Hybrid Query Answering Over Expressive DL Ontologies

Giorgos Stoilos and Giorgos Stamou
School of Electrical and Computer Engineering, NTUA, Greece

1 Introduction

The design of tractable Description Logics, like $\mathcal{EL}$ [1], DL-Lite [2], and $\mathcal{RL}$ [5] have spurred the design and implementation of highly scalable systems such as OWLIM, Oracle’s Semantic Graph, and more. The attractive properties of these systems have in many occasions led application developers to use them even in cases where the input ontology is expressed in the far more expressive SROIQ language. Although, in these cases there can be combinations of inputs for which these systems will fail to compute all certain answers, nevertheless, this notion of completeness is too coarse and these might still be able to compute the correct answers for many input queries.

In the current paper, we report on the following problem: given a query $Q$ containing only distinguished variables (i.e., a SPARQL query) over a SROIQ TBox $T$ and a (scalable) system $\text{ans}$ complete for some fragment $L$ of SROIQ, is it possible to identify in an efficient way if $\text{ans}$ is complete for $Q, T$? We show that this is possible if one has pre-computed the set ($U$) of atomic queries for which $\text{ans}$ is known to be complete. Then, using these techniques we develop an approach to query answering which can decide at run-time whether $Q$ can be evaluated using $\text{ans}$ or a fully-fledged SROIQ reasoner needs to be employed together with $\text{ans}$ to compute the right answers.

We have instantiated our framework using the well-known $\mathcal{RL}$ system OWLIM and the SROIQ reasoner HermiT. Our experimental evaluation has shown that for most test queries over LUBM and UOBM we can correctly determine if OWLIM is (in)complete in less than 50 millisecond, while there are only 3 queries for which our tool replied “unknown”. Finally, our hybrid query answering system was able to compute all answers to the test queries within milliseconds. Compared to previous techniques that attempt to deliver scalable query answering over SROIQ TBoxes by using $\mathcal{RL}$ systems [11, 8], our approach has the advantage that the set $U$ always exists regardless of the expressivity of $T$, hence it is readily applicable to arbitrary SROIQ ontologies. Moreover, the framework is highly modular—that is, it can be instantiated using any combination of systems and not only $\mathcal{RL}$ ones. This is an extended abstract of the paper [9].

2 Checking (In)Completeness of Systems

Consider the TBox $T = \{ A \sqsubseteq \exists S, \exists S \sqsubseteq B, D \sqsubseteq B \}$ and a system $\text{ans}$ that is complete for $\mathcal{RL}$ fragment of $\mathcal{SHOIQ}$ [5]. Then, the behaviour of $\text{ans}$ can be
characterised by the TBox $T_{R\mathcal{L}} = \{ \exists S \sqsubseteq B, D \sqsubseteq B \}$ since $\text{ans}$ discards axioms with existentials in the RHS. Clearly, $\text{ans}$ is complete for the atomic queries $Q_1 = S(x,y)$ and $Q_2 = D(x)$ and, moreover, it is clearly complete for the query $Q = S(x,y) \land D(x)$.

The completeness of $\text{ans}$ w.r.t. $Q, T$ is rather expected given the fact that $Q$ is formed by atoms $S(x,y)$ and $D(x)$ and $\text{ans}$ is complete for queries $Q_1$ and $Q_2$ which correspond to these atoms. This suggests that given a set of atomic queries over which $\text{ans}$ is known to be complete, called query base (QB) $\mathcal{U}$ of $\text{ans}$ for $T$, then we can deduce the completeness of $\text{ans}$ w.r.t. an arbitrary SPARQL query $Q$ by checking if for each of its atoms there is an isomorphic query in $\mathcal{U}$. Actually, as shown next, it suffices for an atom to “appear” in $\mathcal{U}$ or be “entailed” by other atoms of $Q$ that appear in $\mathcal{U}$.

**Theorem 1.** Let $T$ be a SROIQ-TBox, let $\text{ans}$ be a system complete for a DL $\mathcal{L}$ which over $T$ is characterised by the TBox $T_{R\mathcal{L}}$, let $\mathcal{U}$ be a QB of $\text{ans}$ for $T$, and let $Q$ be a SPARQL query. Let also $\mathcal{B}$ be all atoms of $Q$ for which an isomorphic query in $\mathcal{U}$ exists. If each atom $\alpha$ in $Q$ is either in $\mathcal{B}$ or $T_{R\mathcal{L}} \models \land \mathcal{B} \rightarrow \alpha$, then $\text{ans}$ is $(Q, T)$-complete.

However, the previous provides only a sufficient condition for completeness (cf. [9, Example 7]). Unfortunately, to devise both a sufficient and necessary condition one most likely needs to quantify over all minimal “representative” ABoxes $\mathcal{A}$ such that $\text{ans}$ would not compute all answers over $T \cup \mathcal{A}$ and this set can be infinite [3]. Hence, there are (perhaps) strong theoretical limitations in devising such a condition. To alleviate this issue we have also devised a sufficient (easy to check) syntactic condition for concluding incompleteness. This condition is based on the notion of reachability between the symbols of a TBox $T$ [$10, 7$]. Note that $Q$ needs to additionally satisfy a certain consistency Property ($\sharp$) [9].

**Theorem 2.** Let $\mathcal{L}_H$ be a Horn DL and let $\text{ans}$ be a system whose behaviour over $T$ is characterised by $T_{R\mathcal{L}_H}$. Moreover, let $T, \mathcal{U}, \mathcal{B}$ be as in Theorem 1, and let $Q$ be a SPARQL CQ satisfying Property ($\sharp$) [9]. If an atom $\alpha$ of $Q$ is not $\text{Sig}(\mathcal{B})$-reachable in $T_{R\mathcal{L}_H}$, then $\text{ans}$ is $(Q, T)$-incomplete.

Clearly, there can be CQs for which neither of the above theorems hold and hence for which we cannot determine if $\text{ans}$ is complete or not.

**Constructing QBs In Practice** An interesting feature of QBs is that they always exist (a system $\text{ans}$ is either complete or not for an atomic query and for a given TBox $T$ there is only a finite number of them). However, a central issue is how to construct the QB in an easy (i.e., automatic) way. By recent results [3] and developed software$^1$ this is to a large extent possible and, although there are some negative results regarding the techniques in [3, 4], we strongly believe that even in the theoretically problematic cases QBs can be computed effectively (cf. [9, Section 5]).

$^1$https://code.google.com/p/sygenia/
3 Hybrid Query Answering

The straightforward way to exploit our techniques is to use them at query time to decide if a given query $Q$ can be evaluated using some scalable system $\text{ans}$ or we need to resort to a $SROIQ$ reasoner. However, in queries with only distinguished variables we can provide an even more fine-grained approach. More precisely, $Q$ can be split into the part $Q_c$ for which $\text{ans}$ is known to be complete (determined using the previous techniques) and the part $Q_i$ that is not. Then, we can evaluate $Q_c$ using $\text{ans}$, $Q_i$ using a $SROIQ$ reasoner and finally join the results. Even more, $\text{ans}$ can also be used to further improve the performance of the second step as follows: first, the evaluation of $Q_c$ using $\text{ans}$ provides an upper bound of the answers (since the constraints in the $Q_i$ part are missing) and, second, we can also evaluate $Q_i$ using $\text{ans}$ to quickly provide with a lower bound. These two bounds are passed to the $SROIQ$ reasoner in order to restrict the search to only those individuals that are in the upper and not in the lower bound. For the detailed algorithm the reader is referred to [9].

4 Evaluation

We have instantiated our framework using the well-known scalable $\mathcal{RL}$ system OWLlim and the $SROIQ$ reasoner HermiT. The system, called Hydrowl, is available at http://www.image.ece.ntua.gr/~gstoil/hydrowl.

First, at a pre-processing step, we used Hydrowl to compute a QB for OWLlim for ontologies LUBM and UOBM. For LUBM we required 14.5 seconds while for UOBM we required 48.7 seconds (some manual editing was required for UOBM). Second, we used Hydrowl to check if OWLlim is complete for all the test queries of LUBM and UOBM. When all the atoms of a test query appear in the set $B$ (cf. Theorem 1) our tool replies “complete” almost instantaneously (less than 5ms). However, even when for some atom $\alpha$ we tested $T|_{\mathcal{L}} |= \bigwedge B \rightarrow \alpha$, the tool still required less than 50 milliseconds. For LUBM we were able to correctly identify (in)completeness of OWLlim for all except for two queries (Hydrowl replied “unknown”), while for UOBM for all except just one. It is worth noting that for the unknown cases OWLlim is actually incomplete.

Finally, we used Hydrowl to evaluate all test queries of LUBM using 5 universities and of UOBM using 1 department and we have compared against the HermiT-BGP system [6]. Table 1 presents the results (in seconds) for all the interesting queries (for the rest both systems have similar response times). Grey coloured columns mark those queries where Hydrowl uses both OWLlim and HermiT; in all other cases only OWLlim is used. As can be seen, in all queries Hydrowl is considerably faster than HermiT-BGP.

Acknowledgements Giorgos Stoilos was funded by a Marie Curie Career Reintegration Grant within EU’s FP7 under grant agreement 303914. Giorgos Stamos was partially supported by the Greek Strategic Reference Framework 2007-2013 Operational Programme ‘Competitiveness and Entrepreneurship’ under the contract ‘DARIAH-ATTIKH: Developing a Greek Research Infrastructure for the Humanities’. 
Table 1. Query Answering Times

<table>
<thead>
<tr>
<th>Query</th>
<th>LUBM</th>
<th>UOBM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>HermiT-BGP</td>
<td>2.5</td>
<td>1.4</td>
</tr>
<tr>
<td>Hydrowl</td>
<td>.07</td>
<td>.07</td>
</tr>
</tbody>
</table>

References