereof. $(Exp x_0)$ is then added in $XPool$ and $(x_0, Var)$ in the set

\[372\]
begin
OntPool := the from part of the ontology query;
let C0 x0 ∈ OntPool
if concept2x(C0, XExp, Var)
then XPool := {XExp x0}; Bindings := {(x0, Var)};
OntPool := OntPool \ {C0 x0};
else fail;
do OntPool ≠ ∅
let x1, Exp xj ∈ OntPool
  if (x1, vj) ∈ Bindings and vj/XExp as Var → Exp ∈ M*;
    then XPool := XPool ∪ {x1, Exp xj}; OntPool := OntPool \ {x1, Exp xj};
    Bindings := Bindings ∪ {(xj, Var)};
  else fail;
let x1, Exp/ * xj ∈ OntPool
  if (x1, vj) ∈ Bindings and vj/XExp as Var → Exp’ ∈ M*;
      where Exp is a prefix of Exp’
    then XPool := XPool ∪ {x1, Exp xj}; OntPool := OntPool \ {x1, Exp/ * xj};
    Bindings := Bindings ∪ {(xj, Var)};
  else fail;
let a(x1, xj) ∈ OntPool
  if (x1, vj) ∈ Bindings and vj/0XExp → a ∈ M*;
    then XPool := XPool ∪ {x1, 0XExp xj}; OntPool := OntPool \ {a(x1, xj)};
  else fail;
end

Fig. 6. Query rewriting algorithm oquery2xquery.

/* for concepts C and C/Rolepath */
/* if Rolepath is the empty string, “/Rolepath” stands for the empty string */

concept2x(C/Rolepath, U/X, Var):-
P isa C,
U/X as Var → P/Rolepath ∈ R.

concept2x(C/Rolepath, U/X, Var):-
target(Path) isa C,
U/X as Var → Path/Rolepath ∈ R.

Fig. 7. Procedure concept2x.
**Bindings.** Expression $C_0 \, x_0$ is removed from OntPool. The algorithm then examines the query variables in preorder by considering a variable after its parent. Let $x_j$ be the variable considered and let $x_j, \text{Exp} \, x_j$ be an expression in OntPool. Let $x_i$ be bound to variable $v_1$ (i.e. $(x_1, v_1) \in \text{Bindings}$). The algorithm will then look for a relative mapping rule $R$ starting with $v_1$. As far as Exp is concerned, we distinguish between two cases:

- If Exp is of the form $S$, R is selected if its ontology path is equal to $S$;
- if Exp is of the form $S/\ast$ where $S$ is a role path, then R is selected if $S$ is a prefix of R’s ontology path;

Let the relative rule $R$ be of the form $R: v_1/XExp as \text{Var} \rightarrow \text{Exp}$. Then expression $x_1/XExp \, x_j$ is added to XPool, $x_1, \text{Exp} \, x_j$ is removed from OntPool and $(x_j, \text{Var})$ is added to Bindings. If the algorithm examines an expression of the form $a(x_i, x_j)$ where $a$ is an attribute in the ontology, and $x_i$ is bound to variable $v_1$, then the algorithm looks for a relative mapping rule of the form $v_1/@XExp \rightarrow a$. As previously, $x_i/@XExp \, x_j$ is added in XPool and $a(x_i, x_j)$ is removed from OntPool.

We arrived at the central result of the paper:

**Theorem 4.** The query rewriting algorithm presented in Fig. 6 is correct and complete. That is, $\hat{a}$ is a certain answer to the query $q(\bar{x})$ posed to the source $(\bar{u}, M)$ published in $O$ if and only if there exists an “Xquery” $p(\bar{x})$ such that $\text{oquery2xquery}(q(\bar{x}), p(\bar{x}))$ is true and $\hat{a}$ is an element of the answer set of $p(\bar{x})$ evaluated on source $u$.

### 6 Conclusions

In this paper we have shown completeness and correctness of the query rewriting algorithm proposed in [4] in a restricted setting. That work proposed a novel framework for query mediation over a set of XML resources using an ontology-based mediator. The ontology is an ER schema with subsumption relations between concepts. The sources are mapped to this ontology by means of mapping rules which associate XPath location paths to ontology paths. The main problem for a query rewriting algorithm in this setting is to map node labeled trees (XML documents) to edge labeled graphs with inverse roles and global keys. Leaving out these last two features from the ontology makes that the universal canonical solution has the shape of a tree, a much simpler structure to work with. The mapping language proposed in [4] is very rich: all XPath 1.0 location paths are allowed. This introduces problems for the proposed algorithm which binds query variables to mapping rule variables representing sets of XML elements. Such an algorithm can only be complete if for each XML element there is a unique location path leading to it in the mapping rules. This explains our restriction to location paths using only the child and attribute XPath axes in the mapping rules. The restrictions we had to pose seem huge but are still reasonable for practical applications (cf [13] for an argument). Our main contribution thus is that mediation in a setting with two very different data formats is feasible in
a simplified but non-trivial case. The main challenge for future work is to cope with inverse roles and global keys in a computationally attractive way as well as to raise the limitations of using only the previously mentioned XPath axes in the mapping rules. Especially efficient algorithms which reason about equalities between data from different XML documents will be needed. Moreover, another axis of research is to introduce XML Schemas in the place of XML DTDs. The other contributions of the paper are a side effect of our effort to prove completeness of the algorithm. They are: (i) the definition of an interpretation function which maps fragments of XML sources into models of the ontology (ii) the notion of certain answer in this context, (iii) a tableau calculus for creating a canonical model (a data warehouse) for the source data and (iv) using this tableau calculus to design other algorithms for query answering.

References


375
A Proofs

Proof (Theorem 2). Construct $\mathfrak{B}_T$ as specified in the theorem. By replacement, $=$ is a congruence, whence the valuation of the concepts, roles and attributes is well defined. $\mathfrak{B}_T$ is a model for $O$ by the isa, source, target and global key rules.

Now let $f$ be the mapping from $u$ to the domains of $\mathfrak{B}_T$ defined by $f(e) = \bar{e}$ for elements and $f(n) = n$ for attribute values.

We now show that $f$ respects the rules of $M^*$. This is immediate for atomic concepts, roles and attributes by the tableau rules. For the paths, we need the following result for all concept paths $C$ and role paths $R$: (i) if $C(e_1)$ on the tableau, then $\mathfrak{B}_T \models C(e_1)$; (ii) if $e_1Re_2$ on the tableau, then $\mathfrak{B}_T \models e_1Re_2$, which is proved by a simple induction, using the concept path and composition rules.

We note that the other direction only holds in the following sense: $\mathfrak{B}_T \models e_1R_1 \ldots R_n\bar{e}_2$ only if there exists $a_1, \ldots, a_{n-1}$ such that all of $e_1R_1a_1 \ldots a_{n-1}R_n\bar{e}_2$ are on the tableau, and similarly for the concept paths.

Proof (Theorem 3). Suppose $\bar{e}_1$ is a certain answer to $\phi(\bar{x})$. Then in any model $\mathfrak{B}$, $\mathfrak{B} \models \phi(\bar{e}_1)$. In particular, $\mathfrak{B}_T \models \phi(\bar{e}_1)$. By assumption, all conjuncts of $\phi$ are atomic. But then by Theorem 2, all these conjuncts appear on the tableau. But then the tableau procedure yields $\bar{e}_1$ as an answer.

For the other direction, we need some more work. It is easy to see that all tableau rules are sound: that is, if the condition of the rule is true for a model of $O$ in which $(u, M)$ is interpreted, then the conclusion as well. To continue, we say that the tableau system gives $\bar{e}_1$ as an answer to $\phi(\bar{x})$ asked to ontology $O$ and source $(u, M)$ if the tableau system with the additional following query rule closes: (here $\phi_1 \ldots \phi_n$ are all atomic conjuncts of $\phi$)

\[
\phi(\bar{x}) \text{ query rule } \begin{align*}
\quad \phi_1(\bar{e}_1, \bar{e}_2) & \quad \sim \phi_2(\bar{e}_1, \bar{e}_2) & \quad \ldots & \quad \sim \phi_n(\bar{e}_1, \bar{e}_2) \\
\end{align*}
\]

The vertical bars in the tableau rule indicate a branching of the tableau. A branch of a tableau closes if it contains a formula and its negation. A tableau closes if all its branches are closed.

Clearly if the tableau yields $\bar{e}_1$ as an answer without the new rule, $\bar{e}_1$ is also an answer with the added rule. Now suppose that $\bar{e}_1$ is not a certain answer to $\phi(\bar{x}, \bar{y})$. Then for any choice $\bar{e}_2$ for $\bar{y}$, there is a model in which one of the conjuncts $\phi(\bar{e}_1, \bar{e}_2)$ is false. So, applying the query rule yields a satisfiable situation. But since all rules, preserve satisfiability, there must be an open branch, whence the tableau system does not yield $\bar{e}_1$ as an answer.

Proof (Lemma 1). Suppose $T$ contains an application of the domain rule

\[
\begin{align*}
\text{eRe} & \quad \text{source}(R) = C \\
& \quad C(e) \\
\end{align*}
\]

Then $\text{eRe}$ must have come from an application of (a) the concept path rule, (b) the relative mapping rule or (c) the composition rule. In case (a) we had $D/R(e')$ for some concept $D$. But by assumption, the path $D/R$ is safe, whence $D \text{ isa source}(R)$ holds. But then an application of the isa rule yields the desired conclusion $C(e)$.
In case (b) there are two rules $R_i : u/q \rightarrow D$ and $R_j : v_i/q' \rightarrow R$. As the paths in $M$ are safe, it must be that $D$ is $\text{source}(R)$. So to derive $\text{source}(R)(e)$ apply the absolute mapping rule with rule $R_i$ to derive $D(e)$ and then the isa rule to derive $\text{source}(R)(e)$.

In case (c) $eRe'$ came from an application of the composition rule. If the relation $R$ is always the first on a path $eR/Pn$ then this path must have originated from the concept path rule or the relative mapping rule rule, and we can reason as before. Otherwise, we had the following application of the composition rule to get $eRe'$:

$$\frac{nP/Re'}{nP,eRe'}$$

It is easy to show that all paths occurring in the tableau must be a subpath of a path which occurs in the conclusion of some rule in $M$. Whence by assumption then $\text{target}(P)$ is $\text{source}(R)$. But now we derive $\text{target}(P)(e)$ from $nP\epsilon$ by the target rule, and from that the desired $\text{source}(R)(e)$ with the isa rule.

**Proof (Lemma 2).** For (i), assume the hypothesis of the lemma. By the completeness theorem $q(\bar{a})$ is an answer iff there exists a tableau proof for $q(\bar{a})$. By Lemma 1, the domain rule is not needed in this proof. Obviously the inverse and the global key rules are not needed in a tableau proof as their antecedent is never true. The two rules for reasoning about equality are not needed because 1) in queries = cannot be used to relate domain elements, and 2) without global keys no statement of the form $e = e'$ is produced in a tableau.

(ii) is immediate by (i), the fact that the composition and concept path rules use new parameters and Theorem 2.

**Proof (Lemma 3).** (i) Immediate by the meaning definition. (ii) By an inspection of the tableau rules yielding the universal canonical solution and Lemma 1.

**Proof (Theorem 4).** For ease of reference we list once more the restrictions imposed:

- **C1**: Ontologies contain neither inverse roles nor global keys;
- **C2**: All mapping rules are well presented;
- **C3**: The location paths of the mapping rules use only the child and attribute XPath axes.

Assume a source $(u, M)$ and an ontology $O$ are fixed. Throughout we assume that $(u, M)$ is published in $O$ and that it satisfies conditions **C1–C3**.

First define a partial function $b$ from the elements of $u$ to the set of variables occurring in $M$ as follows:

$$b(e) = v \text{ iff there exists a } X \text{ and } C \text{ such that } u/X \text{ as } v \rightarrow C \in \hat{M} \text{ and } e \in X.$$  

To see that $b$ is a function assume that there are $X, X', C, C'$ such that $u/X \text{ as } v \rightarrow C, u/X' \text{ as } v' \rightarrow C' \in \hat{M} \text{ and } e \in X \cap X'$. Then by the constraints mentioned above $X = X'$, whence by $C2 v = v'$. In the sequel, as usual, if $b(e)$ occurs in a statement it is assumed that $b$ is defined on $e$.

The next lemma proves the theorem for the four different kinds of conjuncts occurring in queries.

**Lemma 4.** 1. For any role path $P$, the following are equivalent

(i) $e_i, e_j$ is a certain answer to $x_i, P x_j$;
(ii) for some $X$, there is a rule $b(e_i)/X as b(e_j) \rightarrow P$ and $e_j \in e_i/X$. 

378
2. For any role path $P/\ast$, the following are equivalent
   (i) $e_i, e_j$ is a certain answer to $x_i.P/\ast x_j$;
   (ii) for some $X$ and role path $R'$, there is a rule $b(e_i)/X$ as $b(e_j) \rightarrow R'$ and $e_i \in e_i/X$ and $P$ is a prefix of $R'$.
3. For any concept path $C/R$ (where $R$ can be the empty path), the following are equivalent
   (i) $e$ is a certain answer to $C/R(x)$;
   (ii) for some XPath expression $u/X$ the procedure concept2x($C/R, u/X, b(e)$) succeeds and $e \in u/X$.
4. For any attribute $a$, the following are equivalent:
   (i) $e$ is a certain answer to $a(x) = n$;
   (ii) for some XML attribute $X$ there exists a rule $b(e)/\@X \rightarrow a \in M^*$ and $n \in e/\@X$.

*Proof (Lemma 4).* In all proofs we use the fact that the certain answers can be computed by the tableau algorithm using only the mapping rule rules, isa, target, composition and concept path rules (Theorem 3 and Lemma 2).

1. The direction from (ii) to (i) is straightforward: since $b(e_i)$ is defined there exists an absolute rule $u/q$ as $b(e_i) \rightarrow C$. This together with the rule $b(e_i)/X$ as $b(e_j) \rightarrow P$ and the fact that $e_j \in e_i/X$ yields $e_iPe_j$ by the relative mapping rule rule.
   For the other direction we proceed by induction on the length of $P$. Let $|P| = 1$.
   The relative mapping rule rule is the only rule which produces statements of the form $a/Pb$ for $a, b$ elements in the source. Whence $e_iPe_j$ must have been produced by an application of this rule. Then there are rules $u/Y$ as $v \rightarrow C$ and $e_i \in u/Y$ and $u/X$ as $w \rightarrow P$ and $e_j \in e_i/X$. But then $v = b(e_i)$ and $w = b(e_j)$, by definition of $b$.
   For the induction step, let $|P| = n + 1$. If the tableau proves $e_iPe_j$ then it is obtained by a direct application of the relative mapping rule. In this case we have the desired result, as before. Otherwise the tableau proves $e_iP\ast n$ and $nP_2e_j$ for $n$ an element and $P_1/P_2 = P$. By inductive hypothesis then the following two rules are in $M^*$:
   \[
   b(e_i)/X_1 \rightarrow P_1, \quad (n)/X_2 \rightarrow P_2,
   \]
   and $n \in e_i/X_1$ and $e_j \in n/X_2$. But then also $b(e_i)/X_1/X_2$ as $b(e_j) \rightarrow P_1/P_2 \in M^*$, and $e_j \in e_i/X_1/X_2$ as desired.

2. By the previous item and the fact that $x.R/\ast y$ is an abbreviation of the disjunction $x.R' y$ for which there is a mapping rule $v_i/q$ as $v_j \rightarrow R'$ and $R$ is a prefix of $R'$.

3. It is easy to see that (ii) is equivalent to
   (iii) for some XPath expression $u/X$, there exists an absolute rule $u/X$ as $b(e) \rightarrow C/R$ in $M$ and $C' \isa C$ and $e \in u/X$, or there exists a concept path $C'$ and an absolute rule $u/X$ as $b(e) \rightarrow /R$ in $M$ and target($C'$) $\isa C$ and $e \in u/X$.
   We now prove that (iii) is equivalent to (i). The direction from (iii) to (i) is immediate by an inspection of the tableau rules. The other direction is again by an induction on the length of the role path $R$. For $|R| = 0$, the statement follows directly from an inspection of the tableau rules and Lemma 2. The inductive case is proved using the first item of this lemma by a similar argument as used in the proof of that item.

4. The statement for attributes is immediate from the tableau rules.
Now we are ready to prove the theorem. By Lemma 3 we may assume without loss of
generality that there are no quantified variables in the query. First we prove correctness.
Let \( p(\bar{x}) \) be the “XQuery” and \( s(\bar{x}) \) the answer. We show that \( s(\bar{x}) \) is an answer to
\( q(\bar{x}) \) as well. Let \( B \) be the set of bindings produced by the algorithm. By construction
of the algorithm, for all \((x_i,b_i) \in B\), \( b \circ s(x_i) = v_i \). The query consists of four kinds
of conjunctions. Here we show for the relational expressions that \( s(\bar{x}) \) is an answer to those.
The other expressions have a similar proof. Let \( x_i/\text{Exp } x_j \) be a conjunct in \( p(\bar{x}) \). Thus
\( s(x_j) \in s(x_i)/\text{Exp} \). This holds if \( x_i \cdot \text{Exp } x_j \) in the ontology query \( q(\bar{x}) \) and there is a
mapping rule \( v_i/\text{Exp as } v_j \rightarrow \text{Exp in } M^* \). Whence \( b \circ s(x_i)/\text{Exp as } b \circ s(x_j) \rightarrow \text{Exp}
in \( M^* \). Moreover \( s(x_j) \in s(x_i)/\text{Exp} \) by assumption. Whence by the previous Lemma,
\( s(x_i) \cdot \text{Exp} \circ s(x_j) \) holds in the universal canonical solution.

Conversely, let \( s(\bar{x}) \) be a certain answer to the ontology query \( q(\bar{x}) \). We show that
the algorithm produces a rewriting which has \( s(\bar{x}) \) in its answer set. Let the set of bindings \( B \) be \( \{(x_i, v_i) \mid b \circ s(x_i) = v_i \} \). By Lemma 4, for every conjunct of \( q(\bar{x}) \), there
exists a corresponding mapping rule. For instance, if \( x_i \cdot \text{Exp } x_j \) is a conjunct, then there
is a mapping rule \( b \circ s(x_i)/\text{Exp as } b \circ s(x_j) \rightarrow \text{Exp in } M^* \) and \( s(x_j) \in s(x_i)/\text{Exp} \).
Choose for each conjunct a mapping rule. With this choice the algorithm succeeds with
the defined bindings and produces an “XQuery” \( p(\bar{x}) \) with \( s(\bar{x}) \) in its answer set.