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

Abir Qasem[1], Dimitre A. Dimitrov[2], and Jeff Heflin[1]

[1] Lehigh University, 19 Memorial Drive West, Bethlehem, PA 18015
{abq2, heflin}@cse.lehigh.edu
[2] Tech-X Corporation, 5621 Arapahoe Avenue, Suite A, Boulder, CO 80303
dad@txcorp.com

**Abstract.** We present an approach to identifying the minimal set of potentially relevant Semantic Web data sources for a given query. Our solution involves the adaptation of an efficient information integration algorithm that has polynomial time complexity. We then use these selected sources and an OWL reasoner to answer queries on the Semantic Web. We introduce a concept of source relevance expressed in OWL to reduce the number of sources needed to get the answers to a query. As the Semantic Web is an autonomous entity, some of the data sources may contain data that are not described directly in terms of a given query ontology. In our solution we define and use a mapping language that is a subset of OWL for the purpose of aligning heterogeneous ontologies. Our implemented system supports a subset of SPARQL queries, simple OWL ontologies and data sources that commit to them. Since the time to load sources is a dominating factor in performance, and our system identifies the minimal set of potentially relevant sources, it is very efficient. 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 given work load of 20 domain ontologies, 20 map ontologies and 400 data sources is approximately 1 second. Furthermore, our system returned correct answers to 200 randomly generated queries in three different data configurations.

## 1 Introduction

Regardless of the advances in reasoning techniques and our most sincere efforts in clever coding we will always have to deal with data that are just too big for a given Semantic Web knowledge base system. Fensel and van Harmelen [5] suggest that one way around this problem is to incrementally refine a subset of data sources that will give us an acceptable result. In this work we consider an approach for identifying a sufficient subset in a single shot. 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. 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 . This is reflected in Swoogle's present cache which has 10,000 ontologies but a staggering 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.

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. 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 but relevant problem but this is not the focus of this paper.

We hypothesize that a web like 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.

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 PDMS reformulation algorithm [9]. PDMS uses both Global-As-View (GAV) and Local-As-View (LAV) maps, which are the two most well known information integration formalisms. From a knowledge representation point of view, a GAV rule is essentially equivalent to a Horn clause without function symbols and a LAV rule is a First Order Logic (FOL) implication with a single antecedent and multiple consequents.

Our query language is based on the conjunctive query language for Description Logic (DL) that has been proposed by Horrocks et al. [10]. 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 which we use as our reasoner. KAON2 can only answer the so called DL-safe conjunctive queries [7].

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 two technical contributions in this paper.

1. We present our source selection algorithm that uses OWL maps and REL statements to identify relevant data sources for a given query.
2. 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 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 3, we describe the details of the Ontology Based Information Integrator (OBII), including a source selection algorithm. In Section 4, we describe some experiments that we have conducted to evaluate our system. In Section 5, we compare some related work with our approach and in Section 6 we conclude and discuss future work.

## 2 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 start with Description Horn Logic (DHL) [6], which is contained in the intersection of Description Logic and Horn Logic. More specifically, it is a fragment of OWL corresponding to Horn clauses. As we have observed GAV rules are essentially equivalent to Horn clauses without function symbols. Thus if we were using a GAV information integration system, DHL would be a suitable mapping language.

Although DHL is a very good starting point, the PDMS query reformulation algorithm also supports LAV rules, and as such we can use a language that is more expressive than DHL.

We follow a process similar to Grosof et al. [6] 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 1** *$L_{ac}$ 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 2** *$L_a$ includes all classes in $L_{ac}$. Also, if C and D are classes then $C \sqcup D$ is also a class.*

**Definition 3** *$L_c$ includes all classes in $L_{ac}$. Also, if C and D are classes then $\forall$ R.C is also a class.*

**Definition 4** *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 [6]. Due to limited space, we do not present it here. Please see our technical report [12] for details.

In Section 1, we introduced the notion of a REL statement. If a source can express that it has relevant information, we can choose to query it as opposed to other sources that do not express this information. In this way we can locate the desired information without querying every possible source. Having relevant information, however, does not mean that the source is capable of answering the query completely. It just says that the source may have some useful information on the query.

For example, consider the statement REL(http://sourceURL, Electronics, CinemaDisplay $\sqcap \exists$ madeBy."DELL"). It asserts that http://sourceURL contains some information about cinema displays made by Dell and the data is marked up with the class "Electronics". 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. In the above example the meta:container will be the class expression that defines CinemaDisplay $\sqcap \exists$ madeBy."DELL", the meta:contained will be Electronics and meta:source will be http://sourceURL. The meta:RelStatement will encapsulate these three predicates. REL statements can be translated into LAV rules using a minor variation of the function $\mathcal{T}$ mentioned above. Due to limited space, we can not present the detailed description of the REL statements in this paper. Please see our technical report [12] for details.

## 3 OBII: A Semantic Web Query Answering System

In this section we present our modified algorithm and briefly discuss the implemented architecture of OBII to give the readers 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, and 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 [9]. 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 2), and sources are described using the REL statements and only contain ABox assertions that use named classes and properties.

Given a conjunctive query we unfold it to give us the starting point of our rule goal tree. Now for each atom in the unfolded 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 details. Interested readers are referred to our technical report [12]. 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 [11] 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.

**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 v has not been used then
16:         if GAV(v) and MATCH(n, sp, HEAD(v)) then
17:             for each sub goal ∈ v do
18:                 create an AND child goal node
19:         else if LAV(v) and MATCH(n, sp, b) for some b ∈ BODY(v) then
20:             create an OR child node cgn for the view using MINICON
```

The basic PDMS does not allow cycles in the maps as this makes query answering undecidable. It prevents cycles by never expanding a goal node with a map that was already used on the path from the root to that node in question. Therefore the set of

AND-OR traversals will not be a complete reformulation in the presence of cyclic maps. However, our source selection algorithm only needs to identify the unique set of sources from the leaves of the tree. Since the "cycle check" only prunes subtrees that would be exact copies of other portions of the rule-goal tree, we do not miss any relevant sources, even in the presence of cyclic maps.

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.

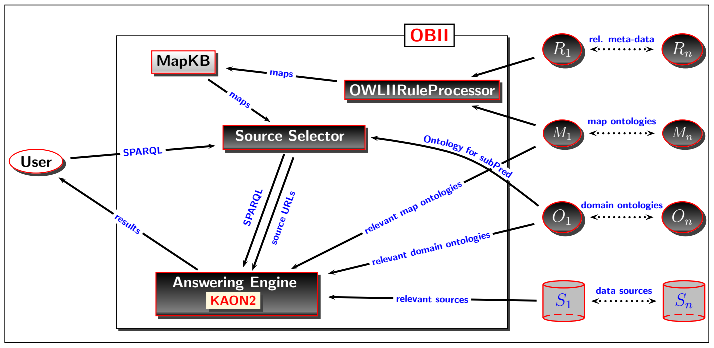


**Fig. 1.** OBII architecture diagram

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.

## 4 Evaluation

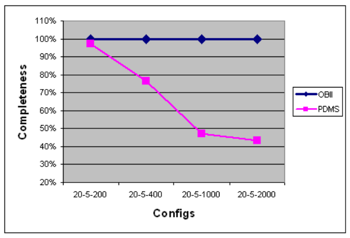
We implemented a workload generator that allows us to control several characteristics of our dataset. In generating the synthetic domain ontologies we decided to have on the average 20 classes and 20 properties (influenced by the dominance of small ontologies in the current Semantic Web). The class and 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. This number is referred to as the diameter by Halevy et al. [9]. We adopt the 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 number of data sources that commits to a given ontology. In the second experiment (herein after referred to as experiment 2) we have varied the diameter. In both experiments we kept the number of ontologies to 20. 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.

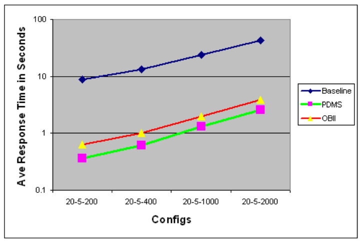
In our experiments the two main metrics we collected and examined are response time and the percentage of complete responses to queries. The response time is the time it takes from the issue of a query to the delivery of its result. We compared three systems in our experiments: A baseline system that 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 twice as slow as the basic PDMS. Note:

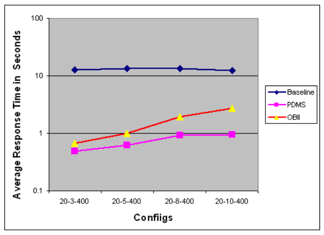


(a) Completeness of OBII and PDMS compared to baseline

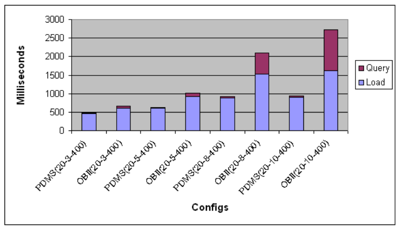


(b) Scalability w.r.t. the increase of sources

**Fig. 2.** The performance of OBII over PDMS and baseline in terms of completeness and scalability.



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



(b) Query to Load time ratio for PDMS and OBII

**Fig. 3.** The scalability of OBII w.r.t. the increase of diameter and the query to load time ratio for PDMS and OBII .

the graphs use a logarithmic scale. However, OBII is 10 times faster than the baseline system, which provides identical functionality.

This time penalty versus the PDMS is essentially unavoidable. In order to get complete answers OBII loads 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 the 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, the response time for OBII is little over 2.5 seconds (as opposed to PDMS's 1 second). Furthermore, PDMS is only 60% complete in that scenario.

## 5 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 [13]. 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 [8] 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 [7] 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 sublanguage 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.

There has been some work done in selecting the appropriate RDF source for a query. One approach is to develop an index of sources [14] and develop a query answering algorithm that exploits these indexes. However, their source description language is limited to RDF where as our OWLII is a much richer language for source description.

## 6 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 10 times faster than a naive approach of reasoning with all sources, and only 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 mapped 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.

## 7 Acknowledgment

This work was funded by Department of Energy via a Tech-X Corporation subcontract. We thank Ameet Chitnis and Fabiana Prabhakar for comments on this paper.

## 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 *International Semantic Web Conference*, 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, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. C. Doshi, and J. Sachs. Swoogle: A Search and Metadata Engine for the Semantic Web. In *Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management*. ACM Press, November 2004.
5. D. Fensel and F. van Harmelen. Unifying reasoning and search to web scale. *IEEE Internet Computing*, 11(2):96, 94–95, 2007.
6. 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.
7. 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.
8. 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*, pages 764–777, 2004.
9. A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proc. of ICDE*, 2003.
10. I. Horrocks and S. Tessaris. A conjunctive query language for description logic aboxes. In *AAAI/IAAI*, pages 399–404, 2000.
11. R. Pottinger and A. Halevy. MiniCon: A scalable algorithm for answering queries using views. *The VLDB Journal*, 10(2-3):182–198, 2001.
12. A. Qasem, D. A. Dimitrov, and J. Heflin. An efficient and complete distributed query answering system for semantic web data. Technical report, Lehigh University, 2007.
13. L. Serafini and A. Tamilin. Distributed instance retrieval in heterogeneous ontologies. In *In Proc. of the Second Italian Semantic Web Workshop Semantic Web Applications and Perspectives (SWAP'05)*, 2005.
14. H. Stuckenschmidt, R. Vdovjak, G.-J. Houben, and J. Broekstra. Index structures and algorithms for querying distributed rdf repositories. In *Proceedings of WWW '04*, pages 631–639, New York, NY, USA, 2004. ACM Press.