Data Integration for Querying Geospatial Sources[⋆]

Isabel F. Cruz and Huiyong Xiao

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

Abstract. Geospatial data management is fundamental for many applications including land use planning and transportation and is critical in emergency management. However, geospatial data are distributed, complex, and heterogeneous due to being independently developed by various levels of government and the private sector. Until now, the formulation of expressive queries on geospatial data, which contrast with simple keyword-based queries, requires both user expertise and a great deal of manual intervention to determine the mappings between concepts in potentially dozens of data sources.

In this paper, we describe an ontology-based approach to the problem of data integration, specifically focusing on the issue of query processing in a heterogeneous setting. Our contributions include a mechanism for metadata representation, an ontology alignment process, and a sound query rewriting algorithm for answering queries across distributed geospatial data sources. We demonstrate the practical impact of our approach in land use applications, which are exemplary of the extreme heterogeneity of data. We are leveraging current and emerging Semantic Web standards and tools for modeling, storing, and processing data. Our contribution to geospatial data integration is significant because new data sources can be added with relatively little effort, thus allowing for data manipulation and querying to extend seamlessly to the new data sources.

1 Introduction

Years of autonomic and uncoordinated development of classification schemes by government organizations and the private sector pose enormous challenges in integrating geospatial data. In this paper, our focus is on the integration of geospatial data that is created by the different counties and municipalities in the state of Wisconsin and stored locally by the counties and municipalities. Among the available geospatial data, we have concentrated on land use data. We have worked with the Wisconsin local government within the scope of WLIS (Wisconsin Land Information System) and the National Science Foundation under their Digital Government and Information Integration programs. Data heterogeneity

[⋆] This research was partially supported by the National Science Foundation under Awards ITR IIS-0326284 and IIS-0513553.

in the land use data domain has been hindering the cooperation among the local governments to achieve comprehensive land use planning across the borders of the different jurisdictions [33].

We propose an ontology-based approach to enable integration and interoperability of the local data sources. In our work we deal with two kinds of ontologies: an axiomatized set of concepts and relationship types and a taxonomy of entities [17]. We call the first kind *schema-like ontologies*, because they are associated with the structure or *schema* of the local sources. The second kind model the entities (instance names) that describe land usage (for example, *agricultural, commercial*, or *residential*), where the only type of relationship is that of subconcept (or subclass) (for example, *single family residences* and *multiple family residences* are two subconcepts of *residential*), and are called *taxonomy-like ontologies* in our discussion.

The ontologies that we use to represent the structure of the local data sources, which we call *local ontologies*, belong to the first type and can be obtained from the source schemas through a schema transformation process. The second type of ontologies are the *land use ontologies*, which are part of the local ontologies; they represent land use taxonomies that classify land parcels in the local data sources according to their usage. In addition, our approach uses a *global* or *domain* ontology that models the domain associated with the task at hand and enables mediation across the local data sources. The global ontology contains a global land use ontology that describes the land use domain.

The key to our approach lies in establishing mappings between the concepts of the global ontology and the concepts of the local ontologies, a process called *alignment*. Using those mappings, a single query can then be expressed in terms of the concepts of the global ontology (or of a local ontology) and be automatically *rewritten* and posed against the other ontologies. We focus on database-style queries as opposed to simpler and less expressive keyword-based queries.

Query processing can be performed in two ways: *global-to-local* and *local-to-local*. In the former case, we rewrite a query posed on the global ontology into subqueries over the local sources (the global ontology acts as a uniform query interface of the integration system). In the latter case, we translate a query posed on a geospatial source to a query on any other geospatial source.

In this paper, we consider the alignment process of the local land use ontology with the global land use ontology and propose an ontology alignment algorithm based on a set of deduction rules, which can be performed automatically when certain pre-conditions are established. We propose a *sound* query rewriting algorithm. The algorithm can compute a *contained* rewriting of a query in both global-to-local and local-to-local querying. Query containment ensures that all the answers retrieved by executing the rewriting are a subset of the answer to the original query, thus guaranteeing precise query answering across distributed data sources [23].

The rest of the paper is organized as follows. The data heterogeneity issues in land use management are discussed in Section 2. The ontology creation process is described in Section 3. In Section 4, we focus on an automatic algorithm for ontology alignment. Query answering is presented in Section 5. In Section 6, we describe briefly the user interfaces that support ontology alignment and query processing. We summarize related work in Section 7. Finally, we draw conclusions and outline directions for future research in Section 8.

2 Data Heterogeneities

In this section, we describe in detail the kinds of heterogeneities that we encounter when integrating data from the local geospatial sources. In these sources, data is stored in XML format. Figure 1 shows two fragments of land parcel data, including their DTD (on the left-hand side) and an XML fragment (on the right-hand side), which respectively exist in the local systems of Eau Claire County and Madison County. As we can observe, even though the local XML sources present different structures and naming conventions, they share a common domain with closely related meanings (or semantics), thus being ideal candidates for an integration system.

The previous examples display *syntactic homogeneity* in that they both use XML but have different structures, therefore displaying *schematic heterogeneity*. They may also encode their instances or values in different ways, thus displaying *semantic heterogeneity*, in the sense that the same values may represent different meanings and that different values may have the same meaning [32]. Our discussion elaborates further on both kinds of heterogeneities. In the example shown in Figure 1, we see that the two source schemas overlap on most elements and both have the same nesting depth. However, the elements of the land use codes are represented differently in the two schemas: the schema S_1 uses four elements (`broad`, `lu1`, `lu2`, and `lu3`), whereas S_2 uses a single element (`land_use`). Furthermore, the values of such land use codes (in the XML instances) are encoded in different ways, namely characters for S_1 and numbers for S_2.

Land use codes in WLIS stand for land use types (or categories) and include, for example, *agriculture*, *commerce*, *industry*, *institutions* and *residences*. Besides using different names in different local source schemas, such land use codes have different classification schemes associated with them, thus resulting in semantic heterogeneities across the local source schemas. This is illustrated by Table 1, where there are four element names (`Lucode`, `Tag`, `Lu1` and `Land_use`) from four different classification schemas. The descriptions in the table show that different values represent closely related land use types.

In our approach, a *local ontology* is generated for each local XML source that represents its schema. In addition, a *global* or *domain* ontology is defined to act as an integrated view and a uniform access interface to the distributed data sources. Every local ontology is mapped to this global ontology, by establishing the correspondences of their elements and attributes, which results in an alignment on the local names. In addition to this schema level reconciliation, it is also necessary to have a global land use taxonomy, to which the local land use taxonomies are mapped, so as to achieve a common understanding of the se-

```
<?xml encoding="ISO-8859-1"?>           <LandUse>
<!ELEMENT LandUse (LandParcel)>           <LandParcel>
<!ELEMENT LandParcel (AREA, BROAD, LU1,     <AREA>1704995.587470</AREA>
  LU2, LU3, ..., JurisType, JurisName)>     <BROAD>A</BROAD>
<!ELEMENT AREA (#PCDATA)>                   <LU1>AF</LU1>
<!ELEMENT BROAD (#PCDATA)>                  ......
<!ELEMENT LU1 (#PCDATA)>                    <JurisType>County</JurisType>
......                                      <JurisName>EauClaire</JurisName>
<!ELEMENT JurisType (#PCDATA)>            </LandParcel>
<!ELEMENT JurisName (#PCDATA)>              ......
                                        </LandUse>
```

a) Local XML data source S_1 of Eau Claire County.

```
<?xml encoding="ISO-8859-1"?>           <LandUse>
<!ELEMENT LandUse (LandParcel)>           <LandParcel>
<!ELEMENT LandParcel (AREA, LAND_USE,       <AREA>1007908.5</AREA>
  PARCEL_ID, ..., JurisType, JurisName)>    <LAND_USE>9100</LAND_USE>
<!ELEMENT AREA (#PCDATA)>                   <PARCEL_ID>246710</PARCEL_ID>
<!ELEMENT LAND_USE (#PCDATA)>               ......
<!ELEMENT PARCEL_ID (#PCDATA)>              <JurisType>County</JurisType>
......                                      <JurisName>Madison</JurisName>
<!ELEMENT JurisType (#PCDATA)>            </LandParcel>
<!ELEMENT JurisName (#PCDATA)>              ......
                                        </LandUse>
```

b) Local XML data source S_2 of the City of Madison.

Fig. 1. Local XML land use data sources. In the data source S_1, `BROAD` and `LU1` define the land use code, with `BROAD` as the first level and `LU1` as a child level of `BROAD`. In the data source S_2, `LAND_USE` specifies the land use code. In both sources, the elements `JurisType` and `JurisName` contain the jurisdiction type and name, respectively.

mantics of the land use codes in the local sources. All ontologies are represented using RDF and RDFS.

3 Ontology Creation

The first step of the integration of XML geospatial data sources is the transformation from the XML source schema and data to an RDFS ontology and to RDF data. This transformation encompasses the following steps:

Element-level transformation This transformation defines the basic classes and properties of the local RDFS ontology according to the transformation correspondences shown in Table 2, with the structural relationships between the elements not being considered for the time being. No new RDF metadata need be defined here because `rdfs:Class` and `rdfs:Property` are sufficient to express classes and properties. For instance, to transform the DTD of S_1

Table 1. Semantic heterogeneity resulted from different encodings of land use data.

Local Source	*Element Name*	*Land Use Type Value*	*Description*
Dane County RPC	Lucode	91	Cropland Pasture
Racine County (SEWRPC)	Tag	811	Cropland
		815	Pasture and Other Agriculture
Eau Claire County	Lu1	AA	General Agriculture
City of Madison	Land_use	8110	Farms

in Figure 1, we define two classes: `LandUse` and `LandParcel` for the elements with the same name. The other elements become properties of `LandParcel`, because they are simple-type subelements.

Table 2. Element-level transformation

XML Schema concepts	**RDF Schema concepts**
Attribute	Property
Simple-type element	Property
Complex-type element	Class

Structure-level transformation This transformation encodes the nesting structure of the XML schema into the local RDFS ontology [11]. In particular, nesting may occur between two complex-type elements or between a complex-type element and its child (as a simple-typed 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`. 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. Table 3 lists the correspondences between the XML elements and the classes or properties in the local RDFS ontology.

As an example, Figure 2 shows the local ontologies (represented as graphs where nodes are classes and edges are properties) transformed from the XML schemas in Figure 1. The land use taxonomies are transformed into a hierarchy of classes and incorporated as part of the local ontologies, rooted from `LandUseTag` and `LandUseType`, respectively.

4 Ontology Alignment

The ontology alignment process takes as input a local ontology and the global ontology and produces the class and property correspondences between them. We must consider two cases, which correspond to the schema and taxonomy

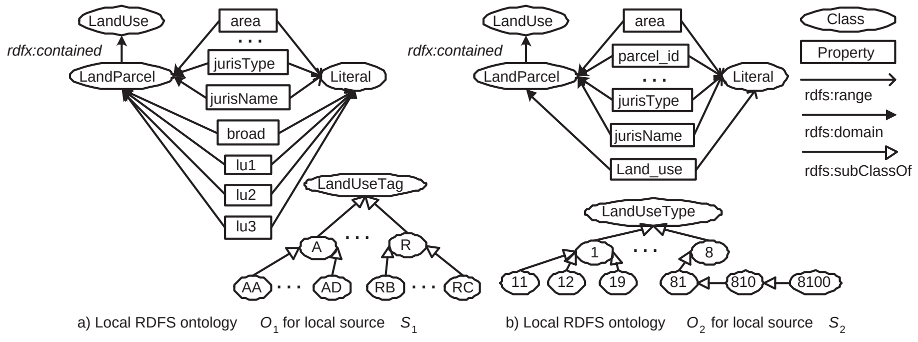


Fig. 2. An example of local RDFS ontologies.

Table 3. Mappings between local XML schema D_1 and local RDFS ontology O_1

XPath expressions in D_1	**RDF expressions in O_1**
/LandUse	LandUse
/LandUse/LandParcel	LandParcel
/LandUse/LandParcel/AREA	LandParcel.area
/LandUse/LandParcel/BROAD	LandParcel.broad
/LandUse/LandParcel/LU1	LandParcel.lu1
...	...
/LandUse/LandParcel/JurisType	LandParcel.jurisType
/LandUse/LandParcel/JurisName	LandParcel.jurisName

components in the global ontology and local ontologies (see Figure 2): 1) *schema-level* mapping between the schema parts of two ontologies, where a concept (or a role) of one ontology is mapped to a concept (or a role) of another ontology, and 2) *instance-level* mapping, where two corresponding concepts use two different classification schemes for their instances, that is, land use codes with different underlying taxonomies in WLIS.

Ontology alignment is in general a challenging task, with its degree of difficulty depending on the types of ontologies being considered [30]. In our framework we have two kinds of ontologies and therefore two kinds of mappings, instance-level mappings and schema-level mappings. The former applies to two taxonomy-like ontologies consisting of only `subClassOf` relationships and the latter to two schema-like ontologies containing various properties and relationships (schema-level mappings). Next, we describe in detail the instance-level mappings and give an example with both schema-level and instance-level mappings.

Mapping types Figure 3 shows a fragment of two concrete land use taxonomies: the one on the left hand side is from the local ontology O_1 in Eau Claire County (as depicted in Figure 2) and the one on the right hand side is from the global ontology G.

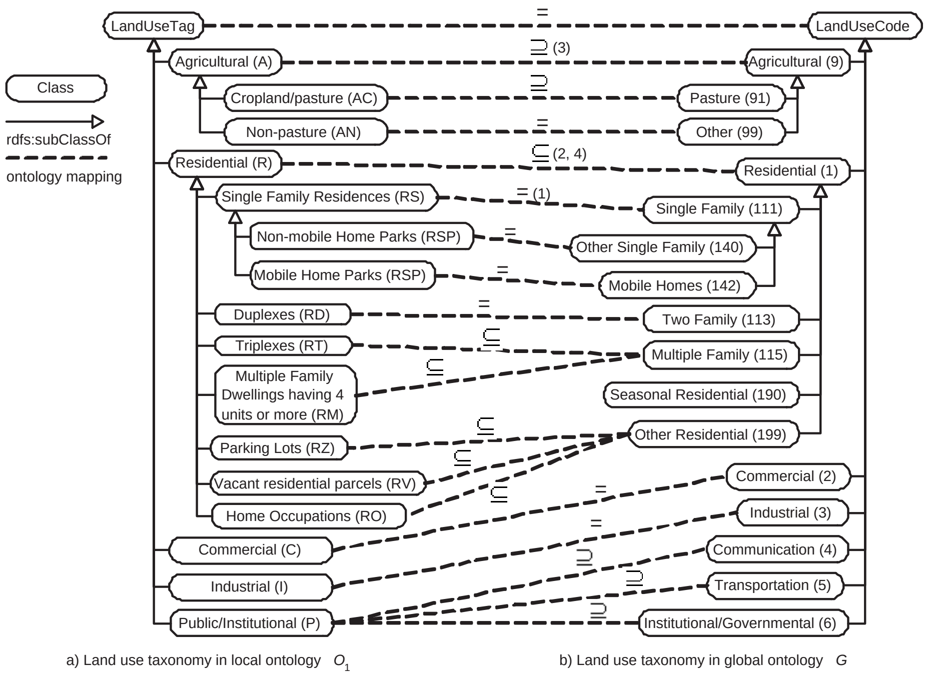


Fig. 3. An example of mapping between two land use taxonomies. The labels over the edges represent mappings types, followed (in between parentheses) by the deduction rule(s) that can be applied, if any.

The two taxonomies are respectively rooted from `LandUseTag` and from `LandUseCode`. A node in each taxonomy represents a class of land use, where the label contains its description and the code (in parenthesis). The dashed lines represent the mappings that are established based on the semantics of the classes. We consider the following types in ontology mappings:

Semantic relationships Considering a set-theoretic semantics, the mapping between two classes A and B (seen as two sets of instances) can be classified into five categories: *superclass*, *subclass*, *equivalent*, *approximate* (or *overlapping*), and *disjoint*, respectively, $A \supseteq B$, $A \subseteq B$, $A = B$, $(A \cap B \neq \emptyset) \wedge (A - B \neq \emptyset) \wedge (B - A \neq \emptyset)$, and $A \cap B = \emptyset$.

Cardinality Class correspondences are established pairwise between two ontologies (producing *one-to-one mappings*). However, it is possible that a class from one ontology is mapped to multiple classes from the other ontology, in a *many-to-one* mapping and that multiple classes are mapped to a single class, in a *one-to-many* mapping. To express such mappings we consider the union of the classes to which a single class (in the other ontology) maps. For example, given two mappings $A = B$ and $A = C$, we have that $A = B \cup C$.

Coverage We distinguish two types of mappings: *fully covered* and *partially covered*. Let C and C' be two classes to be mapped, such that $C_1, ..., C_m$ are subclasses of C, and $C'_1, ..., C'_n$ are subclasses of C'. We say that C (resp. C') is *fully covered* if for each child $C_i \in \{C_1, ..., C_m\}$ (resp. for each child $C'_j \in \{C'_1, ..., C'_n\}$) there is a non-empty subset of $\{C'_1, ..., C'_n\}$ to which C_i is mapped (respectively there is a non-empty subset of $\{C_1, ..., C_m\}$ to which C'_j is mapped).

Deduction process In our approach, the ontology mapping process is performed using an inference process based on deduction rules. In the case that the deduction rules do not apply, then manual intervention by the user is needed.

This semi-automatic ontology mapping process follows two principles: (1) The deduction of the mapping between two nodes (from both taxonomies being mapped) is determined by the mappings between their children. In other words, the mapping between two ontologies are performed in a level-wise fashion, driven by the deduction rules that are defined based on the mapping semantics. (2) The user intervention is needed in two cases: when the mapping between two nodes has insufficient information to determine its type (for example, when some of the children of one node have not been mapped) or when there is conflicting information (for example, that a node is both a superset and a subset of the corresponding node).

We make the *complete-partition assumption*: for any class C in the taxonomy, its subclasses $C_1, ..., C_n$ form a complete partition of the class, that is, $C = C_1 \cup ... \cup C_n$. For instance, in the global taxonomy depicted in Figure 3, the two children `Pasture(91)` and `Other(99)` of the `Agricultural(9)` class form a complete partition of `Agricultural(9)`, since `Other(99)` includes all agricultural lands that are not used for pasture.

We consider the following deduction rules:

Definition 1 (Deduction rules). *Let C and C' be two fully covered classes, and $C_1, ..., C_m$ and $C'_1, ..., C'_n$ be the subclasses of C and C', respectively. Then, the mapping between C and C' can be obtained according to the following rules:*

1) $C = C'$, if for each $C_i \in C$, C_i is mapped to some k-element subset C'' of $\{C'_1, ..., C'_n\}$ $(1 \leq k \leq n')$, such that $C_i = \bigcup_{l=1}^{k} C''_l$.
2) $C \subseteq C'$, if for each $C_i \in C$, C_i is mapped to some k-element subset C'' of $\{C'_1, ..., C'_n\}$ $(1 \leq k \leq n')$, such that $C_i = \bigcup_{l=1}^{k} C''_l$ or $C_i \subseteq \bigcup_{l=1}^{k} C''_l$.
3) $C \supseteq C'$, if for each $C_i \in C$, C_i is mapped to some k-element subset C'' of $\{C'_1, ..., C'_n\}$ $(1 \leq k \leq n')$, such that $C_i = \bigcup_{l=1}^{k} C''_l$ or $C_i \supseteq \bigcup_{l=1}^{k} C''_l$.

The deduction rules in Definition 1 can be proved to be *sound* and *complete* by an induction on the set-theoretic semantics of each rule, under the complete-partition assumption and the assumption that the user-defined mappings are semantically correct.

The above rules assume a full mapping between C and C'. However, they still hold for the case of a partial mapping, provided that we define the following supplemental rule: *4) Suppose that a class C is partially covered by C' and that*

S is the subset of subclasses of C that are not mapped to any children of C'. Then, we create a temporary and empty subclass $\perp$ of C', and add a superclass *mapping from each class in S to $\perp$.*

In Figure 3, the symbols and numbers (in between parentheses) over the dashed lines (i.e., the class correspondences) indicate the mapping type and the adopted inference rule(s), respectively. For example, "$\subseteq$ $(2, 4)$" over the mapping between the class `Residential(R)` and `Residential(1)` means that `Residential(R)` is a superclass of `Residential(1)`, which is computed by rules 2 and 4. The application of rule 4 is due to the fact that `SeasonalResidential (190)` is unmapped, thus making `Residential(1)` partially covered.

Mapping representation The ontology mappings that result from matching the ontologies are stored in a file, called the *agreement* file. We use RDFS to express such mappings. Owing to the multiple inheritance feature of RDFS classes, the RDF property `rdfs:subClassOf` can be used in representing the three different kinds of mappings that may relate two classes A and B, namely $A \supseteq B$, $A \supseteq B$, and $A = B$, in the taxonomy-like components of the two ontologies. For example, the first case is represented by the following RDFS segment:

```
<rdfs:Class rdf:about="A">
   <rdfs:subClassOf rdf:Class="B"/>
</rdfs:Class>
```

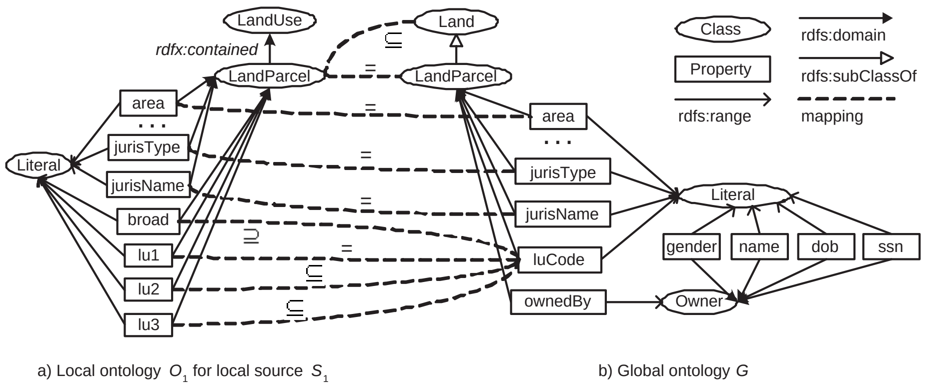
The second kind ($A \supseteq B$), will be represented in the same way, by considering $B \subseteq A$. Finally, the third kind of mapping ($A = B$) will be represented simultaneously by the two different ways in which $A \supseteq B$ and $A \subseteq B$ are represented.

Regarding the mappings between the schema-like components of the two ontologies, in addition to relationships between classes, relationships between properties need to be expressed, namely *superproperty*, *subproperty*, or *equivalent (property)* mappings. Similarly to the previously described use of `rdfs:subClassOf` to represent class mappings, we can use the RDF property `rdfs:subPropertyOf` to represent these property mappings. Figure 4 shows a fragment of the RDFS representation of the mappings between the global ontology G and the local ontology O_1, which include the mappings between the schema-like and the taxonomy-like components of the ontologies.

5 Query Processing

A query such as "*Where are all the multiple family land parcels in Wisconsin?*" cannot be currently answered without manual rewriting this query for each of the dozens of local data sources. In this section, we describe how such queries can be automatically rewritten by our integrated system, using the agreement files that are generated by the alignment process.

The above query, if posed over the global ontology, can be expressed by the following RQL [7] expression:



```
<!DOCTYPE rdf:RDF [ <!ENTITY G "urn:ontologies-advis-lab:global-ontology#">
   <!ENTITY O1 "urn:ontologies-advis-lab:local-ontology-1#">
   <!ENTITY O2 "urn:ontologies-advis-lab:local-ontology-2#"> ]>
<rdfs:Class rdf:about="&G;LandParcel">
   <rdfs:subClassOf rdf:Class="&O1;LandParcel"/>
</rdfs:Class>
<rdfs:Class rdf:about="&O1;LandParcel">
   <rdfs:subClassOf rdf:Class="&G;LandParcel"/>
   <rdfs:subClassOf rdf:Class="&G;Land"/>
</rdfs:Class>
......
<rdf:Property rdf:about="&O1;lu2">
   <rdfs:domain rdf:Class="&O1;LandParcel"/>
   <rdfs:range rdf:Class="rdfs:Literal"/>
   <rdfs:subPropertyOf rdf:Property="&G;luCode"/>
</rdf:Property>
......
<rdfs:Class rdf:about="&O1;RT">
   <rdfs:subClassOf rdf:Class="&G;115"/>
</rdfs:Class>
<rdfs:Class rdf:about="&O1;RM">
   <rdfs:subClassOf rdf:Class="&G;115"/>
</rdfs:Class>
......
```

Fig. 4. A fragment of the agreement file as represented in RDFS. Local ontology O_1 uses a hierarchical land use code containing four properties: `broad`, `lu1`, `lu2`, and `lu3`, such that `lu1` is a subclass of `broad`, `lu2` is a subclass of `lu1`, and so on. In contrast, the local ontology O_2 only has one property, `luCode`, for land use coding. The mappings between the two ontologies are as follows: `broad`, `lu1`, `lu2`, and `lu3` are respectively mapped to `luCode`, as a broader class (i.e., superclass), an equivalent class, a narrower class (i.e., subclass), and another narrower class.

```
SELECT  a, b, c
FROM    {$x}xyCoordinates{a}, {$x}bounding{b}, {$x}jurisName{c},
        {$x}state{d}, {$x}luCode{e}
WHERE   d = "Wisconsin" and e = "115"
```

In the FROM clause, we use basic schema path expressions composed of the property name (e.g., bounding) and data variables (e.g., $x) or class variables (e.g., a). The properties xyCoordinates and bounding stand for the geographical coordinates and boundaries of the land parcel, respectively. The other properties were already discussed and shown in Figures 1 and 4. In what follows, we focus on a particular subset of RQL, namely *conjunctive RQL* (c-RQL), which is of the following form: $ans(\mathbf{x}) :- R_1(\mathbf{x}_1), ..., R_n(\mathbf{x}_n).$, where $\mathbf{x} \subseteq \mathbf{x}_1 \cup ... \cup \mathbf{x}_n$ are variables or constants, and $R_i(\mathbf{x}_i)$ ($i \in [1..n]$) is either a class predicate $C(x)$ or a property predicate $P(x, y)$. As usual, $ans(x)$ is the *head* of the query, denoted $head_Q$, and $R_1(\mathbf{x}_1), ..., R_n(\mathbf{x}_n)$ is the *body* of the query, denoted $body_Q$. For instance, the RQL query on multiple family land parcels can be expressed in c-RQL as follows:

$$\begin{aligned} ans(a,b,c) :- \ & xyCoordinates(x,a),\ bounding(x,b),\ jurisName(x,c), \\ & state(x, \texttt{"Wisconsin"}),\ luCode(x, \texttt{"115"}). \end{aligned}$$

Query processing across the whole system can be performed in two directions: global-to-local and local-to-local. We propose a query rewriting algorithm, QueryRewriting, which can be used in both cases. Query rewriting can be seen as a function $Q' = f(Q, M)$, where Q is the query to be rewritten, called *source query*, M is the set of ontology mappings, and Q' is the resulting query, called *target query*. The algorithm is shown in Figure 5.

In the global-to-local case, the source query Q is posed on the global ontology G, M is the set of mappings from G to every local ontology $O_1, ..., O_n$, and the target query Q' is the union of multiple subqueries over $O_1, ..., O_n$. In the local-to-local case, Q is a local query posed on a local ontology O_i ($i \in [1..n]$), M is the set of mappings from O_i to one or more local ontologies O_j ($j \in [1..n]$ and $j \neq i$), and Q' is the union of multiple subqueries over all O_j. In the latter case, M is, in fact, a set of compositions of the mappings from O_i to G with those from G to O_j.

The QueryRewriting algorithm consists of four main steps: 1) *source query expansion* using the source ontology constraints, 2) *schema-level mapping* where the expanded source query is rewritten into an intermediate target query using schema-level mappings, 3) *intermediate target query expansion* using the target ontology constraints, and 4) *instance-level mapping* where the expanded intermediate target query is rewritten using instance-level mappings to obtain the final target query. In what follows, we cover the overall query processing by describing the three key components of the four main steps listed above: query expansion, schema-level mapping, and instance-level mapping. Finally, we discuss some of our assumptions and prove the correctness of the query rewriting algorithm.

```
Algorithm QueryRewriting ($Q$, $M$)
Input: a conjunctive query $Q$ over ontology $O$; the mappings $M$ between
ontologies $O$ and $O'$.
Output: a union $\mathcal{Q}$ of conjunctive queries $Q'$ over $O'$.
1   $head_{Q'} = head_Q$;  $body_{Q'} = null$;
2   $Q^* =$ QueryExpand($Q$, $\Sigma$), where $\Sigma$ is the set of constraints over $O$;
3   Let $\phi$ be $body_{Q^*}$;
4   Let $M_1$ be the part of schema-level mappings in $M$;
5   For each $R(\mathbf{x})$ of $\phi$
6       For each $\psi \in M_1$
7           Let $R'(\mathbf{x}')$ be the result of applying $\psi$ on $R(\mathbf{x})$;
8           $body_{Q'} = R'(\mathbf{x}') \wedge body_{Q'}$;
9   $Q' =$ QueryExpand($Q'$, $\Sigma'$), where $\Sigma'$ is the set of constraints over $O'$;
10  Let $M_2$ be the part of instance-level mappings in $M$;
11  $\mathcal{Q} =$ ConstantMapping($Q'$, $M_2$);
12  Return $\mathcal{Q}$;
```

Fig. 5. The QueryRewriting algorithm.

Query expansion In the above description of the QueryRewriting algorithm, both the source query Q and the intermediate target query Q' are expanded using the ontology constraints, respectively in Lines 2 and 9. This query expansion process, as described by the QueryExpand function of Figure 6, uses the strategy of applying the ontology constraints to "chase" the query, similarly to the *chase* algorithm that is used in relational databases to compute dependency implications or optimize queries [1]. In relational databases, a database constraint can be represented as a *tgd* (*tuple generating dependency*) in the form $\forall\mathbf{x}\exists\mathbf{y}\ \varphi(\mathbf{x}) \rightarrow \psi(\mathbf{x}, \mathbf{y})$, where φ and ψ are conjunctions of atoms. In an ontology setting, we consider three kinds of constraints, namely, *subclass*, *subproperty*, and *typing* constraints, all of which can be represented as a *tgd*. Specifically, the *tgd* $\forall x\ C_1(x) \rightarrow C_2(x)$ corresponds to a subclass constraint $C_1 \subseteq C_2$; the *tgd* $\forall x \forall y\ P_1(x, y) \rightarrow P_2(x, y)$ corresponds to a subproperty constraint $P_1 \subseteq P_2$; and the *tgd* $\forall x \forall y\ P(x, y) \rightarrow A(x)$ (resp. $\forall x \forall y\ P(x, y) \rightarrow B(y)$) corresponds to a typing constraint that the instances of x (resp. y) are of type A (resp. B).

Similarly to the chase algorithm, QueryExpand is a non-deterministic process that terminates, provided that the dependencies are *acyclic* (we assume no constraints such as $A \subseteq B$, $B \subseteq C$, and $C \subseteq A$ in an ontology) and the applications of dependencies do not introduce new variables into the query (since all the three constraints: *subclass*, *subproperty*, and *typing* do not contain the existence quantifier). Under these conditions, given a conjunctive query Q and constraints Σ over an ontology O, it has also been proved that the algorithm QueryExpand has the resulting query $Q' =$ QueryExpand (Q, Σ) equivalent to Q, denoted $Q \equiv Q'$ [1]. This means that the answers to both queries are the same over all the ontology instances that satisfy the constraints. As an example, let us take the preceding query on multiple family land parcels, and denote it by Q.

```
Algorithm QueryExpand ($Q$, $\Sigma$)
Input: a conjunctive query $Q$ over ontology $O$; the constraints $\Sigma$ over $O$.
Output: The query $Q$ after the expansion.
1  Repeat
2      Let $\phi$ be $body_Q$;
3      Let $\psi : R_1(\mathbf{x}) \rightarrow R_2(\mathbf{x})$ be any dependency in $\Sigma$;
4      If there exists a homomorphism $h$ from $R_1(\mathbf{x})$ to $\phi$, but not from
       $R_1(\mathbf{x}) \wedge R_2(\mathbf{x})$ to $\phi$, then
5          Extend $h$ to a new homomorphism $h'$ from $R_1(\mathbf{x}) \wedge R_2(\mathbf{x})$ to $\phi$;
6          Add $h'(R_2(\mathbf{x}))$ into $body_Q$;
7      Else exit repeat;
8  End repeat
```

Fig. 6. The QueryExpand algorithm.

As specified on the global ontology G, all the properties (e.g., xyCoordinates) referred in Q belong to the class LandParcel, thus leading to the corresponding typing constraints. Such constraints can be represented by a *tgd* of the form $\forall x \forall y\ P(x, y) \rightarrow A(x)$ (e.g., $\forall x \forall y\ xyCoordinates(x, y) \rightarrow LandParcel(x)$). By applying them to Q, we obtain the following expansion of Q:

$$ans(a, b, c) :- \ xyCoordinates(x, a),\ bounding(x, b),\ jurisName(x, c),\ state(x, \texttt{"Wisconsin"}),\ luCode(x, \texttt{"115"}),\ LandParcel(x).$$

Furthermore, given that the LandParcel class is a subclass of Land in G, the corresponding *tgd* (e.g., $\forall x\ LandParcel(x) \rightarrow Land(x)$) of such constraint is still applicable to the above query. The final resulting expansion Q^* of Q is as follows:

$$ans(a, b, c) :- \ xyCoordinates(x, a),\ bounding(x, b),\ jurisName(x, c),\ state(x, \texttt{"Wisconsin"}),\ luCode(x, \texttt{"115"}),\ LandParcel(x),\ Land(x).$$

Schema-level mapping The key to query rewriting lies in Lines 4 to 7 of the QueryRewriting algorithm, which maps the expanded source query Q^* to a new query Q' over the target ontology, based on the set of schema-level mappings in M. Similarly to the ontology constraints used by QueryExpand, ontology mappings can be treated as constraints specified over the source and the target ontologies. Therefore, we express ontology mappings in a *tgd*. However, the use of these mappings is different from the use of ontology constraints for query expansion, as explained in what follows.

Consider two ontologies O_1 and O_2. Given the *tgd* $\psi : \forall \mathbf{x}\ R_1(\mathbf{x}) \rightarrow R_2(\mathbf{x})$, if ψ represents an ontology constraint $R_1 \subseteq R_2$, where R_1, $R_2 \in O_1$, and $R_1(\mathbf{x})$ is part of the query's body, then ψ is applicable to the query, and $R_2(\mathbf{x})$ should be added to the query. This query expansion, as described by the QueryExpand function, will not bring false positives to the query's answer, since the instances that satisfy ψ are in O_1. In comparison, if ψ is an ontology mapping $R_1 \subseteq R_2$, where $R_1 \in O_1$ and $R_2 \in O_2$, then this constraint implies a potential data transfer from O_1 to O_2. In this sense, $\psi : \forall \mathbf{x}\ R_1(\mathbf{x}) \rightarrow R_2(\mathbf{x})$ is not applicable to queries containing R_1 (like in the ontology constraint case), but is applicable to those containing R_2. This happens because a query retrieving instances of R_2 is also retrieving instances of R_1, given the semantics of ψ.

Therefore, the application of a dependency $\psi : R_2(\mathbf{x}) \rightarrow R_1(\mathbf{x})$ to a query Q, as Line 7 of QueryRewriting indicates, is performed by taking the converse ψ' of ψ (i.e., $R_1(\mathbf{x}) \rightarrow R_2(\mathbf{x})$), followed by the operations specified in Lines 4 and 5 of QueryExpand. The resulting $R'(\mathbf{x}')$ (in Line 8 of QueryRewriting) is then $h'(R_2(\mathbf{x}))$ as in Line 6 of QueryExpand. The following shows the result of mapping Q^* (the expanded source query) to a query Q' on the local ontology O_1 according to the mapping M as presented in Figure 4:

$$\begin{aligned} ans(a,b,c) \coloneq\ & xyCoordinates(x,a),\ boundingBox(x,b),\ jurisName(x,c), \\ & state(x, \texttt{"Wisconsin"}),\ lu1(x, \texttt{"115"}),\ LandParcel(x). \end{aligned}$$

If we compare Q' to the previous two queries (Q and Q^*) obtained in the query rewriting process, we notice that $Land(x)$ was first added into Q' by the query expansion step, and then it disappeared after the schema-level query mapping. In reality, the $LandParcel(x)$ in Q^* is different from $LandParcel(x)$ in Q': the former is against the global ontology G, whereas the latter is against the local ontology O_1, as shown in Figure 4. Therefore, the disappearance of $LandParcel(x)$ from Q' is due to the mapping from $LandParcel(x)$ and $Land(x)$ on G to $LandParcel(x)$ on O_1.

Instance-level mapping Both the QueryExpand function and the query mapping process are performed at the schema level. In comparison, the rewriting of the constants that are referred to in the query is based on the instance-level mappings between two ontologies, particularly the mappings between two land use taxonomies. We describe next the instance rewriting process of Figure 7.

In this case, we have $\mathbf{c} = \{\texttt{"Wisconsin"}, \texttt{"115"}\}$. From the mapping between G and O_1 shown in Figure 3, it follows that $\texttt{RT} \subseteq \texttt{115}$ and $\texttt{RM} \subseteq \texttt{115}$. Therefore, from Lines 3 to 9, we have that $A_1 = \{\texttt{"Wisconsin"}\}$ and $A_2 = \{\texttt{"RT"}, \texttt{"RM"}\}$. Now that we have two vectors of constants ($\mathbf{c}'$ in the algorithm): $\{\texttt{"Wisconsin"}, \texttt{"RT"}\}$ and $\{\texttt{"Wisconsin"}, \texttt{"RM"}\}$, we obtain the union of the following two queries (see Lines 10 to 13).

$$\begin{aligned} ans(a,b,c) \coloneq\ & xyCoordinates(x,a),\ boundingBox(x,b),\ jurisName(x,c), \\ & state(x, \texttt{"Wisconsin"}),\ lu1(x, \texttt{"RT"}),\ LandParcel(x). \\ ans(a,b,c) \coloneq\ & xyCoordinates(x,a),\ boundingBox(x,b),\ jurisName(x,c), \\ & state(x, \texttt{"Wisconsin"}),\ lu1(x, \texttt{"RM"}),\ LandParcel(x). \end{aligned}$$

Discussion We have assumed that the schema-level mapping M between two ontologies is a *full mapping*, that is, all relation atoms (including classes and properties) in the body of the query need to have been mapped to some atom in the other ontology, with the mapping type being $\supseteq$ or $\equiv$. Under this assumption,

```
Algorithm ConstantMapping ($Q$, $M$)
Input: a conjunctive query $Q$ over ontology $O'$ with constants $c_1, ..., c_n$ from
$O$; the instance level mappings $M$ between ontologies $O$ and $O'$.
Output: a union $\mathcal{Q}$ of conjunctive queries $Q'$ with constants from $O'$.
1    $\mathcal{Q} = \emptyset$;
2    $\mathbf{c} = (c_1, ..., c_n)$;
3    For each $c_i$, with $i \in [1..n]$
4        $A_i$={};
5        Let $C$ be the class standing for $c_i$;
6        For each $C \supseteq C'$ or $C = C'$ in $M$
7            $A_i = A_i \cup \{c'\}$, where $c'$ is the constant represented by $C'$;
8        If there is no $C \supseteq C'$ or $C = C'$ in $M$ then
9            $A_i = \{c\}$;
10   For each $\mathbf{c}' \in A_1 \times ... \times A_n$
11       $Q' = Q$;
12       Substitute $\mathbf{c}$ in $Q'$ with $\mathbf{c}'$;
13       $\mathcal{Q} = \mathcal{Q} \cup Q'$;
```

Fig. 7. The ConstantMapping algorithm.

we can prove the *soundness* of the QueryRewriting algorithm on its computation of a rewriting (i.e., target query) $\mathcal{Q}$ *contained* in the source query Q, denoted $\mathcal{Q} \subseteq Q$. A proof sketch follows.

Let Q^* be the expanded source query, Q' be the intermediate target query, Q'' be the expanded intermediate target query. Given that $Q \equiv Q^*$, $Q' \equiv Q''$, and $Q'' \equiv \mathcal{Q}$ [1], it suffices to prove that $Q' \subseteq Q^*$. Suppose that t is an instance in the answer to Q', i.e., $t \in Q'(\mathbf{O})$, where $\mathbf{O}$ is the local ontology instance. Then t makes every predicate $R(x)$ in $body_{Q'}$ true. According to Lines 5 to 8 of the QueryRewriting algorithm, every predicate $S(x)$ in $body_{Q^*}$ is also made true by t. This means that $t \in Q^*(\mathbf{G})$, where $\mathbf{G}$ is the global ontology instance, therefore $Q' \subseteq Q^*$. We note that we obtain a *contained* rewriting, instead of a *maximally contained* rewriting [23]. This is actually due to our preference for high precision rather than for high recall, which we discuss below.

There are two important steps involved in the local-to-local query rewriting: *query conversion* and *mapping composition*. The query conversion deals with the conversion of a query (e.g., in XPath) native to the local system to a query (in c-RQL) on the local ontology. However, c-RQL can only represent a particular class of XML queries that have the same expressive power as c-RQL. Therefore we only consider such XPath queries. The other step has to do with the transitivity of the mappings. That is, the composition of two equivalent mappings yields an equivalent mapping. The composition of two subclass mappings (or one subclass and one equivalent mappings) results in a subclass mapping. In the same way, the composition of two superclass mappings (or one superclass and one equivalent mappings) results in a superclass mapping. We do not consider any other mapping compositions.

The last issue we discuss relates to the trade-off between the precision and recall of the query processing. Currently, the the query rewriting algorithm only uses mappings that guarantee the correctness of the query. For instance, given a query $Q : \{x|A(x)\}$, our query rewriting algorithm only rewrites Q to $\{x|B(x)\}$ in two cases: $A \supseteq B$ or $A \equiv B$. This ensures that we will not return to the user instances that do not belong to A. But we may miss some instances of B that are also instances of A and should be included in the answer to Q, thus lowering recall. An alternative is to allow the approximate semantic relationship and to assign a score between [0..1] to every mapping based on the similarity of the mapped classes or properties. Thus, query rewriting can calculate an estimated precision of the target query. In practice, different scenarios impose different requirements on the mappings. For example, an eCommerce application involving purchase orders requires a very precise and complete translation of a query, whereas a search engine usually does not require an exact transformation [8].

6 User Interfaces

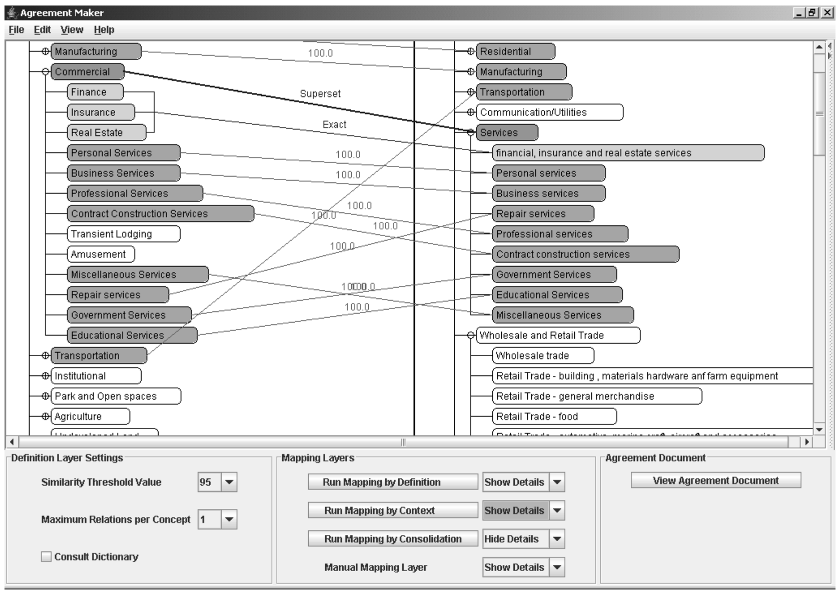
In this section, we briefly describe the two user interfaces that assist respectively in ontology alignment (and in particular instance-level mapping) and in query processing.

6.1 Visual Ontology Alignment

The AgreementMaker is a visual software tool that is used to create the mappings between the global ontology and each local ontology and to generate the agreement documents [9]. With this tool, users load two ontologies side-by-side and display each one as a tree of concepts as shown in Figure 8. The global ontology is displayed on the left hand side and the local (target) ontology is displayed on the right hand side. Concept (or class) names are displayed in rectangular nodes on the ontological trees.

The AgreementMaker implements a *hybrid* ontology matching process [28], by combining several matching methods to determine the schema-level and instance-level mappings between both ontologies.

The first two criteria are the *rule-based deduction* and the *user interaction*, as discussed in Section 4. User interventions are necessary when deductions are not applicable. Users map concepts manually based on their knowledge of the domain represented by the ontologies. The particular mappings that are established depend on the perceived semantic relationships among concepts. The deduction process automates the creation of new mappings based on existing mappings, provided that the pre-conditions for a particular deduction are satisfied. The degree of automation depends on the graph topologies and on the degree of similarity between the ontologies [10]. In the case where the topologies differ substantially and deduction cannot be used, the burden on users to perform mappings manually increases [25].



Similarity Threshold Value 95

Maximum Relations per Concept 1

Consult Dictionary

Run Mapping by Definition — Show Details

Run Mapping by Context — Show Details

Run Mapping by Consolidation — Hide Details

Manual Mapping Layer — Show Details

Fig. 8. The ontology alignment interface.

In addition to the previous criteria, the tool also provides *matching by definition*, which matches the name and the description of the concepts. The procedure consults a dictionary (e.g., WordNet[1]) and returns a semantic relationship (e.g., *hypernym*, *hyponym*, or *synonym*) between both concepts and a *similarity* score ranging from 0 to 100. The shortcoming of this matching criterion is that two concepts can have the same name and the same description, but they could be semantically mismatched because they occur in different contexts. To address this problem our tool considers the paths leading to the concepts [10].

The tool supports a fourth matching step, *matching by consolidation*, whereby users provide a ranking of the matching criteria. In this way, wherever there are conflicting results for the matchings, the highest ranked criterion will take precedence.

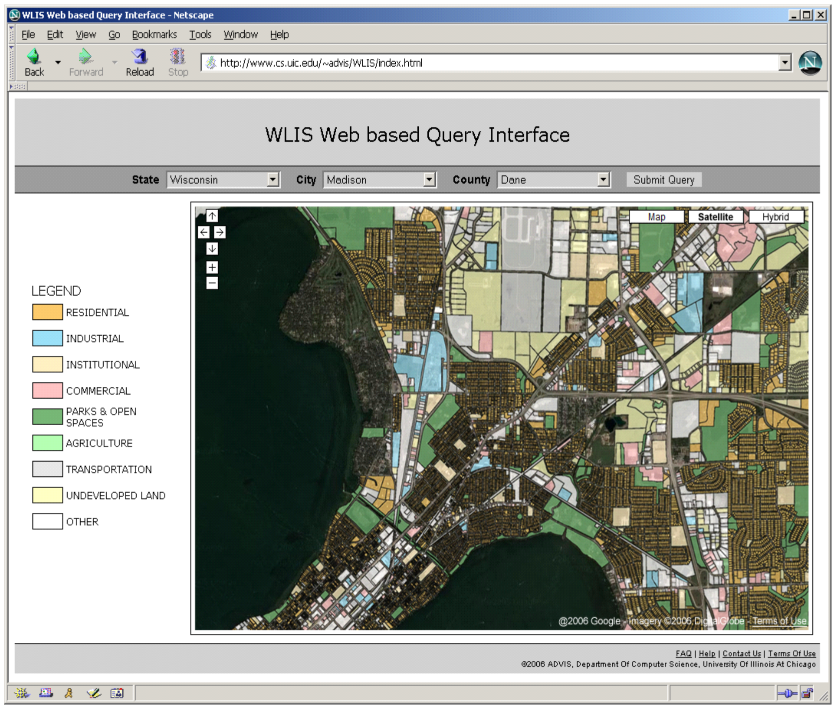
6.2 Web-based Query Interface

The prototype of a web-based visual query interface has been implemented for browsing different types of land usage in a geospatial area that can span several local data sources. This interface serves as a proof-of-concept for the interoperability of heterogeneous geospatial data based on the query rewriting algorithm discussed in Section 5.

Figure 9 shows the land use map of the city of Madison where parcels are highlighted with colors indicating their associated land use categories. In partic-

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

ular, the user initiates a query by selecting a State, County, and City, which is then represented by a c-RQL query on the global ontology. The global query is rewritten into subqueries over the local data sources. The results are integrated and visualized in the form of a land use map, which is superimposed on satellite images obtained from Google Maps.[2]


WLIS Web based Query Interface - Netscape
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://www.cs.uic.edu/~advis/WLIS/index.html
WLIS Web based Query Interface
Submit Query
Map Satellite Hybrid
LEGEND
RESIDENTIAL
INDUSTRIAL
INSTITUTIONAL
COMMERCIAL
PARKS & OPEN SPACES
AGRICULTURE
TRANSPORTATION
UNDEVELOPED LAND
OTHER
@2006 Google - Imagery ©2006 DigitalGlobe - Terms of Use
FAQ | Help | Contact Us | Terms Of Use
©2006 ADVIS, Department Of Computer Science, University Of Illinois At Chicago

State: Wisconsin City: Madison County: Dane

Fig. 9. A Web-based user query interface.

7 Related Work

We discuss related work in the three main topics considered in this paper: ontology alignment, query processing, and geospatial data integration. While in each category there is work related to our own, there is no work that spans the three main topics that form the basis of our approach.

Ontology alignment The problem of ontology alignment has received much attention recently [30]. A large effort has been devoted to techniques that enable

[2] http://maps.google.com

the automatic (or semi-automatic) alignment of concepts across ontologies [15]. Existing alignment approaches make use of one or more ontology alignment (or matching) techniques belonging to the following three categories:

Element-level At the element level, matching can use various similarity measures based, for example, on names of elements or their textual descriptions. A normalized numerical value is calculated for each of the matching candidates, and the best one is selected [4, 6, 26].

Structure-level The structure-level information that can be used by the matching process include the *graph* or *taxonomy* underlying the schema or ontology. Graphs are used as contextual information to map pairs of elements and the taxonomy can provide the matching process with more semantics [24, 25]. An example of semantic-level matching determines the similarity of two concepts based on the similarities of their ancestors [29]. In our approach, we consider the semantic similarity of the concepts' children, instead.

Instance-level Instance-level matching uses the actual contents (or instances) of the schema or ontology elements [12, 21].

Although we have two types of ontologies to align, namely the *schema-like* ontologies and the *taxonomy-like* ontologies, in this paper we have mainly concentrated on the alignment of the latter type. Taxonomy-like ontologies use only the subconcept (or subclass) relationships between two entities, therefore they lend themselves well to the use of structure-level methods. However, element-level and instance-level approaches can be used in conjunction with our structure-level methods. In fact, our prototype makes use of element-level alignment as described in Section 6.

Related systems to ours include Clio [20], COMA++ [3], and Falcon-AO [22]. The first two place a strong emphasis on the user interface, like we do, while the last two share with our approach their support for structure-level automatic matching methods.

Query processing When mappings are defined as (relational) views, query processing is often referred in the literature as view-based query answering or rewriting [19]. However, few view-based query processing algorithms address the issue of query rewriting over ontologies involving the specific kinds of issues involved, which we must take into account [31].

Schema and ontology-based query processing techniques have been proposed both for centralized [2, 27, 34] and for peer-to-peer architectures [5, 13, 16]. While most of these approaches focus on XML or relational query languages to perform the query rewriting, we use RQL because of our choice of metadata and data representation.

Our ontology-based query rewriting algorithm is similar to the *computeWTA* algorithm proposed by Calvanese *et al.* for query reformulation [5] as both assume consistent ontology mappings. However, we allow for the transformation of the values that are contained in the query based on the instance-level ontology mappings. In this way, we can address semantic heterogeneity, which occurs in the land use codes.

Another related approach considers constraint-based query processing in the Clio system [34]. It focuses on schema mapping and data transformation between nested (XML) schemas and relational databases by taking advantage of the schema semantics to generate consistent translations from source to target and by considering the constraints and structure of the target schema.

Geospatial data integration Information integration methodology from the database community has been applied to spatial information systems. For example, the MIX framework offers a meditation approach for integrating heterogeneous sources containing spatial data types (e.g., vector graphics, maps) and associated data (e.g., text, tables, figures, images) [18], which supports a wide range of spatial applications. The system architecture comprises three layers: a foundation layer consisting of databases and wrappers, a mediation layer supporting query and result exchange among the wrapped sources, and an application/user interface layer. In the foundation layer, the data model is exported from the sources in the form of an XML DTD. In the case of spatial information, for example, the wrapper constructs the DTD by using the associated catalog information. The wrappers support scripts that execute complex queries as a combination of several primitive queries. As compared to our approach, semantic relationships are supported, for example in the form of spatial predicates such as *within(region1,region2)*, but there is not an overall “semantic graph” that would support, for example, the alignment of spatial attributes.

VirGIS is a more recent approach for mediation of geographical information systems [14]. It differs from MIX in that it adopts newer standards such as GML (Geography Markup Language) for data modeling and WFS (Web Feature Servers) to perform communications (e.g., queries) with clients. It supports mappings between attributes or classes but no semantic overall framework is presented.

8 Conclusions

In this paper, we focused on data integration and interoperability across distributed geospatial data sources. To illustrate the impact of our approach we showed practical examples that are derived from land use applications.

We propose an ontology-based approach to achieve the integration and interoperability of the distributed geospatial data sources by solving both schematic and semantic heterogeneities. Two different kinds of mappings are established between the global ontology (which describes the domain) and each local ontology (which describes each data source): *schema mappings* between the schema of both ontologies and *instance mappings* between the (land use) taxonomies of both ontologies.

We have discussed two modes of query processing in our system, *global-to-local* and *local-to-local* (or *peer-to-peer*). Query rewriting in both modes uses the previously established mappings. We propose a c-RQL (conjunctive RQL) query rewriting algorithm, such that the resulting target query is *contained* in the source query, thus providing sound answers to the source query.

Future work will focus on:

– Ontology alignment, and in particular the deduction-based method. Currently, we make some assumptions on the topology of the ontologies. Without such assumptions, we may need to consider the combination of our bottom-up deduction process with top-down reasoning on mappings (e.g., [29]).
– Query rewriting, so as to take into account "approximate" mappings. In this case, precision and recall of query answering will depend on the similarity of the underlying mappings, thus making the ability to determine mapping similarities a critical task.

9 Acknowledgements

We would like to thank Nancy Wiegand and Steve Ventura, from the Land Information & Computer Graphics Facility at the University of Wisconsin-Madison, and the members of WLIS for discussions on land use and other scenarios related to geospatial data integration. We would also like to thank Sujan Bathala, Nalin Makar, Afsheen Rajendran, and William Sunna for their help with the design and implementation of the user interfaces.

Bibliography

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[2] B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Querying XML Sources Using an Ontology-Based Mediator. In *Confederated International Conferences DOA, CoopIS and ODBASE*, volume 2519 of *Lecture Notes in Computer Science*, pages 429–448. Springer, 2002.

[3] D. Aumueller, H. H. Do, S. Massmann, and E. Rahm. Schema and Ontology Matching with COMA++. In *ACM SIGMOD International Conference on Management of Data*, pages 906–908, 2005.

[4] S. Bergamaschi, S. Castano, and M. Vincini. Semantic Integration of Semistructured and Structured Data Sources. *SIGMOD Record*, 28(1):54–59, 1999.

[5] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. What to Ask to a Peer: Ontology-based Query Reformulation. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 469–478, 2004.

[6] S. Castano, V. D. Antonellis, and S. D. C. di Vimercati. Global Viewing of Heterogeneous Data Sources. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):277–297, 2001.

[7] V. Christophides, G. Karvounarakis, I. Koffina, G. Kokkinidis, A. Magkanaraki, D. Plexousakis, G. Serfiotis, and V. Tannen. The ICS-FORTH SWIM: A Powerful Semantic Web Integration Middleware. In *International Workshop on Semantic Web and Databases (SWDB)*, pages 381–393, 2003.

[8] V. Cross. Uncertainty in the Automation of Ontology Matching. In *International Symposium on Uncertainty Modeling and Analysis (ISUMA)*, pages 135–140, 2003.

[9] I. F. Cruz, W. Sunna, and A. Chaudhry. Semi-Automatic Ontology Alignment for Geospatial Data Integration. In *International Conference on Geographic Information Science (GIScience)*, volume 3234 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2004.

[10] I. F. Cruz, W. G. Sunna, and K. Ayloo. Concept Level Matching of Geospatial Ontologies. In *GIS Planet International Conference and Exhibition on Geographic Information*, 2005.

[11] I. F. Cruz, H. Xiao, and F. Hsu. An Ontology-based Framework for Semantic Interoperability between XML Sources. In *International Database Applications and Engineering Symposium (IDEAS)*, pages 217–226, July 2004.

[12] A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy. Learning to Map between Ontologies on the Semantic Web. In *International World Wide Web Conference (WWW)*, pages 662–673, 2002.

[13] M. Ehrig, C. Tempich, J. Broekstra, F. van Harmelen, M. Sabou, R. Siebes, S. Staab, and H. Stuckenschmidt. SWAP - Ontology-based Knowledge Man-

agement with Peer-to-Peer Technology. In *German Workshop on Ontology-based Knowledge Management (WOW)*, 2003.

[14] M. Essid, F.-M. Colonna, O. Boucelma, and A. Bétari. Querying Mediated Geographic Data Sources. In *International Conference on Extending Database Technology (EDBT)*, volume 3896 of *Lecture Notes in Computer Science*, pages 1176–1181. Springer, 2006.

[15] J. Euzenat, A. Isaac, C. Meilicke, P. Shvaiko, H. Stuckenschmidt, O. Šváb, V. Svátek, W. R. van Hage, and M. Yatskevich. First Results of the Ontology Evaluation Initiative 2007. In *Second ISWC International Workshop on Ontology Matching*. CEUR-WS, 2007.

[16] E. Franconi, G. M. Kuper, A. Lopatenko, and I. Zaihrayeu. A Distributed Algorithm for Robust Data Sharing and Updates in P2P Database Networks. In *Current Trends in Database Technology - EDBT 2004 Workshops*, Lecture Notes in Computer Science, pages 446–455. Springer, 2004.

[17] T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[18] A. Gupta, R. Marciano, I. Zaslavsky, and C. K. Baru. Integrating GIS and Imagery Through XML-Based Information Mediation. In *International Workshop on Integrated Spatial Databases (ISD), Selected Papers*, volume 1737 of *Lecture Notes in Computer Science*, pages 211–234. Springer, 1999.

[19] A. Y. Halevy. Answering Queries Using Views: A Survey. *VLDB Journal*, 10(4):270–294, 2001.

[20] M. A. Hernández, R. J. Miller, and L. M. Haas. Clio: A Semi-Automatic Tool For Schema Mapping (demo). In *ACM SIGMOD International Conference on Management of Data*, page 607, 2001.

[21] R. Ichise, H. Takeda, and S. Honiden. Rule Induction for Concept Hierarchy Alignment. In *IJCAI Workshop on Ontologies and Information Sharing*, 2001.

[22] N. Jian, W. Hu, G. Cheng, and Y. Qu. Falcon-AO: Aligning Ontologies with Falcon. In *K-CAP 2005 Workshop on Integrating Ontologies*. CEUR Workshop Proceedings 156, 2005.

[23] M. Lenzerini. Data Integration: A Theoretical Perspective. In *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.

[24] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In *IEEE International Conference on Data Engineering (ICDE)*, pages 117–128, 2002.

[25] N. F. Noy and M. A. Musen. Anchor-PROMPT: Using Non-local Context for Semantic Matching. In *IJCAI Workshop on Ontologies and Information Sharing*, 2001.

[26] L. Palopoli, D. Saccà, and D. Ursino. An Automatic Techniques for Detecting Type Conflicts in Database Schemes. In *International Conference on Information and Knowledge Management (CIKM)*, pages 306–313, 1998.

[27] M. Peim, E. Franconi, N. W. Paton, and C. A. Goble. Query Processing with Description Logic Ontologies Over Object-Wrapped Databases. In *In-*

ternational Conference on Scientific and Statistical Database Management (SSDBM), pages 27–36, 2002.

[28] E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal*, 10(4):334–350, 2001.

[29] M. A. Rodríguez and M. J. Egenhofer. Determining Semantic Similarity among Entity Classes from Different Ontologies. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):442–456, 2003.

[30] P. Shvaiko and J. Euzenat. A Survey of Schema-Based Matching Approaches. In *Journal on Data Semantics IV*, volume 3730 of *Lecture Notes in Computer Science*, pages 146–171. Springer, 2005.

[31] H. Stuckenschmidt. Query Processing on the Semantic Web. *Künstliche Intelligenz (KI)*, 17(3):22–26, 2003.

[32] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-Based Integration of Information - A Survey of Existing Approaches. In *IJCAI Workshop on Ontologies and Information Sharing*, 2001.

[33] N. Wiegand, D. Patterson, N. Zhou, S. Ventura, and I. F. Cruz. Querying Heterogeneous Land Use Data: Problems and Potential. In *National Conference for Digital Government Research (dg.o)*, pages 115–121, 2002.

[34] C. Yu and L. Popa. Constraint-Based XML Query Rewriting For Data Integration. In *ACM SIGMOD International Conference on Management of Data*, pages 371–382, 2004.