Query Translation for Ontology-extended Data Sources

Jie Bao[1], **Doina Caragea**[2], **Vasant Honavar**[1]
[1]Artificial Intelligence Research Laboratory,
Department of Computer Science,
Iowa State University, Ames, IA 50011-1040, USA
[2]Department of Computing and Information Sciences
Kansas State University, Manhattan, KS 66506, USA
[1]{baojie, honavar}@cs.iastate.edu, [2]dcaragea@ksu.edu

Abstract

The problem of translating a query specified in a user *data content* ontology into queries that can be answered by the individual data sources is an important challenge in data integration in e-science applications. We develop the notions of *semantics-preserving query translation* and *maximally informative query translation* in such a setting. We describe an algorithm for maximally informative query translation and its implementation in INDUS, a suite of open source software tools for integrated access to semantically heterogeneous data sources. We summarize experimental results that demonstrate the scalability of the proposed approach with very large ontologies and mappings between ontologies.

Introduction

New discoveries in biological, physical, and social sciences are increasingly being driven by our ability to discover, share, integrate and analyze disparate types of data. This has led to a compelling vision for cyber-infrastructure for collaborative e-science. Inevitably, autonomous data sources will differ with respect to their structure, organization, query capabilities, and more importantly data semantics. Data sources that are created for use in one *context* often find use in other contexts. For many applications, there is also no single universal choice of data semantics that meets the needs of all data providers and users in all contexts. Therefore, differences in data semantics among data sources, users, or for that matter, even the same users in different contexts are fairly common.

Effective use of multiple data sources by a user in a given context requires context and user-specific reconciliation of differences in data semantics among them. Hence, ontologies that make explicit, the usually implicit data modeling assumptions (ontological commitments) regarding semantics of the data sources are beginning to play increasingly important roles in the reconciliation of semantic differences across data sources (Hull 1997; Wache *et al.* 2001; Levy 2000).

It is useful to distinguish between two distinct types of data semantics: the semantics of terms used in specifying the *structure* (e.g., names of attributes used to describe the data) of the data source, i.e., *data schema semantics*; and the semantics associated with contents (e.g., values of attributes used to describe the objects) of the data source i.e., *data content semantics*.

Much of the work in data integration has focused on bridging the semantic gaps between different data source *schema* definitions (See survey papers (Hull 1997; Ziegler & Dittrich 2004a; Wache *et al.* 2001)). However, barring a few exceptions (Wache & Stuckenschmidt 2001; Goh *et al.* 1999), the problem of bridging the semantic gaps with respect to *data content* semantics has received relatively little attention.

Consequently, there is an urgent need for approaches that:

- Facilitate a separation of concerns between data schema semantics and data content semantics;
- Allow each user to impose his or her own point of view (ontology) on the content of data sources of interest and formulate queries using terms in that ontology;
- Can automatically transform user queries into queries that are understood by the respective data sources;
- Can map the query results into the form expected and understood by the users;
- Can scale up to settings with very large ontologies and mappings.

Hence, in this paper, we focus on the problem of semantics-preserving translation of queries expressed using terms from a user ontology into queries that can be understood and answered by data sources. Our major contributions include:

- Separation of data schema semantics and data content semantics. Such a separation of concerns makes it possible to more effectively reuse mappings designed for bridging gaps in data content semantics and for bridging differences in data source schemas across a broader range of data integration scenarios. For example, we can use the same mappings between data content ontologies in settings where the data source schema can differ greatly (e.g., XML schema versus RDF schema versus relational database schema). It also allows us to leverage appropriate (and in some cases, relatively mature) techniques for each type of data source (relational versus XML or RDF) to answer aggregate queries against large data sets, in a manner that accommodates differences between data

sources in terms of data content semantics (e.g, due to different levels of granularity in data description).

- Investigation of the query translation from a user's *point of view*. The proposed approach circumvents the need for a global view of the data, and offers a principled approach to user and context-specific reconciliation of differences in data content semantics.
- Study of the semantics-preserving query translation process across different data content ontologies based on ontology-extended tuple relational calculus (OE-TRC). Specifically, we establish criteria for *sound*, *complete* and *exact* query translation in the presence of differences in data content semantics. We illustrate the proposed approach in the case of hierarchical ontologies. A novel feature of our approach is the ability to take into consideration, the semantics of not only constant terms but also ontological assertions about those terms in the query formulae.
- Implementation of the proposed query translation algorithm within INDUS (Caragea *et al.* 2005), an open source suite of tools for data integration. Of particular interest are optimizations that enhance the scalability of the system, and experimental results that offer evidence of such scalability in settings with very large ontologies and mappings.

Motivating Example

We introduce the query translation problem through a simple, intuitive example. Consider an academic department that collects information about its students, classes and instructors, as shown in Figure 1 (a). The data source consists of 3 entity tables and 2 relations: entity tables *Students*, *Classes* and *Instructors*; *Students* are related to *Classes* through the *RegisteredFor* relation and *Classes* are related to *Instructors* through the *OfferedBy* relation.

In this example, the (implicit) *data schema semantics* corresponds to the semantics of the terms used in the data source schema definition. For example, both *Students* and *Instructors* are *People*, therefore a query posed against the data sources for names of *People* should return the names in both *Students* and *Instructors* tables.

The *data content semantics* corresponds to the semantics of the terms used to specify the values of attributes used to describe the data instances. For example, the attribute value of *StudentStatus* in the *Student* table can be used to infer that, if a student status is "2nd year", then that student must also have status "Undergrad". In general, attribute values can be organized in an Attribute Value Hierarchy (see Figure 1 (b) for an example).

Such implicit data semantics can be made explicit by means of *ontologies* associated with the data sources. Suppose the data providers specify the intended semantics of the data, from the provider's point of view, by associating the data source with an ontology that captures the relevant semantic commitments. In a setting where both the data providers and data users are autonomous, and the *scope* of the relevant ontologies and their semantics are inevitably context and user specific, there is no single context and user-independent interpretation of data.

For example, consider a statistician named Bob who is interested in the question: Do *Ph.D.* students spend less time in classes than $Masters$ students? Suppose the analysis software that he uses has a different classification model for student status (See Fig. 2 (a)). Recall that the data source has no conception of what it means for the $Student.Status$ attribute to take the value "*Masters*". Furthermore, Jane, a second statistician, might subscribe to another point of view (and hence ontology), that is different from Bob's ontology (See Fig. 2 (b)) and also from that of the data source describing the department of interest. How can we support users like Bob and Jane with their needs to query a data source from their own points of view, given that their queries cannot be understood by the relevant data sources? We need to translate a query expressed in terms of a *user ontology* into a query that is expressed in terms of the *data source ontology*.

In what follows, we develop an approach to solving this problem thereby yielding sound methods for flexibly querying semantically rich relational data sources.

Ontology-Extended Data Sources

Inspired by ontology-extended relational algebra (Bonatti, Deng, & Subrahmanian 2003), we have introduced ontology-extended data sources (OEDS), which make explicit the ontologies associated with data sources, thus facilitating specification of semantic correspondences between data sources (Caragea, Pathak, & Honavar 2004). The separation of concerns between data source schema ontologies and data content ontologies makes it possible to use external knowledge with data, thereby expanding the universe of queries that can be meaningfully answered from a set of autonomous data sources.

Specifying an Ontology-extended data source

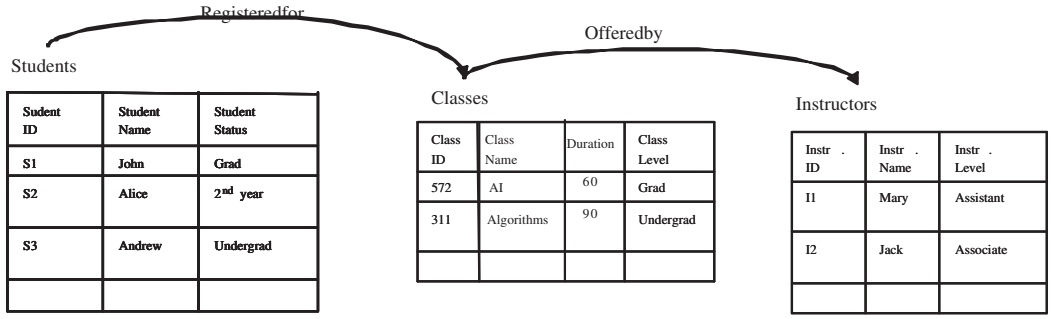
An OEDS is a tuple (S, D, O), where S is the *schema* of the data source, D is the *data set* contained in the data source, and O is the set of *ontologies* associated with the schema and the data set.

A *Schema* captures the information about the *structure* of the corresponding data source. Here, we restrict ourselves to structured data sources with associated relational schemas.

A *Data Set* is an instantiation of the data source schema. In relational databases, a data set consists of tables containing data tuples that conform to the relational database schema (see Figure 1 (a) for some examples).

An *Ontology* represents commitments regarding *semantics* of the vocabulary used in the data source schema and in the data. In other words, an ontology is a knowledge base that associates additional semantics with a database. Thus, in our example, we might find it useful to have an ontology that states that:

- (a) $Students$ are $People$, where the concept $People$ is more general than the concept $Student$;
- (b) The expression $Classes.Duration$ refers to time expressed in minutes;

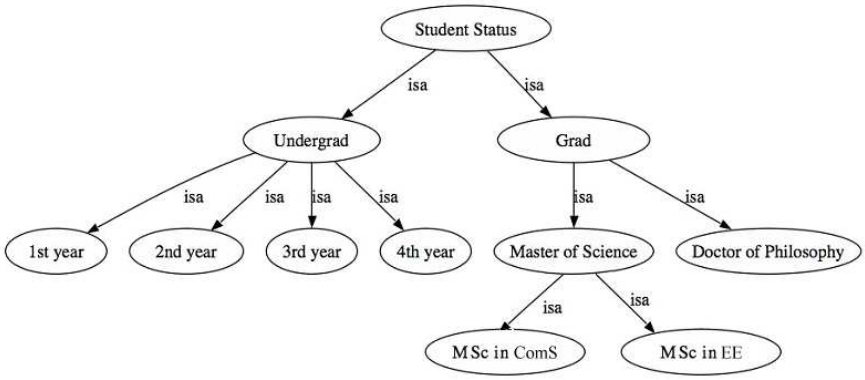


Sudent ID	Student Name	Student Status
S1	John	Grad
S2	Alice	2nd year
S3	Andrew	Undergrad

Class ID	Class Name	Duration	Class Level
572	AI	60	Grad
311	Algorithms	90	Undergrad

Instr . ID	Instr . Name	Instr . Level
I1	Mary	Assistant
I2	Jack	Associate

(a) The Data Schema and The Example Data Set



(b) The Data Content Ontology for Attribute *Student Status*

Figure 1: Ontology-Extended Data Source Example

- (c) Student status "$2ndYear$" implies that the student is an "$Undergrad$".

As noted above, it is useful to separate ontology O associated with a data source into a *data schema ontology* O_S (e.g., item (a) above) and *data content ontologies* O_C (e.g., items (b) and (c) above). In this paper, we will focus our discussion on data content ontologies. The related work section discusses some work on data schema ontologies.

Formally, we have the definition:

Definition 1 (Ontology-Extended Data Source) *An ontology-extended data source is specified by a tuple* (S, D, O) *where*

- *S is the schema of the DS, which is a first order language that corresponds to the relational language defined with a finite set of constants d_S, denoted domains Δ, no function symbols and a finite set $\mathbf{R}_S$ of predicate symbols (corresponding to relational tables); for every $R \in \mathbf{R}_S$, we call every argument of R an attribute of R; for every attribute a, its domain $domain(a)$ is a subset of Δ.*
- *D is the data set (i.e., an extension) corresponding to the DS, i.e., a first order interpretation $\mathcal{I}_D$ of S on Δ.*
- *O is the data content ontology, which is a tuple (L_O, D_O), where L_O is a first order language defined by a set of constants d_O, denoted domains Δ, and a finite set of predicate symbols $\mathbf{R}_O$, disjoint from $\mathbf{R}_S$; D_O is a first order interpretation $\mathcal{I}_O$ of L_O on Δ. O is called the data source ontology of DS if $d_O = d_S$.*

The definition of relational schema follows the model-theoretical description of Relational Model by (Reiter 1982). Note that we may also associate a schema ontology to a data source, i.e., a first order language with a predicate set $\mathbf{R}'$, such that $\mathbf{R}_S \subseteq \mathbf{R}'$ and $\mathbf{R}'$ is disjoint from $\mathbf{R}_O$. Typically, a schema ontology is aimed at extending the data source with intensional knowledge (e.g., TBox in description logics) about schema symbols (e.g., relational table names), whereas a data content ontology is aimed at extending the data source with extensional knowledge (e.g., ABox in description logics). Such a separation is useful from a modeling standpoint because processing extensional and intensional knowledge efficiently typically requires different mechanisms. For example, relational databases provide extremely efficient techniques for dealing with extensional knowledge (or instances).

In what follows, we assume that there are no semantic differences on the schema ontologies between data users and the data source, and focus on how to deal with semantic differences in data content ontologies.

Example: the OEDS description of the data source introduced in our example contains:

- The schema S with predicates $\mathbf{R}_S = \{ Students,$

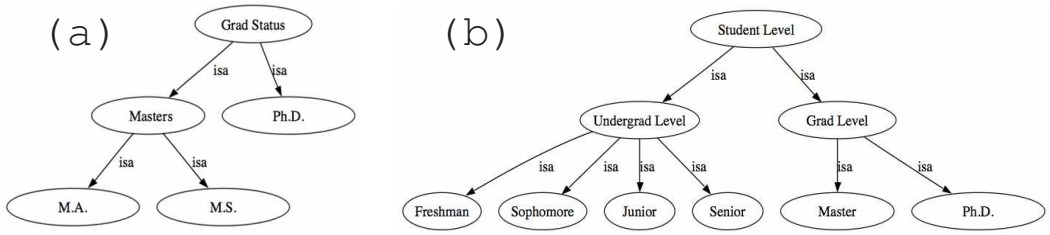


Figure 2: User Data Content Ontologies (a) Bob's Ontology (b) Janes's Ontology

$Classes$, $Instructors$, $RegisteredFor$ and $OfferedBy\}$; for the predicate $Students$, its attributes are $Student.ID$, $Student.Name$ and $Student.Status$; the set of constants is the set of all strings and integers that are legal in the database[1]; the domain Δ corresponds to the data contained in the database.

- The data set D is the instantiation of S, e.g.,

$$Student(s1, John, Grad)$$

$$Classes(572, AI, 60, Grad)$$

- The data content ontology $O = (L_O, D_O)$, where L_O has a binary transitive predicate isa, D_O contains an assertion set:

$$isa(MasterOfScience, Grad)$$

$$isa(DoctorofPhilosophy, Grad)...$$

Ontology-extended Query

Once the data content ontologies and the data source schema are specified, we can construct the usual relational queries with respect to such a data source. In what follows, we will use an extended form of the tuple relational calculus (TRC) to represent queries.

Definition 2 (Ontology-Extended Query) *A* tuple *is an ordered multiset of attributes defined in the database schema, in the form of* $(a_1, a_2, ..., a_n)$*, where each* a_i *is an attribute name. An* atomic formula *is in one of the following forms:*

- $t.a$ *op* c*, where* t *is a tuple variable,* a *is an attribute name,* c *is a constant from* a*'s domain, op is an operator defined on* $domain(a)$ *;*
- $t_1.a_1$ *op* $t_2.a_2$*, where* t_1, t_2 *are tuple variables,* a_1, a_2 *are attribute names, op is a common operator defined on* $domain(a_1)$ *and* $domain(a_2)$ *;*
- $R(t)$*, where* t *is a tuple variable,* $R \in \boldsymbol{R}_S \cup \boldsymbol{R}_O$*.*

A TRC formula *is constructed from atomic formulae using boolean connectors* $\wedge$ *(and),* $\vee$ *(or),* $\neg$ *(not), the existential quantifier (*$\exists$*) and the universal quantifier (*$\forall$*).*

A query $q(t)$ *is a TRC formula where* t *is the tuple of free variables. A query is said* ontology-extended *if it contains predicates from* $\boldsymbol{R}_O$*. The answer set of a query* $q(t)$ *against an OEDS* (S, D, O) *is the set of tuples of constants in* D *that satisfies* q *and it is enoted as* $\{q(t)\}$*.*

[1]Length of strings and range of integers are limited in realistic databases, hence they form a finite set.

For example, the set of "all students with the status of Master of Science" can be formulated as

$$q(t) = Students(t) \wedge isa(t.Status, MasterOfScience)$$

Query Translation for OEDS

Query Translation and Ontology Mapping

Query translation involves transforming a query expressed in one ontology into a query expressed in another ontology using a specified set of mappings that assert semantic correspondences between the two ontologies, while preserving the semantics of the original query.

We first define the notion of mapping between two data content ontologies.

Definition 3 (Ontology Mapping) *Let* $O_1 = (L_1, D_1)$ *and* $O_2 = (L_2, D_2)$ *be two data content ontologies, where* L_i *(with the predicate set* $\boldsymbol{R}_i$*) is the language of* O_i*, and* D_i *is an interpretation of* L_i *over a domain of* Δ*. A mapping between* O_1 *and* O_2 *contains a set of predicates* $\boldsymbol{R}_M$ *which is disjoint from* $\boldsymbol{R}_1$ *and* $\boldsymbol{R}_2$*, and a set of rules in the form of*

$$\forall \boldsymbol{x}, p_1(\boldsymbol{x}, \boldsymbol{c}_1) \wedge m(\boldsymbol{c}_1, \boldsymbol{c}_2) \rightarrow p_2(\boldsymbol{x}, \boldsymbol{c}_2)$$

$$\forall \boldsymbol{x}, p_2(\boldsymbol{x}, \boldsymbol{c}_1) \wedge m(\boldsymbol{c}_1, \boldsymbol{c}_2) \rightarrow p_1(\boldsymbol{x}, \boldsymbol{c}_2)$$

$$m(\boldsymbol{c}_1, \boldsymbol{c}_2)$$

where $\boldsymbol{x}$ *are tuples of free variables and* $\boldsymbol{c}_1, \boldsymbol{c}_2$ *are tuples of constants,* p_i *is a formula in* L_i*,* m *is a predicate in* $\boldsymbol{R}_M$*. The rules in the last form are called ground rules.*

For example, suppose we have two ontologies O_1, O_2 that model hierarchies, with $\mathbf{R}_1$=$\{isa_1\}$, $\mathbf{R}_2$=$\{isa_2\}$, where isa_i is a transitive predicate. The mapping M between O_1 to O_2 contains predicates $into$ and $onto$, along with the rules:

$$\forall x, isa_1(x, c_1) \wedge into(c_1, c_2) \rightarrow isa_2(x, c_2) \quad (1)$$

$$\forall x, isa_1(c_1, x) \wedge onto(c_1, c_2) \rightarrow isa_2(c_2, x) \quad (2)$$

$$\forall x, isa_2(c_2, x) \wedge into(c_1, c_2) \rightarrow isa_1(c_1, x) \quad (3)$$

$$\forall x, isa_2(x, c_2) \wedge onto(c_1, c_2) \rightarrow isa_1(x, c_1) \quad (4)$$

Definition 4 (Sound, Complete and Exact Query Translation) *Given a schema* S *and a data set* D*, two data content ontologies* O_1 *and* O_2*, an ontology mapping* M *between* O_1 *and* O_2*, Let* Q_i *be the set of all possible queries posed against* (S, D, O_i) $(i = 1, 2)$*. A query translation procedure* γ *is a mapping from* Q_1 *to* Q_2*.* γ *is*

- sound *if* $\forall q, q', (q, q') \in \gamma$*, we have* $\{q', M, O_1, O_2\} \models q$*.*

- complete *if* $\forall q, q'(q, q') \in \gamma$, *we have* $\{q, M, O_1, O_2\} \models q'$.
- exact *if* γ *is both sound and complete.*

where $\models$ is logical entailment.

A complete query translation procedure ensures that the desired result set is a subset of the returned result set. A sound query translation procedure ensures that the returned result set is always a subset of the desired result set. In an ideal setting, it is desirable for the query translation to be both sound and complete. However, since different ontologies usually represent different local points of view of the domain of interest, *exact* translations across different distributed data interpretations may not always be feasible. For example, in the motivating example, suppose that Bob wants to find out all students of the "MA" program:

$$Students(t) \wedge isa(t.Status, MA) \tag{5}$$

However, the data source does not contain the constant "MA" but only "$Grad$"; the available mapping contains the rules (1)-(4) and a ground rule $into(MA, Grad)$. Hence, there is no exact translation but only a complete translation for (5).

$$Students(t) \wedge isa(t.Status, Grad) \tag{6}$$

The only possible sound translation for (5) is

$$Students(t) \wedge \mathsf{false} \tag{7}$$

which will always return an empty set.

A naive way to ensure completeness of the query translation would be to return all of the data from a data source for any query. A naive way to ensure the soundness of the query translation would be to return an empty result set for any query. Neither of these approaches would be of much use in practice. Hence, we introduce the notion of a most informative translation:

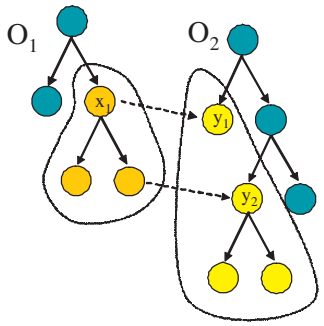
Definition 5 (Most Informative Translation) *A sound query translation procedure is said to be the most informative sound query translation procedure if,* $\forall q \forall M$*, given the task of translating the query* q *using the mapping* M*, it returns a sound query* q' *of* q *and for any other sound translation* q'' *of* q*, we have* $\{q'', M, O_1, O_2\} \models q'$.

A complete query translation procedure is said to be the most informative complete query translation procedure if $\forall q \forall M$*, given the task of translating the query* q *using the mapping* M*, it returns a complete query* q' *of* q *and for any other complete translation* q'' *of* q*, we have* $\{q', M, O_1, O_2\} \models q''$.

For example, in Figure 3, different sound translation procedures could translate $isa_1(t, x_1)$ into $isa_2(t, y_1)$ or $isa_2(t, y_2)$, but a most informative sound translation would yield $isa_2(t, y_1) \vee isa_2(t, y_2)$.

Query Translation for Hierarchical Ontologies

In what follows we introduce concrete query translation strategies for hierarchical ontologies. Precisely, a hierarchy is a set of isa assertions over a finite set of constants S, where isa is a transitive and reflexive binary predicate.



Solid lines represent isa_i $(i = 1, 2)$ relations ($x \rightarrow y$ indicates $isa_i(y, x)$), dotted lines represent $onto$ ground mapping rules ($x \rightarrow y$ indicates $onto(x, y)$).

Figure 3: Multiple Possible Sound Translations

Ontology mappings between hierarchies contains $into$ and $onto$ ground rules. To simplify the notation, we will use partial-ordering notation "$\leq$" and "$\geq$" as follows:

- $isa(x, y)$ iff $x \leq y$, for x, y in one ontology
- $into(x, y)$ iff $x \leq y$, for x, y in different ontologies
- $onto(x, y)$ iff $y \leq x$, for x, y in different ontologies
- $x \leq y$ iff $y \geq x$
- $x \equiv y$ iff $x \leq y$ and $y \leq x$
- $super(x) = \{y | x \leq y, y \neq x\}$, for x, y in one ontology
- $sub(x) = \{y | y \leq x, y \neq x\}$, for x, y in one ontology

By definition, the result of a most informative sound translation of a formula q must be a logical consequence of the result of any other sound translation of q. Thus, we have a most informative sound translation procedure, that works as follows. Consider the translation of $A \leq O_1 : x$ to an ontology O_2:

1. Find the term set S (called the greatest lower bound of $O_1 : x$ in O_2, denoted as $GLB(x)$) from O_2 such that for any term $y \in S$, (1) there exists a term x', $x' = x$ or $x' \in sub(x)$, such that there is a ground mapping rule $x' \geq y$; (2) there is no another term y' in S such that $y' \geq y$.
2. Output the disjunction: $\vee_{y \in S} A \leq y$.

It can be easily shown that this procedure is guaranteed to be a most informative sound translation procedure. Similarly, we have the strategy for a most informative complete translation. Suppose the task is to translate $A \leq O_1 : x$:

1. Find the term set S (called the least upper bound of $O_1 : x$ in O_2, denoted as $LUB(x)$) from O_2 such that for any term $y \in S$, (1) there exists a term x', $x' = x$ or $x' \in super(x)$, such that there is a ground mapping rule $x' \leq y$; (2) there is no another term y' in S such that $y' \leq y$.
2. Output the conjunction: $\wedge_{y \in S} A \leq y$.

The translation rules for atomic queries using constants from hierarchial ontologies is summarized in the Table 1.

Exact translation requires that for every term in the source ontology, there is an equivalent term in the target ontology.

Table 1: Translation Rules for Atomic Query Formulas Using Hierarchical Ontologies

Translation	$A \geq x$	$A \leq x$
Sound	$\vee_{y \in GLB(x)} A \geq y$	$\vee_{y \in LUB(x)} A \leq y$
Complete	$\wedge_{y \in LUB(x)} A \geq y$	$\wedge_{y \in GLB(x)} A \leq y$
Exact	$A \geq y \mid_{x \equiv O_2:y}$	$A \leq y \mid_{x \equiv O_2:y}$

Note: Translation from ontology O_1 to O_2, x is a term in O_1.

The task of translating a complex query formula can be reduced to the subtasks of translating its atomic formulas. We denote a translation of a query p using a sound (complete) translation procedure T_s as $T_s(p)$ ($T_c(p)$). For an OE-TRC formula p, a sound translation of p is obtained from the following rules:

- $T_s(p) = T_s(p_1) \vee T_s(p_2)$, if $p = p_1 \vee p_2$
- $T_s(p) = T_s(p_1) \wedge T_s(p_2)$, if $p = p_1 \wedge p_2$
- $T_s(p) = \neg T_c(p_1)$, if $p = \neg p_1$

Similarly, a complete translation can be obtained from the following rules:

- $T_c(p) = T_c(p_1) \vee T_c(p_2)$, if $p = p_1 \vee p_2$
- $T_c(p) = T_c(p_1) \wedge T_c(p_2)$, if $p = p_1 \wedge p_2$
- $T_c(p) = \neg T_s(p_1)$, if $p = \neg p_1$

Recursively applying such translation rules, a complex query formula can be transformed in a way that guarantees the soundness (or completeness) of the overall translation procedure.

Semantics Preserving Query Translation

Inappropriate mappings may result in incorrect translation. For example, in Figure 4, there are several ground mapping rules $O_1 : x \leqslant O_2 : y_1$, $O_1 : x \leqslant O_2 : y_2$ and $O_1 : x \geqslant O_2 : y_3$. A query expressed in O_1, $A \leq O_1 : x_1$, is to be translated into a most informative complete translation $(A \leq O_2 : y_1) \wedge (A \leq O_2 : y_2)$ according to our translation rules. However, this will always return an empty data set since y_1 and y_2 in ontology O_2 have no local common children. On the other hand, $A \leq O_1 : x_1$ has a sound translation $A \leq O_2 : y_3$. Thus, a sound translation procedure may return a data set that is a superset of the data set returned by a complete translation procedure, which contradicts the very definitions of soundness and completeness.

The cause for such an inconsistency can be traced to the semantic differences between the mappings and the ontologies in question. While there is no common child of y_1 and y_2 from the point of view of O_2 alone, the given mapping constraints (denoted as M) actually introduce global common children for y_1 and y_2 in O_1 and O_2 (e.g. $O_1 : x_1$ and $O_2 : y_3$). Thus, the set $\{M, O_1, O_2\}$ infers new knowledge ($y_3 \leq y_1$) which is not inferable from O_2 alone. In general, to ensure that the semantic point of view adopted during the mapping (i.e. $\{M, O_1, O_2\}$) and the semantic point of view adopted during query execution (O_2) are consistent, we should require the mapping to be semantics-preserving:

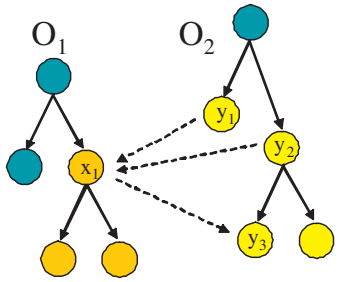


Figure 4: Impropriate mappings may result in incorrect translation

Definition 6 *The set of interoperation constraints M from an ontology $O_1(S_2, \leq)$ to another ontology $O_2(S_2, \leq)$ is semantics-preserving iff*

$\forall y_1, y_2 \in S_2, \{M, O_1, O_2\} \models y_1 \leq y_2 \leftrightarrow O_2 \models y_1 \leq y_2$

The notion of semantics-preserving ontology mapping is closely related to the notion of Conservative Extension of ontologies (Ghilardi, Lutz, & Wolter 2006). An ontology is a conservative extension of one of its subsets if it does not entail new knowledge over the signature of the subset. Hence, semantics-preserving mappings ensure the union of the source ontologies and the mappings taken together, to be a conservative extension of the target ontology, to avoid interfering with the semantics of the target ontology. The mappings must be semantics preserving in order for us to be able to ensure that the query translation does not lead to unexpected results, as illustrated by the above example.

Implementation and Experimental Results

We have implemented the proposed query translation algorithm for hierarchial ontologies inside the INDUS (INtelligent Data Understanding System) environment[2] and have exploited several optimization techniques to handle large ontologies and ontology mappings:

- Database storage for large ontologies and ontology mappings: We store both ontologies and mappings as compressed objects in a database. This greatly reduces the space requirement for storage as well as the communication traffic between clients and the server (usually by an order of magnitude).
- Combining query translation and query rewriting (to SQL queries) using the transitive closure of ontologies: The implementation of the query engine precomputes the transitive closure of partial-ordering in a meta ontology that is constructed from the source ontologies and ontology mappings. This allows us to combine query translation and query rewriting into a single process.
- Avoiding expensive query formulas using server-side caching: Expensive query formulas, such as SQL "IN" clauses with long list of operands, can be replaced by

[2]http://www.cs.iastate.edu/ dcaragea/INDUS/documentation.htm

more efficient ones using server-side caching, e.g, using temporary tables on the data source server. Such a strategy can improve query execution efficiency by three orders of magnitude when working with large ontologies.

- Client-side caching. Once an ontology or ontology mapping is loaded by an INDUS client, it will be cached on the client size. Thus, subsequent queries from the same client will not be forced to reload them.

Our experiments have shown that the implemented algorithm works well on very large ontologies and ontology mappings. The data set we used in the experiment is an enzyme database (4,931 records) annotated with the Enzyme Classification (EC) hierarchy[3] (4,564 terms). The user ontology we used is the SCOP (Structural Classification of Proteins) hierarchy[4] (86,766 terms). The ontology mapping from SCOP to EC is provided by Richard George et. al.[5] with 15,765 mapping rules.

We randomly generated queries in the SCOP ontology that specify enzyme data associated with particular structural families. For each query in the experiment, we randomly selected a non-leaf term t from SCOP and construct a query with selection condition "protein_family $<= t$". The query is translated into another query that uses EC terms.

To estimate the cost of both local computation (e.g. query translation) and communication overhead (e.g. reading remote ontologies or retrieving result data set), we separated the data source server and the INDUS query client on different machines. The server runs on a Intel Pentium 4 2.4GHz /2GB RAM machine with PostgreSQL 8.0 as the RDBMS. The client runs on a Intel Celeron 631 MHz / 384MB RAM machine with Windows XP OS. The two machines are physically separated in different LANs.

Table 2: Scalability Results

	Branch Size	Translation Time(s)	Execution. Time(s)	Result Size	Query Complexity
Avg	29.9	0.064	0.16	2.35	2.76
Max	86765	16.463	48.720	2,837	2,840

Table 2 shows the experimental results on the average and maximal cost in the query translation and execution process for 8,216 randomly selected queries. We measured:

- Branch Size, which indicates the size of the fragment of the ontology involved w.r.t the given query. For a query of the form $A <= t$, it is the size of $sub(t)$. Branch size gives an upper bound of the space complexity of the query before the query translation.
- Translation Time, the time used in translating the query from the user ontology to the data source ontology.
- Execution Time, the time used in executing the translated query and retrieving the data. It measures the time used for sending the translated query to the data source RDBMS via network (in this experiment over JDBC) and the time used to transfer the data fetched from the data source to the INDUS cache database.
- Result Size, the size of the retrieved data set satisfied by the translated query.
- Query Complexity, the space complexity of the translated query (before any optimization). It is defined as the size of the expression tree of the translated query.

Our results suggest that the query translation and query execution strategies adopted in this study are tractable in terms of both time and space requirements when working with very large ontologies (over 100k terms in total in this experiment) and ontology mappings.

Related Work

There is extensive study addressing the problems of structural heterogeneity or schema semantic heterogeneity (see the survey paper (Shvaiko & Euzenat 2005)). Query translation (also called transformation, rewriting) (see the survey paper (Calvanese, Lembo, & Lenzerini 2001)) under such setting has also been studied. However, few of these studies address the problem we investigated in this paper, i.e., translation of a query against data sources with context-specific *data content ontologies*.

OBSERVER (Mena *et al.* 2000) processes query translation not only using term substitution but also considering operators in query formulas. It allows subsumption relations in ontology mappings and partial query translation, and provides a metrics to measure the loss of information. However, ontologies are only associated with data schema, thus this approach does not address data content semantic heterogeneity problem as we addressed in this paper.

In **SIRUP** (Ziegler & Dittrich 2004b), the ontology is added on the top of the data, therefore it is similar to our ontology-*extended* data source approach. It also allows user to have user-specific interpretations on the same set of data sources. However, ontologies in SIRUP, built with *IConcepts*, are also data schema ontologies, not data content ontologies. Users are required to build *Semantic Prospectives* from a selected subset of the data source IConcepts, therefore are not allowed to adopt autonomous, context-specific local ontologies. Thus, it is also not possible in this approach to do query translation form a user ontology to the data source ontology.

The problem of data content semantic heterogeneity is addressed in part by the systems BUSTER and COIN:

BUSTER (Wache & Stuckenschmidt 2001) represents both schema and content semantics explicitly in Description Logics. The approach they adopted uses a logic reasoner to do the query translation. However, this approach does not address the problem of relational query translation and query execution when query conditions include ontological operations. Thus, query translation is equivalent to term substitution. BUSTER assumes that all data source ontologies share a basic terminology, which seems unrealistic in a setting with autonomous data sources, which calls for context and user-specific semantic correspondence between data sources.

[3] http://www.chem.qmul.ac.uk/iubmb/enzyme/

[4] http://scop.mrc-lmb.cam.ac.uk/scop/

[5] http://www.enzome.com

COIN (Goh *et al.* 1999) also addresses data content semantic heterogeneity and uses ontologies as data types. In this approach, semantics of each data source is *contextualized*. Different contexts are linked by conversion functions. However, the the focus was on only numerical scale units, and not on handling more complex structures such as hierarchies. COIN also only addresses the query translation as term substitution (equivalent to our approach when there are equivalence relations between terms in ontologies), without considering ontological assertions about constants in the ontology and the query.

Our approach also differs significantly from attempts at reconciling data-level heterogeneity by solving the data duplication or entity/object matching problem (Doan, Noy, & Halevy 2004). Instead of *matching* data instances, our focus is on exploiting the semantic relation between data instances which need not necessarily represent the same object in the domain of interest.

Conclusions

We have studied the query translation process for data sources extended with context-specific data content ontologies. We have shown how to exploit ontologies and mappings for flexibly querying (from a user perspective) semantic-rich (relational) data sources in a setting where each data source can be viewed as a set of relational tables. We have proposed a query translation strategy that works for hierarchial ontologies. We have analyzed the conditions under which the soundness and completeness of such a procedure can be guaranteed.

There are several promising directions for further work:

Query Translation with More expressive ontologies: This paper focused on query translation for hierarchical ontologies. However, the proposed framework is not necessarily only limited to hierarchies. Study of query translation using more expressive ontology languages, such as a proper subset of description logics, would be interesting to explore.

Complexity Analysis: Study of the theoretical complexity and decidability result of query translation when the ontologies in question are proper subsets of description logics is also of interest.

Statistical query answering: The proposed query translation process can also serve as the basis for knowledge acquisition from semantically heterogeneous distributed data sources using statistical queries (such as SUM, COUNT or AVERAGE).

Query Translation for non-relational OEDS: The discussion in this paper is focused on OEDS with relational schema. It would be interesting to explore extensions to XML-based data sources and RDF-based data sources.

Acknowledgement: This research was supported in part by the US National Science Foundation grants IIS-0219699 and IIS-0639230.

References

Bonatti, P. A.; Deng, Y.; and Subrahmanian, V. S. 2003. An ontology-extended relational algebra. In *IRI*, 192–199.

Calvanese, D.; Lembo, D.; and Lenzerini, M. 2001. Survey on methods for query rewriting and query answering using views. D2I Deliverarble D1.R5, University of Rome.

Caragea, D.; Pathak, J.; Bao, J.; Silvescu, A.; Andorf, C. M.; Dobbs, D.; and Honavar, V. 2005. Information integration and knowledge acquisition from semantically heterogeneous biological data sources. In *Proceedings of the 2nd International Workshop on Data Integration in Life Sciences (DILS'05), San Diego, CA*, 175–190.

Caragea, D.; Pathak, J.; and Honavar, V. 2004. Learning classifiers from semantically heterogeneous data. In *CoopIS/DOA/ODBASE (2)*, 963–980.

Doan, A.; Noy, N. F.; and Halevy, A. Y. 2004. Introduction to the special issue on semantic integration. *SIGMOD Rec.* 33(4):11–13.

Ghilardi, S.; Lutz, C.; and Wolter, F. 2006. Did i damage my ontology? a case for conservative extensions in description logics. In *KR*, 187–197.

Goh, C. H.; Bressan, S.; Madnick, S. E.; and Siegel, M. 1999. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Trans. Inf. Syst.* 17(3):270–293.

Hull, R. 1997. Managing semantic heterogeneity in databases: A theoretical perspective. In *PODS*, 51–61.

Levy, A. Y. 2000. Logic-based techniques in data integration. In *Logic-based artificial intelligence*. Kluwer Academic Publishers. 575–595.

Mena, E.; Illarramendi, A.; Kashyap, V.; and Sheth, A. P. 2000. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases* 8(2):223–271.

Reiter, R. 1982. Towards a logical reconstruction of relational database theory. In *On Conceptual Modelling (Intervale)*, 191–233.

Shvaiko, P., and Euzenat, J. 2005. A survey of schema-based matching approaches. 146–171.

Wache, H., and Stuckenschmidt, H. 2001. Practical context transformation for information system interoperability. In *CONTEXT*, 367–380.

Wache, H.; Vgele, T.; Visser, U.; Stuckenschmidt, H.; Schuster, G.; Neumann, H.; and Hbner, S. 2001. Ontology-based integration of information - a survey of existing approaches. In *Proceedings of the IJCAI-01 Workshop: Ontologies and Information Sharing*.

Ziegler, P., and Dittrich, K. R. 2004a. Three decades of data integration - all problems solved? In *IFIP Congress Topical Sessions*, 3–12.

Ziegler, P., and Dittrich, K. R. 2004b. User-specific semantic integration of heterogeneous data: The sirup approach. In *ICSNW*, 44–64.