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

We have instantiated our framework using the well-known $\mathcal{RL}$ system OWLim and the $\mathcal{SROIQ}$ 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 $\mathcal{SROIQ}$ TBoxes by using $\mathcal{RL}$ systems [11, 8], our approach has the advantage that the set $\mathcal{U}$ *always* exists regardless of the expressivity of $\mathcal{T}$, hence it is readily applicable to arbitrary $\mathcal{SROIQ}$ 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 $\mathcal{T} = \{A \sqsubseteq \exists S, \exists S \sqsubseteq B, D \sqsubseteq B\}$ and a system ans that is complete for $\mathcal{RL}$ fragment of $\mathcal{SHOIQ}$ [5]. Then, the behaviour of ans can be characterised by the TBox $\mathcal{T}|_{\mathcal{RL}} = \{\exists S \sqsubseteq B, D \sqsubseteq B\}$ since ans discards axioms with existentials in the RHS. Clearly, ans is complete for the atomic queries $\mathcal{Q}_1 = S(x,y)$ and $\mathcal{Q}_2 = D(x)$ and, moreover, it is clearly complete for the query $\mathcal{Q} = S(x,y) \wedge D(x)$.

The completeness of ans w.r.t. $\mathcal{Q}, \mathcal{T}$ is rather expected given the fact that $\mathcal{Q}$ is formed by atoms $S(x,y)$ and $D(x)$ and ans is complete for queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$ which correspond to these atoms. This suggests that given a set of atomic queries over which ans is known to be complete, called *query base* (QB) $\mathcal{U}$ of ans for $\mathcal{T}$, then we can deduce the completeness of ans w.r.t. an arbitrary SPARQL query $\mathcal{Q}$ 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 $\mathcal{Q}$ that appear in $\mathcal{U}$.

Theorem 1. *Let $\mathcal{T}$ be a $\mathcal{SROIQ}$-TBox, let* ans *be a system complete for a DL $\mathcal{L}$ which over $\mathcal{T}$ is characterised by the TBox $\mathcal{T}|_{\mathcal{L}}$, let $\mathcal{U}$ be a QB of* ans *for $\mathcal{T}$, and let $\mathcal{Q}$ be a SPARQL query. Let also $\mathcal{B}$ be all atoms of $\mathcal{Q}$ for which an isomorphic query in $\mathcal{U}$ exists. If each atom α in $\mathcal{Q}$ is either in $\mathcal{B}$ or $\mathcal{T}|_{\mathcal{L}} \models \bigwedge \mathcal{B} \rightarrow \alpha$, then* ans *is $(\mathcal{Q}, \mathcal{T})$-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 ans would not compute all answers over $\mathcal{T} \cup \mathcal{A}$ 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 $\mathcal{T}$ [10, 7]. Note that $\mathcal{Q}$ needs to additionally satisfy a certain consistency Property ($\sharp$) [9].

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

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

Constructing QBs In Practice An interesting feature of QBs is that they always exist (a system ans is either complete or not for an atomic query and for a given TBox $\mathcal{T}$ 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 $\mathcal{Q}$ can be evaluated using some scalable system ans or we need to resort to a $\mathcal{SROIQ}$ reasoner. However, in queries with only distinguished variables we can provide an even more fine-grained approach. More precisely, $\mathcal{Q}$ can be split into the part $\mathcal{Q}_c$ for which ans is known to be complete (determined using the previous techniques) and the part $\mathcal{Q}_i$ that is not. Then, we can evaluate $\mathcal{Q}_c$ using ans, $\mathcal{Q}_i$ using a $\mathcal{SROIQ}$ reasoner and finally join the results. Evan more, ans can also be used to further improve the performance of the second step as follows: first, the evaluation of $\mathcal{Q}_c$ using ans provides an upper bound of the answers (since the contraints in the $\mathcal{Q}_i$ part are missing) and, second, we can also evaluate $\mathcal{Q}_i$ using ans to quickly provide with a lower bound. These two bounds are passed to the $\mathcal{SROIQ}$ 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 OWLim and the $\mathcal{SROIQ}$ reasoner HermiT. The implemented system is called Hydrowl and 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 OWLim 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 OWLim is complete for all the test queries of LUBM and UOBM. When all the atoms of a test query appear in the set $\mathcal{B}$ (cf. Theorem 1) our tool replies "complete" almost instantaneously (less than 5ms). However, even when for some atom α we tested $\mathcal{T}|_{\mathcal{L}} \models \bigwedge \mathcal{B} \rightarrow \alpha$, the tool still required less than 50 milliseconds. For LUBM we were able to correctly identify (in)completeness of OWLim for all except for two queries (Hydrowl replied "unknown"), wihle for UOBM for all except just one. It is worth noting that for the unknown cases OWLim is actually incomplete.

Fnially, 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 OWLim and HermiT; in all other cases only OWLim 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.

Table 1. Query Answering Times

	LUBM				UOBM					
Query	1	3	8	9	3	4	9	11	12	14
HermiT-BGP	2.5	1.4	1.4	105	204	5.8	21.6	1.7	1.2	48.7
Hydrowl	.07	.07	.24	.13	.02	.01	.01	.07	.04	35.3

References

1. Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 364–369, 2005.
2. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
3. Bernardo Cuenca Grau, Boris Motik, Giorgos Stoilos, and Ian Horrocks. Completeness guarantees for incomplete ontology reasoners: Theory and practice. *Journal of Artificial Intelligence Research*, 43:419–476, 2012.
4. Bernardo Cuenca Grau, Boris Motik, Giorgos Stoilos, and Ian Horrocks. Computing datalog rewritings beyond horn ontologies. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2013.
5. Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logic. In *Proceedings of the Twelfth International World Wide Web Conference (WWW 2003)*, pages 48–57. ACM, 2003.
6. Ilianna Kollia and Birte Glimm. Optimizing sparql query answering over owl ontologies. *Journal of Artificial Intelligence Research (JAIR)*, 48:253–303, 2013.
7. Riku Nortje, Katarina Britz, and Thomas Meyer. Reachability modules for the description logic $\mathcal{SRIQ}$. In *Proceedings of the International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 19)*, 2013.
8. Giorgos Stoilos. Ontology-based data access using rewriting, OWL 2 RL systems and repairing. In *Proceedings of the 11th European Semantic Web Conference (ESWC 2014)*, 2014.
9. Giorgos Stoilos and Giorgos Stamou. Hybrid query answering for owl ontologies. Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 14), 2014. available at: `http://www.image.ece.ntua.gr/~gstoil/temp/main.pdf`.
10. Boontawee Suntisrivaraporn. Module extraction and incremental classification: A pragmatic approach for ontologies. In *Proceedings of European Semantic Web Conference (ESWC 2008)*, pages 230–244, 2008.
11. Yujiao Zhou, Yavor Nenov, Bernardo Cuenca Grau, and Ian Horrocks. Complete query answering over horn ontologies using a triple store. In *Proceedings of the 12th International Semantic Web Conference (ISWC 2013)*, 2013.