Goal Node Search for Semantic Web Source Selection

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

We present an efficient search approach for selecting all potentially relevant data sources for a conjunctive Semantic Web query. We use map ontologies to align heterogeneous domain ontologies. This allows us to select data sources that may be relevant to the query but generally do not describe their data directly in terms of the ontology of the query. The "Goal Node Search" algorithm is a significant improvement on our original source selection algorithm. The new algorithm allows a more expressive knowledge representation language to describe domain ontologies and it is about three times more efficient than the original source selection algorithm when performing similar tasks.

1. Introduction

The promise of the Semantic Web is that someday we will be able to use the Web as a global knowledge base. 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. 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 we consider a "source selection" approach for identifying the minimal set of potentially relevant Semantic Web data sources for a given query. In our framework 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 without actually querying them. Given a Semantic Web Space (ontologies and data sources), a query and a set of REL statements, the source selection problem is to identify all potentially relevant sources. We use the term potentially relevant as opposed to relevant because we can not determine if a source is relevant until it is actually queried.

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. 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. The map ontologies are written in the Web Ontology Language (OWL) [7], just like any other OWL ontology in the Semantic Web. However, they consist solely of axioms that relate concepts from one ontology to concepts of another ontology. In our discussion we use the term domain ontology to refer to all non map ontologies.

The map ontologies and the REL statements can be used in conjunction with a Description Logic (DL) reasoner to provide an efficient query answering solution for the Semantic Web. Using the map ontologies, a query expressed in terms of a particular domain ontology can be translated to queries in terms of other domain ontologies. The translated queries and the original query can be used with REL statements to select data sources that could potentially contribute to the answer of the query. These selected sources can then be loaded into a DL reasoner to get the answer(s) for the query. Since, the selected sources are loaded in their entirety into a reasoner, any inferences due to a combination of these selected sources will also be computed by the reasoner. Source selection helps reduce the size of the knowledge base during the expensive DL reasoning phase (as we load only the relevant data sources for a query and ignore the rest). Since, DL reasoning time is a function of the size of the knowledge base, source selection provides a more efficient solution. In this paper, we present a "Goal Node Search" (GNS) algorithm that implements an instance of the this approach.

We make two technical contributions in this paper. First, we present a new algorithm that is complete for more expressive domain ontologies than the original source selection algorithm. Second, we show that GNS is up to three times faster at source selection than our original algorithm.

Our algorithm is complete for a subset of OWL which is compatible with Global-As-View (GAV) [2] and Local-As-View (LAV) [6] information integration formalisms. We refer to this subset as OWL for Information Integration (OWLII). An OWLII ontology consists of a restricted set of OWL axioms. In OWLII, named classes, $\sqcap$, $\exists$ and single valued nominals (owl:hasValue) are allowed on both left hand side (LHS) and right hand side (RHS) of an $\equiv$ axiom; in addition $\sqcup$ is allowed on the LHS and $\forall$ is allowed on the RHS of a $\sqsubseteq$ axiom; in the case of a property axiom only named properties and inverse is allowed. The REL statements can be translated into LAV rules and therefore can be expressed in OWLII. We note that since our maps are defined in OWL as opposed to a specialized mapping language, anyone can use existing tools and infrastructure to create alignments and publish them in OWL for others to use.

We have implemented GNS in OBII, a Semantic Web query answering system that incorporates the source selection framework. Figure 1 shows the architecture of OBII with arrows showing the flow of information when processing a query. OWLIIRuleProcessor translates the ontologies and REL meta statements into LAV/GAV rules. The SourceSelector implements the GNS algorithm and AnweringEngine loads the selected sources into KAON2 (a DL reasoner) and obtains the answer to a query after the reasoning. The query in OBII uses SPARQL [1] syntax.

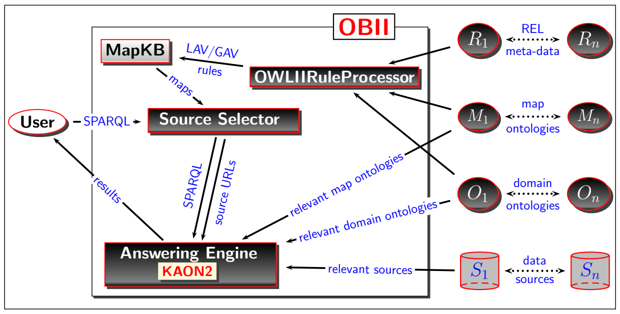

Figure 1. OBII Architecture

Due to limited space we can not describe the source selection framework and the details of OWLII in this presentation. We refer the readers to Qasem et al. [8] for a formal description of the source selection problem and Qasem et al. [9] [2] for an extended exposition of GNS algorithm. The rest of the paper is organized as follows: Section 2 describes the GNS algorithm. Section 3 describes experiments that we have conducted to evaluate the algorithm. Section 4 compares some other research with our work and Section 5 concludes and discusses future work.

2 Goal Node Search Algorithm

Given a conjunctive query and a set of LAV/GAV rules, the GNS 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 by 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 GNS algorithm.

We use a standard technique for pruning redundant trees: we maintain 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. 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 the open list that is about to be expanded is not already in the closed list. The GNS 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.

An important feature of our algorithm is that 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. 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 $\mathcal{R}$ 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

[1]http://www.w3.org/TR/rdf-sparql-query/
[2]http://www3.lehigh.edu/engineering/cse//research/technicalreports.asp

have found all possible expansions, we use the REL statements (stored in MapKB as SourceView objects, see Figure 1) 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 be loaded in the reasoner.

Algorithm 1 OBII node expansion.

```
EXPAND(n:Node, MV:set of MapView)
1: expandedNodes ← $\emptyset$
2: omaps ← $\{\mathrm{m} \mid (\mathrm{n.ont}, \mathrm{m}) \in \mathrm{MV}\}$
3: for each v ∈ omaps do
4:     if GAV(v) and UNIFY(n, HEAD(v)) then
5:         expandedNodes ∪ GAVEXPAND(n, v)
6:     else if LAV(v) and UNIFY(n, b) for some b ∈ BODY(v) then
7:         expandedNodes ∪ LAVEXPAND(n, v)
8: return expandedNodes
```

The GNS 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 GNS algorithm will terminate.

In our algorithm, the domain ontologies were not translated into LAV/GAV rules. 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.

3 Evaluation

We have evaluated the system using synthetic ontologies and data sources. 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.* [5]. We adopt this term in our discussion. We conducted two sets of experiments to evaluate the systems. In the first experiment we have varied the diameter. In the second experiment 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. 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 % (Figure 2).

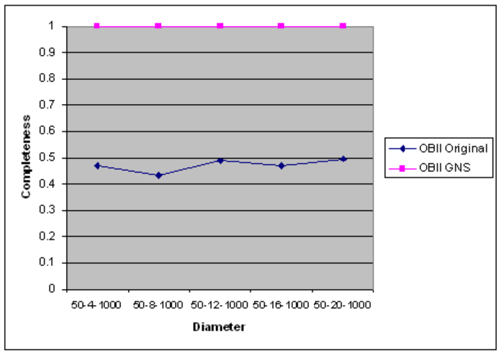

Figure 2. Completeness of OBII-GNS vs. OBII-Original on OWLII ontologies, with 50 ontologies, and 1000 sources when varying diameter.

The second observation is that OBII-GNS shows significant improvement in source selection time as we increase the diameter (see 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.

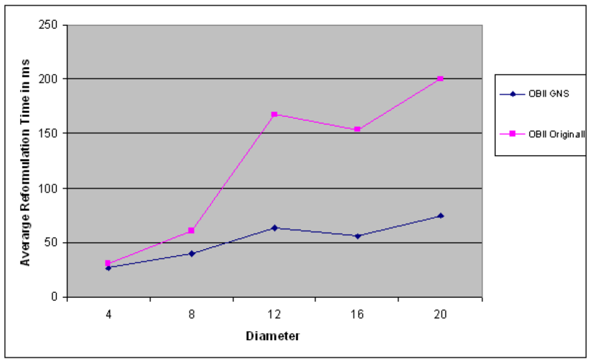

Figure 3. Source selection of OBII-GNS and OBII-Original on simple ontologies, with 50 ontologies, and 1000 sources when varying diameter.

The third observation is that this source selection performance gain does not translate into a significant overall response time gain for OBII-GNS (Figure 4). 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.

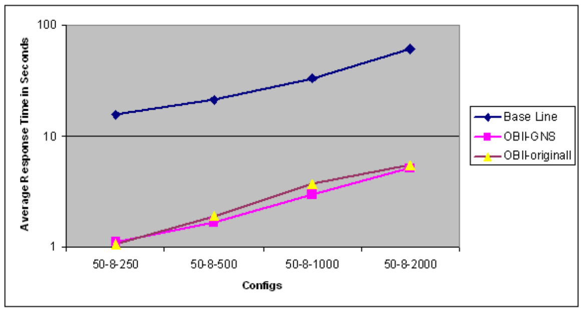

Figure 4. Overall performance of all considered systems on simple ontologies, with 50 ontologies and a diameter of 8, when varying number of sources.

We note, however, both algorithms are about 10 times faster than the baseline system in which we loaded all the files into KAON2. Furthermore, we have noticed that increasing the diameter does not have a significant effect on the overall response time of any of the systems. Increasing the number of sources, however, 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.

4 Related Work

The Piazza system [4] uses the PDMS to integrate XML documents but does not address OWL documents. Haase and Motik [3] 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 introduced restrictions to achieve decidability. Our language, on the other hand, is a sub language of a decidable language. Peer-to-peer systems like Bibster [1] have shown promise in providing query answering solutions for the Semantic Web. However, a peer-to-peer system needs special software installed at every server.

5 Conclusion and Future Work

We have presented a GNS 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 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 source loading parallelization and further constraining the potentially relevant sources.

6 Acknowledgment

Supported by U.S. Department of Energy (DE-FG02-05ER84171 SBIR) grant.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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/.

[8] 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.

[9] A. Qasem, D. A. Dimitrov, and J. Heflin. Goal node search for semantic web source selection. Technical Report LU-CSE-08-010, Lehigh University, 2008.