

# Goal Node Search for Semantic Web Source Selection

Lehigh University Technical Report LU-CSE-08-010

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

Septmeber 10, 2008

## Abstract

*We present an efficient search approach for selecting all potentially relevant data sources for a given a query and a Semantic Web Space. We use map ontologies to align heterogeneous domain ontologies. This allows us to select data sources that may be relevant to query but generally do not describe their data directly in terms of the ontology of the query. The knowledge representation language we use to describe the Semantic Web Space is a subset of OWL that is compatible with Local-as-View (LAV) and Global-as-View (GAV) rules. In our approach, we first translate the Semantic Web Space into a set of LAV/GAV rules. Given a query the “Goal Node Search” algorithm identifies all possible paths that can be found by applying the LAV/GAV rules to each query subgoal or its expansions. We have incorporated the algorithm in OBII, a Semantic Web query answering system and evaluated its performance versus OBII’s original algorithm. The new algorithm is a significant improvement on the original source selection algorithm of OBII. First, it allows a more expressive knowledge representation language to describe domain ontologies than the original algorithm. Second, it is about three times more efficient than the original source selection algorithm when performing similar tasks. In addition the new algorithm is conceptually simpler than the original source selection algorithm.*

## 1. Introduction

The promise of the Semantic Web is that someday we will be able to use the Web as a global knowledge base. Ontologies, expressed in a standard logic language with formal semantics, will be used in concert with web data in order to develop powerful query systems that will be used by both human and software agents alike. Although, significant progress has been made toward realizing this vision, there are major engineering challenges that need to be overcome for the Semantic Web to reach its true potential [13]. In this paper, we explore the scalability and the heterogeneity challenges and present an approach that addresses them in a unified way.

To address the scalability issue in a large, distributed and dynamic setting such as the Semantic (Web), it is often desirable to identify which sources might be potentially relevant to a query before these sources are accessed. We consider an approach for identifying the minimal set of potentially relevant Semantic Web data sources for a given query. In our framework, a *Potentially Relevant* data source can make assertions about its content's relevance by means of REL statements. A data source provider can use REL statements to summarize the contents of a data source in terms of classes whose instances the data source has information about and the properties used to relate them. REL statements allow us to develop algorithms to choose data sources that may be relevant to a query and ignore sources that are definitely irrelevant.

Heterogeneity is an inherent product of autonomy and autonomy is a key success factor behind the Web. Since, the Semantic Web is essentially just an extension of the original Web it also is inherently autonomous and therefore heterogeneous. The standard language for defining ontologies is the Web Ontology Language (OWL) [14]. However, when many ontologies and data sources are created independently of one another, it is likely that many of them will refer to the same or similar concepts, although they may use different terminology. Therefore, while some of the data sources may contain data described directly in terms of a given query ontology, others may not. When identifying sources it is essential 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 and the term domain ontologies for all other ontologies. The map ontologies are like any other OWL ontology except they consist solely of axioms that relate concepts from one ontology to concepts of another ontology.

Since our maps are also defined in OWL, the alignments specified therein are shareable via the Web; anyone 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. Note that we will not have alignments between all pairs of ontologies, but it should be possible to compose an alignment from existing alignments. Moreover, aligning ontologies is a difficult but relevant problem but this is not the focus of this paper.

In our previous work we have adapted an information integration algorithm to select the potentially relevant sources. In this work, we present a "Goal Node Search" algorithm and make the following two technical contributions:

1. Our new algorithm is complete for more expressive domain ontologies than the original OBII algorithm.
2. We show that Goal Node Search is up to three times faster at source selection than our original algorithm.

We also note that the new algorithm is conceptually simpler than the original source selection algorithm.

The rest of the paper is organized as follows: Section 2 provides relevant background information. Section 3 defines the subset of OWL used in this paper. Section 4 describes the Goal Node Search algorithm. Section 5 describes experiments that we have conducted to evaluate the algorithm. Section 6 compares some other research with our work and Section 7 concludes and discusses future work.

## 2 Background

In this section, we first describe the source selection problem for the Semantic Web in terms of identifying potentially relevant sources. We then give a high level description of the Ontology Based Information Integrator (OBII), a query answering system that adapts this framework.

### 2.1 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 [14] 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 [14].

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

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

**Definition 4 (Conjunctive Query Form)** *A conjunctive query has the form  $H(\bar{X}) :- B_1(\bar{X}_1), \dots, B_n(\bar{X}_n)$  where  $\bar{X}$  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  $\theta$ , we use  $Q\theta$  as a shorthand for  $B_1\theta \wedge B_2\theta \dots \wedge B_n\theta$ .

**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  $\theta$  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 [12].

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:  $\mathcal{U} \times \mathcal{S} \times \mathcal{C} \rightarrow 2^{\mathcal{D}}$  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  $\text{SrcInst}(u, s, c) = \{(a^I) \mid a : c \in s(u)\}$ .

2.  $\text{SrcInstP}: \mathcal{U} \times \mathcal{S} \times \mathcal{P} \rightarrow 2^{\mathcal{D} \times \mathcal{D}}$  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,  $\text{SrcInstP}$  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  $\text{SrcInstP}(u, s, p) = \{\langle a^I, b^I \rangle \mid \langle a, b \rangle : p \in s(u)\}$ .

The REL statement has the following two forms: a)  $\text{REL}(u, C_j, CE)$  and b)  $\text{REL}(u, P_j, P')$  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.  $r = \text{REL}(u, C_j, CE)$ ,  $\text{SrcInst}(u, s, C_j)$  is  $\neq \perp$  and  $\text{SrcInst}(u, s, C_j) \sqsubseteq CE$  or  $r = \text{REL}(u, P_j, P')$ ,  $\text{SrcInstP}(u, s, P_j)$  is  $\neq \perp$  and  $\text{SrcInstP}(u, s, P_j) \sqsubseteq P'$*
2. *for all  $a: C_j \in s(u) \exists r \in \mathcal{R}$  s.t.  $r = \text{REL}(u, C_j, CE)$  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.  $r = \text{REL}(u, P_j, P')$  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:

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

**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.*

## 2.2 OBII:A Semantic Web query answering system

OBII is a Semantic Web query answering system that uses the source selection framework and a DL reasoner to provide answers to conjunctive queries posed using SPARQL syntax. Our algorithms are designed to work with a subset of OWL DL (a decidable and commonly used fragment of OWL) which is compatible with Global-As-View (GAV) [4] and Local-As-View (LAV) [10] rules. We refer to this subset as OWL for Information Integration (OWLII).

Currently the implemented system supports a restricted subset of SPARQL queries, OWLII ontologies and data sources that commit to them. Given a SPARQL query OBII uses the goal node search algorithm (or any algorithm that is compatible with the source selection framework) to select the potentially relevant sources. These sources are then loaded into a sound and complete DL reasoner to infer answers. Figure 1 shows the architecture of the system. OBII executes in two asynchronous phases: a conversion phase and the query phase. The conversion phase, is implemented by the OWLIIRuleProcessor and occurs when a new source or a map ontology becomes available to the system (or at system startup to initialize the MapKB). 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 select sources given a query.

OWLIIRuleProcessor translates the SWS into a set of LAV/GAV rules. An OWLII axiom is parsed into the system as a LAV or GAV rule(s) and stored in a MapView object. A REL statement is parsed into the system as a LAV rule and source URL pair and stored in a SourceView object. A collection of MapView objects and SourceView objects is maintained by MapKB.

The SourceSelector encapsulates the source selection algorithm. We have implemented the goal node search algorithm and the original source selection algorithm in this module. The output of this module is a set of potentially relevant sources. AnsweringEngine wraps the KAON2 reasoner and provides convenience methods for loading multiple ontologies and formatting returned answers.

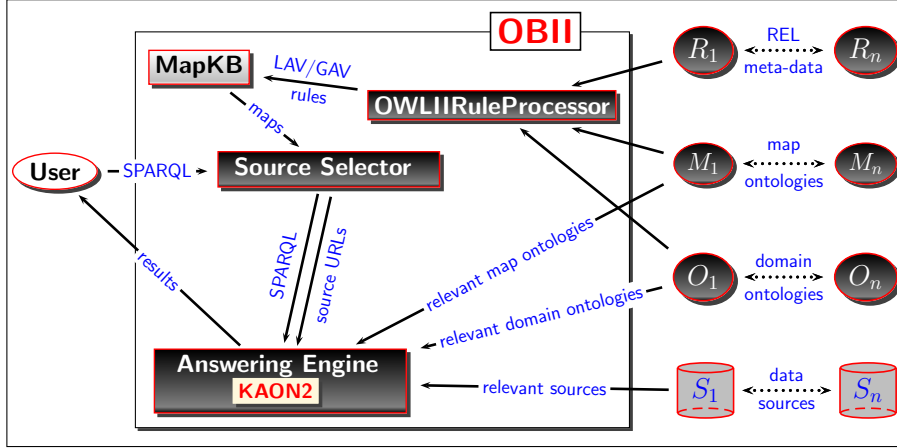


Figure 1. OBII architecture diagram with arrows showing the flow of information when processing a query.

### 3 Knowledge Representation Language: OWLII

We follow a process similar to Grosz *et al.* [5] to define OWLII as a specific subset of DL. 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 (OWLII ontology)** We now define a OWLII 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.

In Qasem *et al.* [15], we have 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 to a FOL implication. Note: an OWLII  $\equiv$  can also be translated into two FOL implications using the same function. Further, the syntax and the semantics of REL statements described in Qasem *et al.* [15] allow REL statements to be translated into LAV rules using a minor variation of the function  $\mathcal{T}$ . A REL statement in our framework is a LAV rule and the URL of the data source.

### 4 Goal Node Search Algorithm

Given a conjunctive query and a set of LAV/GAV rules, the Goal Node Search algorithm identifies all possible additional subgoals that can be found by applying the LAV/GAV rules to each query subgoal or its expansions. Identifying these subgoals can be viewed as a search problem where each node of the search tree is either an original or an expanded subgoal; and the search task is to identify all possible paths that can be derived from applying the LAV/GAV rules to the nodes. As the search space for the algorithm is the set of all possible expanded goal nodes, we refer to the algorithm as the Goal Node Search algorithm.

We implement the search by maintaining two lists: an open list of nodes to be expanded and a closed list of nodes that have been expanded. We continue to expand the open list until it is empty while adding the node that has been expanded to

the closed list and adding the expanded new nodes to the open list. The open list is initialized with goal nodes created from the subgoals of the query to give us the starting point of the search.

The EXPAND routine presented as Algorithm 1 performs LAV or GAV expansions given a goal node. Let  $B(\bar{X})$  and  $H(\bar{X})$  (sometimes subscripted) be unary or binary atoms with a vector  $\bar{X}$  of arguments. A GAV rule has the form  $\mathbb{H}(\bar{X}) :- B_1(\bar{X}_1) \wedge B_2(\bar{X}_2) \wedge \dots \wedge B_n(\bar{X}_n)$ , where the left hand side of the  $:-$  is called the head and the right hand side is called the body. A LAV rule has the form  $\mathbb{H}(\bar{X}) \subseteq B_1(\bar{X}_1), B_2(\bar{X}_2), \dots, B_m(\bar{X}_m)$  where the left hand side of the  $\subseteq$  is called the head and the right hand side is called the body. The routines HEAD(v) and BODY(v) returns the head or the body of given rule.

In an expansion we only consider the rules whose head atom (or head atoms in the case of LAV) has the same ontology as the goal node. This reduces the size of MV as we can ignore all the rules that will have no chance of matching with the goal node. If the head of a GAV rule unifies with the goal node, the expansion includes a set of nodes corresponding to the body of the GAV rule. If any atom from the the body of a LAV rule unifies with the node, then the expansion includes the head of the LAV rule. In both cases, variables from the goal node are substituted into the generated nodes. In order to guarantee termination i.e. to avoid cyclic expansion, we never expand a node twice. This is implemented by checking to make sure that a member of open list that is about to be expanded is not already in the closed list.

The Goal Node Search algorithm is different from backward-chaining in two ways. First, it has LAV rules, and thus is more expressive than Horn; and second given that we are not attempting to return bindings for the variables, we can prune many redundant subtrees from the search space.

In addition to pruning nodes that exactly match a node in the closed list, we can prune nodes that are superseded by a node in the closed list. We say node  $n$  supersedes another node  $m$  if the result from a query using  $n$ 's predicate is necessarily a superset of the result from a query using  $m$ 's predicate. Syntactically,  $n$  supersedes  $m$  if there is a unification involving only substitutions to variables from  $n$ . For example,  $p(x, y)$  supersedes  $p(x, \text{CONST})$  but  $p(x, x)$  does not supersede  $p(\text{CONST1}, \text{CONST2})$  where  $\text{CONST1}$  is not equal to  $\text{CONST2}$  etc. Essentially, supersedes defines a partial order over the nodes.

Using a supersede relationship between nodes as opposed to a strict match to decide if a node has been expanded allows us to keep only the most general node in the list. This reduces the size of the list that we are maintaining. We use a set of  $\langle p, R \rangle$  to hold the nodes, where  $p$  is a predicate of an atom and  $R$  is a list of argument patterns sorted by the partial order introduced by the supersede. This provides for a fast and efficient data structure as we can quickly decide if a node is superseded or not and therefore the list updates become very efficient.

Once the open list is empty i.e. we have found all possible expansions, we use the REL statements (stored in MapKB as SourceView objects) to complete the source selection process. If a node in the closed list matches with a Source View, we extract the source URL and add it to the list of selected data sources that will loaded in the reasoner.

---

**Algorithm 1** OBII node expansion.

---

EXPAND( $n$ :Node, MV:set of MapView)

- 1: expandedNodes  $\leftarrow \emptyset$
  - 2: omaps  $\leftarrow \{m \mid (n.\text{ont}, m) \in \text{MV}\}$
  - 3: **for** each  $v \in \text{omaps}$  **do**
  - 4:   **if** GAV( $v$ ) **and** UNIFY( $n$ , HEAD( $v$ )) **then**
  - 5:     expandedNodes  $\cup$  GAVEXPAND( $n$ ,  $v$ )
  - 6:   **else if** LAV( $v$ ) **and** UNIFY( $n$ ,  $b$ ) for some  $b \in \text{BODY}(v)$  **then**
  - 7:     expandedNodes  $\cup$  LAVEXPAND( $n$ ,  $v$ )
  - 8: **return** expandedNodes
- 

The Goal Node Search algorithm terminates when the open list is empty. After initialization the only way for a node to enter the open list is as a result of an expansion using a GAV or a LAV rule. As GAV and LAV rules are function free (by definition) the open list will be finite if we avoid all cyclic expansions. As mentioned before we never expand a node twice by checking to make sure that a member of open list that is about to be expanded is not already in the closed list. Therefore, the Goal Node Search Algorithm will terminate. Further, since all the axioms of the Semantic Web Space are translated into LAV/GAV rules, the set of all possible inferences within the domain ontologies can be viewed as the set of all possible expansions using the LAV/GAV rules. Also, since after the expansion we load the sources in their entirety into a reasoner and issue the original query, any inferences due to a combination of sources will occur in the reasoner. Therefore, the Goal Node Search algorithm is sound and complete given a set of OWLII ontologies, sources that conform to OWLII REL statements and a sound and complete DL reasoner.

Because the algorithm is basically a breadth-first search, its space complexity is  $O(b^d)$ , where  $b$  is the average number of matches for a goal node and  $d$  is the number of expansions that is needed to reach the data sources. Note: because we need to identify all sources, we need to find all possible paths as opposed to just one path from root to goal. Furthermore, since  $b$  is finite and we never expand a node that has already been expanded, the algorithm will terminate. However, since our closed list results in a pruning of redundant nodes we can also describe space complexity in terms of  $n$ , the number of predicates, and  $m$ , the number of constants in the MapKB. There are at most  $n \cdot m^2$  distinct nodes (due to the supersedes relationship) resulting in a space complexity is also  $O(nm^2)$ . When each of these nodes is expanded, it must be checked if it is in the closed list, resulting in an  $m^2$  linear search. Thus, the overall time complexity is  $O(nm^4)$ . However, we believe that it is unlikely that all combinations of constants will occur in the MapKB thus average space and time complexity will be much smaller. In Section 5, we perform an empirical analysis of the system’s performance.

Before ending this section we make the following observation about the Goal Node Search algorithm’s difference with our previous source selection algorithm. In the former algorithm, the domain ontologies were not translated into LAV/GAV rules. In order to ensure that we do not miss any inferences within a domain ontology, when looking for a match we recursively traversed the descendants of the node to find the set of subclasses (sub properties) of a given node. We then used a variation of standard unification process to unify any of the subclasses (sub properties) of a node with a given rule. This approach restricted the expressive boundary of the domain ontologies and hence the previous algorithm was only sound and complete for domain ontologies that are simple taxonomies.

In the previous algorithm, we have also used a specially optimized LAV expansion technique and a bulkier data structure (a rule goal tree instead of the open list - closed list data structure). These artifacts provide useful optimization/information if the data sources are databases that must be queried to get relevant information (as is the case in a typical information integration scenario). Since, our selected data sources are loaded in their entirety into the reasoner this additional optimization and information become superfluous. As we show in Section 5 we have sped up source selection by “trimming” down the source selection process.

## 5 Evaluation

We have implemented a workload generator that allows us to control several characteristics of the dataset we used. 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). We have generated two sets of domain ontologies for our experiment. The first set of ontologies are simple taxonomies with a branching factor of 4 and an average depth of 3. Since the original algorithm is only complete for such ontologies, we use them to compare the performance of the two algorithms given equivalent tasks. The second set of domain ontologies that we have generated are OWLII ontologies with an average axiom depth of 4 and an even distribution of various constructors (intersection, union) and restrictions (existential and universal) in the axioms. This set is used to compare the completeness of the two algorithms.

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 generated 200 random conjunctive queries with 1 to 3 predicates (75% are properties as opposed to a class).

We have conducted all experiments on a SUN workstation featuring 4GB main memory and 64-bit dual core AMD Opteron CPUs running at 2.2 GHz. In the 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 (hereinafter called the baseline system), OBII using its original algorithm for source selection (hereinafter called OBII-original) and OBII using the new Goal Node Search algorithm (hereinafter called the OBII-GNS). For the baseline system we add the load time (i.e. the time to load the 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.

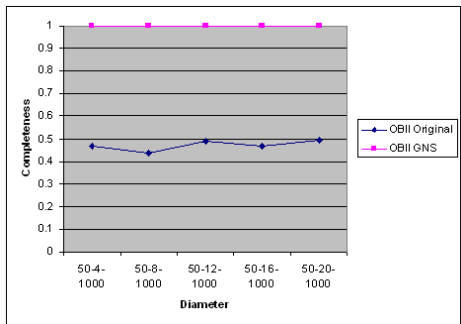
In determining the completeness of queries, we consider the baseline system’s answers to be the reference set. This metric is the percentage of queries where a system returns the same answer as baseline, considering only queries that entail at least

one answer. Note, however, that our implementation is restricted by the query language supported by KAON2 which we use as our reasoner. KAON2 can only answer the so called DL-safe conjunctive queries [6]. Therefore, the baseline system may not be complete for some queries.

In choosing the configurations for the experiments we decided to vary two parameters: the number of data sources that commit to an ontology and the maximum number of maps required to translate from any source ontology to any target ontology. This number is referred to as the diameter by Halevy *et al.* [8]. We adopt this 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 commit to a given ontology. In both experiments we kept the number of ontologies to 50<sup>1</sup>. We denote an experiment configuration as follows: (nO-nD-nS) where nO is number of ontologies, nD is the diameter and nS is total number of sources that commit to a particular ontology.

The first observation is that OBII-GNS is empirically complete in all configurations for data sets with domain and map ontologies expressed in OWLII. OBII-original on the other hand is not complete for domain ontologies expressed in OWLII and its completeness in any configuration is always below 50%. This is evident from Figure 2.

The second observation is that OBII-GNS shows significant improvement in source selection time as we increase the diameter. This is evident from Figure 3. Furthermore, OBII-GNS performs increasingly better in the higher diameter configurations. This is because in the higher diameters OBII-original has to work with very large AND-OR graphs. We see very little performance gain in lower diameters. In fact in diameter 4 the performance of the two algorithms is very similar. We note that the performance of both algorithms are worse in diameter 12 than in diameter 16. This is because the average number of nodes in diameter 12 is about a 1000 more than the average number of nodes in diameter 16. This is most likely due to the randomness of the synthetic data sources and the query patterns. However, the relative performance gain still holds in all the diameters. The source selection performance difference between OBII-GNS and OBII-original holds fairly constant in all configurations.



**Figure 2. Completeness of OBII-GNS vs. OBII-Original on OWLII ontologies, with 50 ontologies, and 1000 sources when varying diameter.**

The third observation is that this source selection performance gain does not translate into a significant overall response time gain for OBII-GNS. This is evident from the graph in Figure 4 which shows a very similar performance for both systems. This is because when considering overall response time, the load time significantly dominates the source selection and query time. Source selection and query accounts for less than 10% of overall response time. However, in large diameters this dominance is reduced. If the query was very selective and required only a few sources, then we would expect to see OBII-GNS outperform OBII-orig in overall response time also.

We note, however, both algorithms are about 10 times faster than the baseline system. Furthermore, we have noticed that increasing the diameter does not have a significant effect on the overall response time of any of the systems. However, increasing the number of sources results in a slow linear growth. In the case of OBII-GNS, it is from 1.3 seconds when there are 250 sources to 5.3 seconds when there are 2000 sources.

<sup>1</sup>An analysis of Swoogle’s [3] data reveals that over 90% of Semantic Web data commits to the top 50 ontologies.

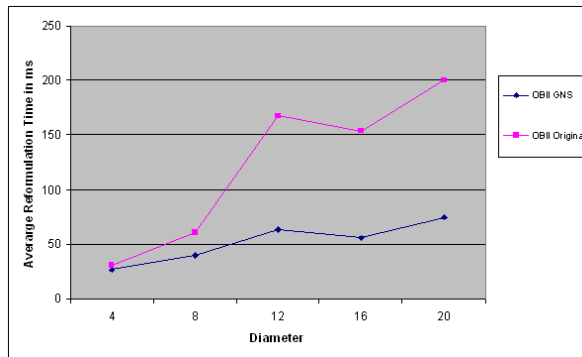


Figure 3. Source selection of OBII-GNS and OBII-Original on simple ontologies, with 50 ontologies, and 1000 sources when varying diameter.

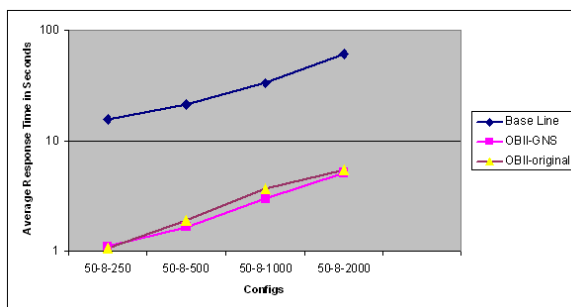


Figure 4. Overall performance of all considered systems on simple ontologies, with 50 ontologies and a diameter of 8, when varying number of sources.

## 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] uses the PDMS and 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.* [11] 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 presented a Goal Node Search algorithm that provides an efficient solution to the source selection problem in the Semantic Web. The algorithm is conceptually simpler than the original source selection algorithm, it is complete for more expressive domain ontologies and up to three times faster at source selection when given equivalent workloads. We put forward, 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.

We note that although we have significant speedup in source selection time, the measurements show a small gain in overall performance. We believe this is a significant result! We have demonstrated that contrary to conventional wisdom, the bottleneck of distributed Semantic Web queries may not be reasoning, but instead is the latency involved in fetching and parsing documents. We plan to consider the possibility of parallelization of source loading and further constraining the potentially relevant sources. In the near future, we also intend to explore methods of automatically generating high quality REL statements and explore the expressive boundary of OWLII and investigate if we can incorporate additional mapping rules than just LAV and GAV.

## 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] 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.
- [4] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.
- [5] B. Groszof, 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. Suci, 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. Suci, 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] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *22nd International Conference on Very Large Data Bases*, Bombay, Sept. 1996.
- [11] 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.
- [12] 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.
- [13] Z. Pan, A. Qasem, and J. Heflin. An investigation into the feasibility of the semantic web. In *Proc. of Twenty First National Conference on Artificial Intelligence (AAAI 2006)*, 2006.
- [14] 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/>.
- [15] A. Qasem, D. A. Dimitrov, and J. Heflin. Efficient selection and integration of data sources for answering semantic web queries. In *ICSC 08: Proceedings of the Second IEEE International Conference on Semantic Computing*. IEEE Computer Society Press, 2008.
- [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.