Efficient Selection and Integration of Data Sources for Answering Semantic Web Queries

Abir Qasem
Lehigh University
19 Memorial Drive West
Bethlehem, PA 18015, USA
abir.qasem@gmail.com

Dimitre A. Dimitrov
Tech-X Corporation
5621 Arapahoe Avenue, Suite A
Boulder, CO 80303, USA
dad@txcorp.com

Jeff Heflin
Lehigh University
19 Memorial Drive West
Bethlehem, PA 18015, USA
heflin@cse.lehigh.edu

Abstract

In this work we adapt an efficient information integration algorithm to identify the minimal set of potentially relevant Semantic Web data sources for a given query. The vast majority of these sources are files written in RDF or OWL format, and must be processed in their entirety. Our adaptation includes enhancing the algorithm with taxonomic reasoning, defining and using a mapping language for the purpose of aligning heterogeneous Semantic Web ontologies, and introducing a concept of source relevance to reduce the number of sources that we need to consider for a given query. After the source selection process, we load the selected sources into a Semantic Web reasoner to get a sound and complete answer to the query. We have conducted an experiment using synthetic ontologies and data sources which demonstrates that our system performs well over a wide range of queries. A typical response time for a substantial work load of 50 domain ontologies, 80 map ontologies and 500 data sources is less than 2 seconds. Furthermore, our system returned correct answers to 200 randomly generated queries in several workload configurations. We have also compared our adaptation with a basic implementation of the original information integration algorithm that does not do any taxonomic reasoning. In the most complex configuration with 50 domain ontologies, 100 map ontologies and 1000 data sources our system returns complete answers to all the queries whereas the basic implementation returns complete answers to only 28% of the queries.

1 Introduction

The Semantic Web provides an infrastructure that has the potential to transform the Web to a true global knowledge medium. Ontologies, expressed in a standard logic language with formal semantics, can be used in concert with web data in order to develop powerful query systems. The research community and the industry have made significant progress toward realizing this vision. For example, the Web Ontology Language (OWL) is now an international standard [13]. However, the Semantic Web is a decentralized medium where different parties can and will, in general, adopt different ontologies. When many ontologies and data sources are created independently of one another it is quite possible that many of them will refer to the same or similar concepts. Therefore, while some of the data sources may contain data described directly in terms of a given query ontology, others may not. Furthermore, regardless of the advances in reasoning techniques and best efforts in clever coding we will always have to deal with data sets that are just too big for a given Semantic Web knowledge base system. In this work we consider an approach for identifying the minimal set of potentially relevant Semantic Web data sources for a given query while making sure that we do not miss relevant sources that commit to ontologies different from the one used in the query.

In order to align heterogeneous ontologies, we use the notion of map ontologies. In our solution, these are like any other OWL ontology except they consist solely of axioms that relate concepts from one ontology to concepts of another ontology. We use the term domain ontologies for all other ontologies. The choice of OWL to articulate the alignments make these maps shareable via the Web, where any one can create alignments and publish them in OWL for others to use. Such maps may be created manually or by using state-of-the-art ontology alignment tools. We note that aligning ontologies is a difficult and relevant problem, but this is not the focus of this paper.

We hypothesize that a user driven alignment framework will enable integration to be an emergent property of the Web. Furthermore, existing OWL tools can be used to process these maps. We note that we will not have alignments between all pairs of ontologies, but it should be possible to compose an alignment from existing alignments.

Our approach uses a concept that we refer to as "source relevance" in conjunction with an adapted information integration algorithm to select the data sources that are sufficient to provide the answers to a given query. Intuitively, a data source is potentially relevant to a query if it has some information on some predicate of the query. In our approach a data source can make assertions about its contents by means of what we call REL statements. We then use these selected sources and an OWL reasoner to obtain answers to distributed queries on the Semantic Web data. Note, that this selective loading of only relevant data sources will provide a more optimal solution because in the Semantic Web, data sources significantly outnumber ontologies. This is reflected in Swoogle's present cache which has 10,000 ontologies but an impressive 2.5 million documents [4]. Currently our implemented system supports a restricted subset of SPARQL queries, simple OWL ontologies and data sources that commit to them.

Once we have the maps between the ontologies established, we need to use them to translate a query in terms of the data sources that are available. Database researchers have developed information integration algorithms that address similar problems. In our work we adapt one such algorithm by Halevy *et al.* known as the Peer Data Management System (PDMS) reformulation algorithm [8]. PDMS uses both Global-As-View (GAV) and Local-As-View (LAV) maps, which are the two most well known information integration formalisms.

Our query language is based on the conjunctive query language for Description Logic (DL) that has been proposed by Horrocks *et al.* [9]. This query language overcomes the inadequacy of DL languages in forming extensional queries. Furthermore, it corresponds to the most common SPARQL queries. Note, however, that our implementation is restricted by the query language supported by KAON2 [12] which we use as our reasoner. KAON2 can only answer the so called DL-safe conjunctive queries [6].

Our design choices were influenced by the following observations about the current state of the Semantic Web.

- *Observation 1:* The majority of Semantic Web data is in flat RDF/OWL files as opposed to databases. Therefore, any system that uses such data must load the whole source as opposed to issuing queries to the sources as is done in PDMS. Furthermore, these files are served via HTTP. We use the term "atomic data sources" to refer to this type of data sources.

- *Observation 2:* Many data sources may commit to the same ontology and a majority of the data commits to simple ontologies like FOAF, Dublin core, etc. As data significantly outnumbers ontologies, our focus is on instance retrieval as opposed to queries about the ontological structure. We also focused on reasoning over simple ontologies given their dominance in the Semantic Web.

- *Observation 3:* Although, ontologies are relatively static, the data may change frequently. This means a centralized knowledgebase approach would be ineffective, since the quality of answers depends on the frequency of crawling and loading data.

This work is an extension of our initial work presented in Dimitrov *et al.* [3]. The previous work was based on a two-tier ontology architecture so it had no support for map composition. Our enhancements now enable us to support a multi-tier ontology architecture and we can compose maps of arbitrary length. Specifically we make the following three technical contributions in this paper.

1. We formally define the source selection problem.

2. We present an algorithm that uses OWL maps and REL statements to identify relevant data sources for a given query.

3. We demonstrate that by loading the sources selected by our algorithm into a DL reasoner, we achieve an efficient and effective solution for answering distributed queries on the Semantic Web.

The rest of the paper is organized as follows: In Section 2 we formally introduce the source selection problem, in Section 3, we describe our mapping language that is compatible with the mapping formalisms used by the PDMS and introduce a mechanism for describing source relevance. In Section 4, we describe the details of the Ontology Based Information Integrator (OBII), including a source selection algorithm. In Section 5, we describe some experiments that we have conducted to evaluate our system. In Section 6, we compare some related work with our approach and in Section 7 we conclude and discuss future work.

2 The Source Selection Problem

In this section we formally define the source selection problem (i.e. how to select the potentially relevant data sources for a given query) in the context of the Semantic Web.

The Semantic Web can be viewed as a set of classes (and individual that belong to them) and properties; axioms that relate these classes, properties and individuals; and documents which contain these various Semantic Web entities. In the discussion that follows we use $\mathcal{C}$ to refer to the set of all classes, $\mathcal{P}$ to refer to the set of all properties, $\mathcal{D}$ to refer to the set of all individuals, and $\mathcal{U}$ to refer to the set of document identifiers (URLs in the case of OWL) in the Semantic Web.

We say an ontology is some subset of $\mathcal{C} \cup \mathcal{P}$ (and possibly a set of axioms that relate them). Note: According to the official OWL description [13] an ontology can also have individuals in addition to the classes and properties that describe those individuals. However, for convenience of analysis we decided to separate ontologies (i.e. the class/property definitions and axioms that relate them) and data sources (assertions of class membership or property values). We say each data source is a set of assertions of the form a:c $\in \mathcal{C}$ or $\langle a, b\rangle : p \in \mathcal{P}$ where a and b are names that denote individuals. Basically, any OWL document that contains a description of a class or a property is an ontology; otherwise, the document is a data source. We note that this separation does not violate any of OWL's formal semantics.

In order to align heterogeneous ontologies, we introduce the notion of map ontologies. These are like any other OWL ontology except they consist solely of axioms that relate concepts from one ontology to concepts of another ontology. We use the term domain ontologies for all other ontologies.

We are now ready to define our problem space. We introduce two functions. First, *o* (hereinafter referred to as ontology function) which maps $\mathcal{U}$ to a set of ontologies. Second, *s* (hereinafter referred to as source function) which maps $\mathcal{U}$ to a set of data sources.

Definition 1 (Semantic Web Space) *A Semantic Web Space SWS is a tuple* $\langle \mathcal{U}, o, s\rangle$.

Definition 2 (Satisfaction) *An interpretation* $\mathcal{I}$ *satisfies a Semantic Web Space* $\langle \mathcal{U}, o, s\rangle$*, iff for each u* $\in \mathcal{U}$*,* $\mathcal{I}$ *satisfies o(u) and s(u).*

We define satisfaction of o(u) and s(u) per the official OWL semantics document [13].

A knowledge base entails (written $\models$) a set of logical sentences α iff every interpretation that satisfies the knowledge base also satisfies α. We now define the notion of entailment of a SWS.

Definition 3 (Semantic Web Space Entailment) *Given a set of description logic sentences* α*, SWS* $\models \alpha$ *iff every interpretation that satisfies SWS also satisfies* α

Definition 4 (Conjunctive Query Form) *A conjunctive query has the form* $H\left(\overline{X}\right)$:-$B_1\left(\overline{X}_1\right), \ldots, B_n\left(\overline{X}_n\right)$ *where* $\overline{X}$ *is a vector of variables and/or individuals and each* B_i *is a unary or a binary atom representing a concept or role term respectively.*

Our query language is based on the conjunctive query language for DLs that has been proposed by Horrocks *et al.* [9]. This query language overcomes the inadequacy of description logic languages in forming extensional queries.

Furthermore, it corresponds to the most common SPARQL queries. We refer to the left hand side of :- as the head of the query and the right hand side as the body of the query. The variables that appear in the head must appear also in the body and are universally quantified. Such variables are called distinguished variables and describe the form of a query's answers. All other variables in the query are called non-distinguished variables and are existentially quantified. For a given query Q and substitution θ, we use Qθ as a shorthand for $\mathrm{B}_1\theta \wedge \mathrm{B}_2\theta \ldots \wedge \mathrm{B}_n\theta$.

Definition 5 (Answer Set) *Given a Semantic Web Space SWS, an answer set* $\mathcal{A}$ *to a query Q is the set of all substitutions* θ *for all distinguished variables in Q such that: SWS* $\models Q\theta$.

All variables in a query should be mapped to individuals explicitly introduced in the data sources. This definition follows the definition presented by Motik and Sattler [11].

We now introduce the concept of source relevance in our framework by means of a "REL" statement. A REL statement allows us to make assertions about the type of information that a source contains so that we can ignore sources that we are certain will not contribute to $\mathcal{A}$ for a given query. In order to give a description of the REL statements we first define two relations that establish relationship between classes and properties defined in ontologies, individuals asserted in data sources and document identifiers needed to access these sources.

1. SrcInst: $\mathcal{U} \times \mathcal{S} \times \mathcal{C} \rightarrow 2^D$ that maps the set of document identifiers, the set of source functions $\mathcal{S}$ and the set of classes $\mathcal{C}$ to the power set of individuals $\mathcal{D}$. Essentially, SrcInst gives us the set of individuals of a given class according to a given data source, i.e. for each u $\in \mathcal{U}$, s $\in \mathcal{S}$ and c $\in \mathcal{C}$, the interpretation of SrcInst(u, s, c) = $\{(a^I) \mid a : c \in s(u)\}$.

2. SrcInstP: $\mathcal{U} \times \mathcal{S} \times \mathcal{P} \rightarrow 2^{D \times D}$ that maps the set of source functions $\mathcal{S}$ and the set of properties $\mathcal{P}$ to the power set of individual pairs $\mathcal{D} \times \mathcal{D}$. Essentially, SrcInstP gives us the sets of individual pairs that are related by a given property according to a given data source, i.e. for each u $\in \mathcal{U}$, s $\in \mathcal{S}$ and p $\in \mathcal{P}$, the interpretation of SrcInstP (u, s, p) = $\{\langle a^I, b^I\rangle \mid \langle a, b\rangle : p \in s(u)\}$.

The REL statement has the following two forms: a) REL(u, C_j, CE) and b) REL(u, P_j, P') where U_i is a document identifier, C_j is an atomic class, CE is a class expression, and P_j, P' are properties.

Definition 6 (Source Conformance) *Given a source function s we say source u conforms to a set of REL statements* $\mathcal{R}$ *iff*

1. *for each $r \in \mathcal{R}$ s.t. r = REL(u, C_j, CE), SrcInst(u, s, C_j) is $\neq \perp$ and SrcInst(u, s, C_j) $\sqsubseteq$ CE* ***or*** *r = REL(u, P_j, P′), SrcInstP(u, s, P_j) is $\neq \perp$ and SrcInstP(u, s, P_j) $\sqsubseteq$ P′*

2. *for all a:C_j $\in$ s(u) $\exists$ r $\in \mathcal{R}$ s.t. r = REL (u, C_j, CE) and s(u) $\models$ {a} $\sqsubseteq$ CE*

3. *for all $\langle a, b\rangle$:P_j $\in$ s(u) $\exists$ r $\in \mathcal{R}$ s.t. r = REL (u, P_j, P′) and s(u) $\models$ $\langle a, b\rangle$:P′*

Definition 7 (Source Function Conformance) *We say a source function s conforms to a set of REL statements R if $\forall$ u $\in \mathcal{U}$, s(u) conforms to R*

Definition 8 (Potential Relevance) *Given a conjunctive query Q, a set of REL statements $\mathcal{R}$, a set of document identifiers $\mathcal{U}$, and an ontology function o, a source u is potentially relevant to Q iff $\exists$ a source function s that conforms to R where $\langle U, o, s\rangle \models Q\theta$ and $\langle U, o, s-u\rangle \not\models Q\theta$.*

Note, s-u denotes a function f(x) as follows:

$$f(x) = \begin{cases} s(x) & if\ x \neq u \\ \oslash & otherwise \end{cases}$$

Definition 9 (Source Selection Problem) *Given a Semantic Web Space SWS, a query Q and a set of REL statements $\mathcal{R}$, the source selection problem is to identify all potentially relevant sources.*

3 Mapping Language

In this section, we introduce OWL for Information Integration (OWLII). It is a subset of OWL, the design of which is influenced by the PDMS algorithm. We use OWLII to describe maps and data sources in the Semantic Web using GAV and LAV rules. Since OWL DL is decidable, its subset OWLII is also decidable.

We follow a process similar to Grosof *et al.* [5] to define the subset of DL for OWLII. In the discussion below the subscript *a* is used to refer to a DL language whose classes can be mapped to antecedents of a FOL implication and the subscript *c* is used to refer to a DL language whose classes can be mapped to consequents. Similarly, we use the subscript *ac* to refer to classes that can be mapped to either the antecedent or the consequent.

Definition 10 *L_{ac} is a DL language where A is an atomic class and i is an individual. If C and D are classes and R is a property, then C $\sqcap$ D, $\exists$ R.C and $\exists$ R.$\{i\}$ are also classes.*

Note: $\exists$ R.$\{$i$\}$ allows us to incorporate owl:hasValue in our language. Otherwise, nominals are not supported.

Definition 11 *L_a includes all classes in L_{ac}. Also, if C and D are classes then C $\sqcup$ D is also a class.*

Definition 12 *L_c includes all classes in L_{ac}. Also, if C and D are classes then $\forall$ R.C is also a class.*

Definition 13 *We now define a OWLII map ontology as a set of OWLII axioms of the form C $\sqsubseteq$ D, A $\equiv$ B, P $\sqsubseteq$ Q, P $\equiv$ Q, P$\equiv$ Q^-, where C is an L_a class, D is an L_c class, A, B are L_{ac} classes and P, Q are properties*

We have also defined a translation function $\mathcal{T}$ which takes a DL axiom of the form C $\sqsubseteq$ D, where C is an L_a class and D is an L_c class, and maps it into the FOL format for OWLII. This definition expands the one for DHL [5]. Due to limited space, we do not present it here. Please see our technical report [15] for details.

We end this section by discussing briefly how the REL statements described in Section 2 are incorporated into OWLII. REL statements can be translated into LAV rules using a minor variation of the function $\mathcal{T}$ mentioned above. When translating a statement REL(u, C_j, CE), the right-hand side of the subclass statement is the intersection of C_j and CE.

In OWL we express the REL statement by introducing four new predicates in a new name space "meta". They are meta:RelStatement, meta:contained, meta:container and meta:source. For example, the statement REL(http://sourceURL, CinemaDisplay, $\exists$ madeBy.“DELL”) can be expressed as follows. The meta:container will be the class expression that defines $\exists$madeBy.“DELL”, the meta:contained will be CinemaDisplay and meta:source will be http://sourceURL. The meta:RelStatement will encapsulate these three predicates. All these statements together say that in a data source located at http://sourceURL there are some individuals of class “CinemaDisplay” that are made by Dell. Due to limited space, we can not present the detailed description of the REL statements in this paper. Please see our technical report [15] for details.

4 OBII: A Semantic Web Query Answering System

In this section we present our adapted algorithm and briefly discuss the implemented architecture of OBII to give the reader a flavor of the working system.

Our algorithm is based on the PDMS algorithm, which takes as input a query, a set of views describing the sources and the maps. It computes a reformulation strictly in terms of the sources. As long as there are no cycles in the maps, the PDMS algorithm computes complete reformulations in polynomial time [8]. The PDMS imposes certain restrictions on the input for it to remain polynomial time. In our adaption we ensure that our input language conforms to these restrictions. The algorithm constructs a “rule-goal” tree: where goal nodes are labeled with atoms of the peer relations, and rule nodes are labeled with peer maps. Each AND-OR traversal from root to leaf of the rule-goal tree represents one way of answering the query. The reformulation then is obtained by the union of all of these traversals.

We now describe our source selection algorithm. Note: from here on we refer to an ontology that only has classes, properties, subClassOf and subPropertiesOf axioms as a simple ontology. For our algorithm we assume that all domain ontologies are simple ontologies given their present dominance in the Semantic Web. Our mapping ontologies are described in OWLII (see Section 3), and sources are described using the REL statements and only contain ABox assertions that use named classes and properties.

Given a conjunctive query we first split its body into its constituent atoms. This give us the starting point of our rule goal tree. Now for each atom in the query we attempt to expand it using maps and source descriptions that are available to the system. By recursively continuing with this expansion until we do not have any more maps (or source descriptions) available we build a rule goal tree in a similar fashion as the basic PDMS described above. However, our expansion is different due to the presence of domain ontologies with class and property taxonomies. We implement taxonomic reasoning as follows. For each query goal we use a reasoner to find the set of subclasses (sub properties). We then implement a variation of standard unification process that attempts to unify any of the sub predicates of a node with a given map. Our extended expansion algorithm is presented as Algorithm 1. Due to limited space we can not explain the algorithm in detail. Interested readers are referred to our technical report [15]. We however, note the following for readability: a) the `subPred` and the `match` routine implements the taxonomic reasoning b) the details of the MINICON routine can be found in Pottinger and Halevy [14] and c) the MapViews and SourceViews objects are data structures that store the LAV and GAV maps and LAV source descriptions respectively. Note, both of these sets are indexed by their source ontologies for efficient retrieval where the source ontology of a map is the left hand side (LHS) ontology of a GAV map or the right hand side (RHS) ontology of a LAV map. After we have built the rule goal tree we read off all of the unique sources from the leaves of the rule goal tree.

The basic PDMS does not allow cycles in the maps as this makes query answering undecidable. Therefore it is incomplete in the presence of cyclic maps. However, by selecting sources that are then input into an external reasoner, we have designed an algorithm that is complete even in the presence of cyclic maps.

Theorem 1 *The OBII source selection algorithm is complete in that given a set of simple domain ontologies, OWLII map ontologies and sources that conform to OWLII REL statements, it will identify exactly the potentially relevant sources. Furthermore, this algorithm terminates in polynomial time.*

Algorithm 1 OBII node expansion.

```
EXPAND(Node n, MapViews MV, SourceViews SV)
 1: sp ← SUBPRED(n.pred, n.ont)
 2: omaps ← $\{\mathrm{m} \mid (\mathrm{n.ont}, \mathrm{m}) \in \mathrm{MV}\}$
 3: for each v ∈ omaps do
 4:   if v has not been used then
 5:     if GAV(v) and MATCH(n, sp, HEAD(v)) then
 6:       for each sub goal ∈ v do
 7:         create an AND child goal node
 8:         for each child goal node cgn do
 9:           EXPAND(cgn, MV, SV)
10:     else if LAV(v) and MATCH(n, sp, b) for some b ∈
        BODY(v) then
11:       create an OR child node cgn for the view using
          MINICON
12:       EXPAND(cgn, MV, SV)
13: smaps ← $\{\mathrm{m} \mid (\mathrm{n.ont}, \mathrm{m}) \in \mathrm{SV}\}$
14: for each v ∈ smaps do
15:   if GAV(v) and MATCH(n, sp, HEAD(v)) then
16:     for each sub goal ∈ v do
17:       create an AND child goal node
18:   else if LAV(v) and MATCH(n, sp, b) for some b ∈
      BODY(v) then
19:     create an OR child node cgn for the view using
        MINICON
```

We now provide a sketch proof of the completeness of our algorithm.

Proof (sketch): *Observe that the original PDMS algorithm is incomplete in the presence of cycles solely due to the check that the same map is not used twice on any path from root to leaf (of course without this check, the tree could be infinite). If we omitted the check there would be goal nodes identical to goal nodes elsewhere in the tree. Any leaves of subtrees rooted at such goal nodes would be identical to leaves of the subtree rooted at the duplicated goal node. Thus they do not identify any new sources. Furthermore, since the PDMS algorithm is polynomial [8] and our algorithm only adds a test that is linear in the size of an ontology, our algorithm is polynomial.*

Figure 1 shows the architecture of our system. OBII executes in two asynchronous phases: a conversion phase and the query phase. The conversion phase, is done by the OWLIIRuleProcessor and occurs when a new source or a map ontology becomes available to the system. OWLIIRuleProcessor parses the OWLII map ontologies and REL meta data into OBII's MapKB as LAV and GAV rules. In this way the system's knowledge base always has the necessary information to reformulate a query.

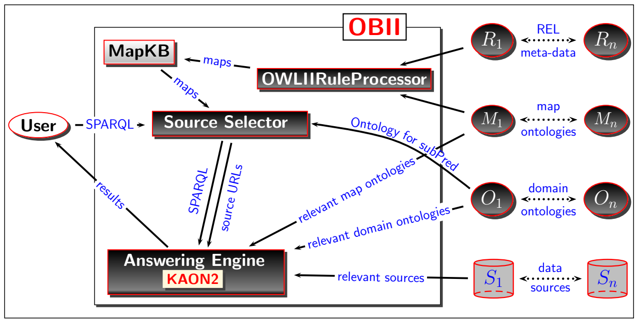


Figure 1. OBII architecture diagram

The query phase occurs when a query comes in. The Source Selector then takes the user's query and mapKB as input and uses our source selection algorithm to produce a set of selected sources which may contribute to an answer. Note, the module will also load some domain ontologies for use in taxonomic reasoning. The AnsweringEngine loads the sources selected by the Source Selector, the domain ontologies that are used in the node expansion and all the relevant map ontologies into KAON2. Then it issues the original query to the reasoner and formats the retrieved answers. Note: by loading only the used ontologies we provide a system that will scale well in terms of reasoning when we have a potentially large number of ontologies.

5 Evaluation

We implemented a workload generator that allows us to control several characteristics of our dataset. In generating the synthetic domain ontologies we decided to have on the average 20 classes and 20 properties (influenced by the dominance of small ontologies in the current Semantic Web). The class and property taxonomy have an average branching factor of 4 and an average depth of 3. In generating the OWLII map ontologies we chose to have an even distribution of various OWLII axioms and chose to map about 30% of the classes and 30% of the properties of a given domain ontology. The resulting GAV and LAV maps contain an average of 5 predicates with some maps containing up to 11 predicates. The average data source has 75 triples and uses 30% of the classes and 30% of the properties of the domain ontology that it commits to. We generate 200 random conjunctive queries with 1 to 3 predicates (75% are properties as opposed to a class).

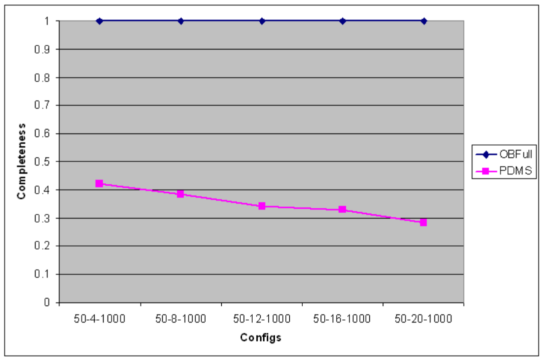
In choosing the configurations for our experiments we decided to vary two parameters: the number of data sources that commits to an ontology and the maximum number of maps required to translate from any source ontology to any target ontology. Since this is equivalent to what Halevy *et al.* [8] refer to as the diameter, we adopt their term in our discussion. We conducted two sets of experiments to evaluate the systems. In the first experiment (herein after referred to as experiment 1) we have varied the diameter. In the second experiment (herein after referred to as experiment 2) we have varied the number of data sources that commits to a given ontology. In both experiments we kept the number of ontologies to 50. We denote an experiment configuration as follows: (nO-nD-nS) where nO is number of ontologies, nD is the diameter and nS is the number of sources that commit to an ontology.

In our experiments the two main metrics we collected and examined are response time and the percentage of complete responses to queries. The response time is the time it takes from the issue of a query to the delivery of its result. We compared three systems in our experiments: A baseline system that loads all the ontologies and data sources and reasons over the complete knowledge base, a system that uses the original PDMS algorithm for source selection and our OBII system. We should note here that because the baseline system has a very different architecture, its response time is calculated differently. For the baseline system we add the load time (i.e. the time to load semantic web space) and the reasoning time to get the answers. The load time is added because as we are considering a dynamic environment, we should always work on fresh data, therefore each query results in a new knowledge base. For the other two systems, load time is calculated as the time to load the domain ontologies, the map ontologies that have been used in the source selection and the selected data sources. The response time for these two systems then is a sum of load time, source selection time and the reasoning time.

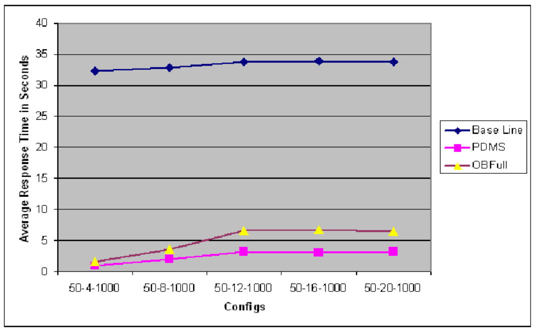
In determining the completeness of queries, we consider the baseline system's answers to be the reference set. This metric is the percentage of queries where a system returns the same answer as baseline, considering only queries that entail at least one answer. Note: KAON2 is only sound and complete for DL-safe conjunctive queries, therefore the baseline system may not be complete for some queries.

The first observation from our experiments is that OBII is complete with respect to KAON2, where as the basic PDMS drops in completeness as we add data sources or increase the diameter. This is evident from Figure 2(a) which shows the completeness over the set of queries for experiment 1. The second observation is that this completeness comes at a price. In Figures 2(b) and 3(a), where we show the average response time for each system, it is clear that OBII is approximately twice as slow as the basic PDMS. However, OBII is 10 times faster than the baseline system, which provides identical functionality. Note: the graph in Figure 3(a) uses a logarithmic scale.

This time penalty versus the PDMS is essentially unavoidable. In order to get complete answers OBII loads

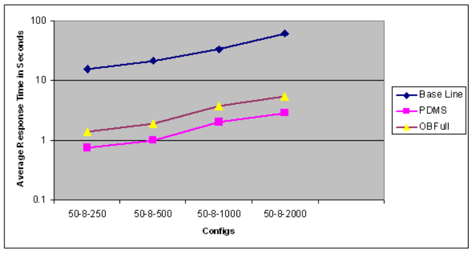


(a) Completeness of OBII and PDMS compared to baseline as we increase the diameter

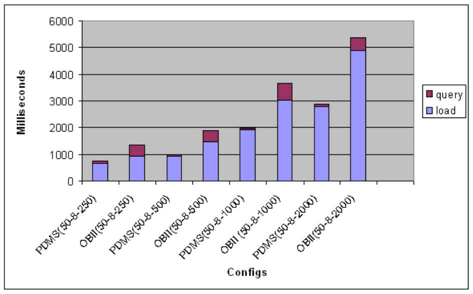


(b) Scalability w.r.t. the increase of diameter

Figure 2. Experiment 1: Varying Diameter.



(a) Scalability w.r.t. the increase of sources



(b) Query to Load time ratio for PDMS and OBII as we increase the sources

Figure 3. Experiment 2: Varying Number of Sources

more sources than PDMS. Recall this is because we incorporate taxonomic reasoning in our algorithm which the original PDMS does not. The cost of loading these sources generally dominates its response time. This is evident from Figure 3(b). However, for large diameter this dominance is reduced as OBII works with deeper rule goal trees. Even so, in the experiment that had the largest diameter (20), the response time for OBII is about 6.5 seconds (as opposed to PDMS's 3.5 seconds). Furthermore, PDMS is only 28% complete for this experiment.

6 Related Work

Serafini and Tamilin have developed the DRAGO system that reasons with multiple semantically related ontologies by using semantic mappings to combine the inferences of local reasoning of each ontology. Although their original work focused only on TBoxes they have recently extended this work to accommodate ABoxes to perform instance retrieval queries [16]. Their work is different from ours in that they consider the map processing (translating query to source) as part of the reasoning process. Therefore they have to work on a much larger knowledge base as they have to consider all the maps available to the system. The Piazza system [7] that uses the PDMS algorithm focuses more on integrating XML documents. The treatment of OWL is limited in this work and is described as a fairly difficult problem.

Haase and Motik [6] have described a mapping system for OWL and proposed a query answering algorithm. They identify a mapping language that is similar to ours. However, as their language adds rules to OWL, it is undecidable and as such they need to introduce restrictions to achieve decidability. Our language, on the other hand, is a sub language of a decidable language. Furthermore, similar to the DRAGO approach, Haase and Motik do not rely on an explicit reformulation step and process all the maps for a query reformulation.

Peer-to-peer systems like Bibster [2] and SomeWhere [1] have shown promises in providing query answering solutions for the Semantic Web. However, a peer-to-peer system needs special software installed at every server. Our system on the other hand makes use of the existing infrastructure of the Web.

A recent work by Liarou *et al.* [10] uses Distributed Hash Tables (DHT) to index and locate relevant RDF data sources. However, they do not address the schema mapping issue and therefore work on a single ontology environment. Furthermore, DHTs are targeted for a more P2P architecture as opposed to a client server web architecture.

7 Conclusion and Future Work

In this paper we have introduced a source selection problem for the Semantic Web. We have defined OWLII, a subset of OWL that is compatible with the GAV and LAV formalisms and which is more expressive than DHL. We have adapted a query reformulation algorithm to solve our source selection problem. As our experiments demonstrate our system is about 10 times faster than a naive approach of reasoning with all sources, and only about twice as slow as the incomplete information integration algorithm that it is based on. Our system returned correct answers with respect to KAON2 to 200 randomly generated queries in three different data configurations.

This work opens up some interesting avenues for further research. First, we have assumed that the domain ontologies only have simple taxonomic axioms. We have not considered the more advanced axioms that are available in OWL. One way to address this is to view the axioms as self-referential maps. Second, we want to investigate ways that will determine the best path to a translation. This may not always be the shortest path. Sometimes due to a translation that loses information, we may choose to follow a path that results in the least loss of information as opposed to the least number of translations. Third, we intend to explore methods of automatically generating high quality REL statements. Finally, we have observed that our rule-goal trees get very big as we increase the diameter of the system. We intend to explore optimizations that remove redundancy from these trees.

8 Acknowledgment

We are grateful to the U.S. Department of Energy for funding this work under the DE-FG02-05ER84171 SBIR grant.

References

[1] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. SomeWhere in the semantic web. In F. Fages and S. Soliman, editors, *Principles and Practice of Semantic Web Reasoning, Third International Workshop, PPSWR*, Lecture Notes in Computer Science, pages 1–16. Springer, 2005.

[2] J. Broekstra, M. Ehrig, P. Haase, F. Harmelen, M. Menken, P. Mika, B. Schnizler, and R. Siebes. Bibster: A semantics-based bibliographic peer-to-peer system. In *Third International Semantic Web Conference (ISWC 2004)*, pages 122–136. Springer-Verlag, 2004.

[3] D. A. Dimitrov, J. Heflin, A. Qasem, and N. Wang. Information integration via an end-to-end distributed semantic web system. In *5th International Semantic Web Conference*, Lecture Notes in Computer Science, pages 764–777. Springer, 2006.

[4] L. Ding, T. Finin, A. Joshi, Y. Peng, R. Pan, and P. Reddivari. Search on the semantic web. *IEEE Computer*, 10(38):62–69, October 2005.

[5] B. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Proceedings of WWW2003*, Budapest, Hungary, May 2003. World Wide Web Consortium.

[6] P. Haase and B. Motik. A mapping system for the integration of OWL-DL ontologies. In A. Hahn, S. Abels, and L. Haak, editors, *Proceedings of the first international ACM workshop on Interoperability of Heterogeneous Information Systems (IHIS'05)*, pages 9–16. ACM, 2005.

[7] A. Halevy, Z. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The Piazza peer-data management system. *Transactions on Knowledge and Data Engineering, Special issue on Peer-data management*, 16(7):764–777, 2004.

[8] A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proc. of ICDE*, 2003.

[9] I. Horrocks and S. Tessaris. A conjunctive query language for description logic Aboxes. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000)*, pages 399–404, 2000.

[10] E. Liarou, S. Idreos, and M. Koubarakis. Continuous RDF query processing over DHTs. In *Proceedings of 6th International Semantic Web Conference / 2nd Asian Semantic Web Conference (ISWC/ASWC 2007)*, pages 324–339, 2007.

[11] B. Motik and U. Sattler. A comparison of reasoning techniques for querying large description logic aboxes. In *Proceedings of 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 227–241, Phnom Penh, Cambodia, 2006.

[12] B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *In Third International Semantic Web Conference (ISWC 2004)*, volume 3298 of *Lecture Notes in Computer Science*, pages 549–563. Springer, 2004.

[13] P. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language semantics and abstract syntax. Recommendation, February 2004. http://www.w3.org/TR/owl-semantics/.

[14] R. Pottinger and A. Halevy. MiniCon: A scalable algorithm for answering queries using views. *The VLDB Journal*, 10(2-3):182–198, 2001.

[15] A. Qasem, D. A. Dimitrov, and J. Heflin. An efficient and complete distributed query answering system for semantic web data. Technical Report LU-CSE-07-007, Lehigh University, 2007.

[16] L. Serafini and A. Tamilin. Instance migration in heterogeneous ontology environments. In *Proceedings of 6th International Semantic Web Conference / 2nd Asian Semantic Web Conference (ISWC/ASWC 2007)*, pages 452–465, 2007.