Schema-Agnostic Query Rewriting for OWL QL*

Stefan Bischof[1], Markus Krötzsch[2], Axel Polleres[3], and Sebastian Rudolph[2]

[1] Vienna University of Technology, Austria and Siemens AG Österreich, Austria
[2] Technische Universität Dresden, Germany
[3] Vienna University of Economics and Business, Austria

Ontology-based query answering (OBQA) has long been an important topic in applied and foundational research, and in particular in the area of description logics (DLs). Query answering has been studied for every major DL, but the most prominent use of DLs in query answering is based on the DLs of the DL-Lite family. In particular, these have widely been used in ontology-based data access (OBDA), e.g., to integrate disparate data sources or to provide views over legacy databases [3,14,4,9]. DL-Lite$_R$ is also the basis of the W3C OWL 2 Web Ontology Language profile OWL QL, which was specifically designed for OBDA applications [12].

On the other hand, research on query languages has led to a range of expressive features beyond basic pattern matching, e.g., by supporting navigational constructs or other forms of recursion. These developments have also affected practical query languages. SPARQL 1.1, the recent revision of the W3C SPARQL standard, introduces significant extensions to the capabilities of the popular RDF query language [7]. Even at the very core of the query language, we can find many notable new features, including *property paths*, *value creation* (BIND), inline data (VALUES), negation, and extended filtering capabilities. In addition, SPARQL 1.1 now supports query answering over OWL ontologies, taking full advantage of ontological information in the data [6,5]. Thus, with the arrival of SPARQL 1.1, every aspect of OBQA is supported by W3C technologies.

In practice, however, SPARQL and OWL QL are rarely integrated. Most works on OBDA address the problem of answering *conjunctive queries* (CQs), which correspond to SELECT-PROJECT-JOIN queries in SQL, and (to some degree) to Basic Graph Patterns in SPARQL. The most common approach for OBDA is *query rewriting*, where a given CQ is rewritten into a (set of) CQs that fully incorporate the schema information of the ontology. The answers to the rewritten queries (obtained without considering the ontology) are guaranteed to agree with the answers of the original queries (over the ontology). This approach separates the ontology (used for query rewriting) from the rest of the data (used for query answering), and it is typical that the latter is stored in a relational database. Correspondingly, the rewritten queries are often transformed into SQL for query answering. SPARQL and RDF do not play a role in this.

In a recent paper [1], we took a fresh look on the problem of OBQA query rewriting with SPARQL 1.1 as our target query language. The additional expressive power of SPARQL 1.1 allows us to introduce a new paradigm of *schema-agnostic query rewriting*, where the ontological schema is not needed for rewriting queries. Rather, the ontology is stored *together with the data* in a single RDF database. This is how many ontologies are managed today, and it corresponds to the W3C view on OWL and RDF, which does

* This extended abstract summarises the recent results of the authors' paper *Schema-Agnostic Query Rewriting in SPARQL 1.1* [1].

not distinguish schema and data components.[4] The fact that today's OBQA approaches separate both parts testifies to their focus on relational databases. Our work, somewhat ironically, widens the scope of OWL QL to RDF-based applications, which have hitherto focused on OWL RL as their ontology language of choice.

Another practical advantage of schema-agnostic query rewriting is that it supports frequent updates of both data and schema. The rewriting system does not need any information on the content of the database under query, while the SPARQL processor that executes the query does not need any support for OWL. This is particularly interesting if a database can only be accessed through a restricted SPARQL query interface that does not support reasoning. For example, we have used our approach to detect an inconsistency of DBpedia under OWL semantics, using only the public Live DBpedia SPARQL endpoint at http://live.dbpedia.org/sparql (the problem has since been corrected).

The main contributions of our work are:

- We expressed standard reasoning tasks for OWL QL, including consistency checking, classification, and instance retrieval, in *single, fixed* SPARQL 1.1 queries that are independent of the ontology. It turned out that SPARQL 1.1 property paths are powerful enough for OWL QL reasoning.
- We showed how to rewrite arbitrary SPARQL Basic Graph Patterns (BGPs) into single SPARQL 1.1 queries of polynomial size. This task was simplified by the fact that SPARQL does not support "non-distinguished" variables as used in CQs.
- We presented a schema-agnostic rewriting of general CQs in SPARQL 1.1, again into single queries of polynomial size. This rewriting is more involved, and we used two additional features: inline data (VALUES) and (in)equality checks in filters.
- We showed the limits of schema-agnostic rewriting in SPARQL 1.1 by proving that many other OWL features cannot be supported in this way. This includes even the most basic features of OWL EL and OWL RL, and mild extensions of OWL QL. It also is not possible to rewrite regular path queries (and thus basic graph patterns of SPARQL 1.1) into SPARQL 1.1, even for RDFS knowledge bases with assumption of standard use. This would require a more expressive query language, such as *monadically defined queries* [15].

Worst-case reasoning complexity remains the same in all cases, yet our approach is certainly more practical in the case of standard reasoning and BGP rewriting. For general CQs, the rewritten queries are usually too complex for today's RDF databases to handle. Nevertheless, we think that our "SPARQL 1.1 implementation" of OWL QL query answering is a valuable contribution, since it reduces the problem of supporting OWL QL in an RDF database to the task of optimizing a single (type of) query. Since OWL QL subsumes RDFS, one can also apply our insights to implement query answering under RDFS ontologies, which again leads to much simpler queries.

The full details of our rewritings are beyond this abstract, as the length of the rewritten queries – polynomial or not – is too long to include full examples here. However, our basic approach to reasoning with SPARQL 1.1 can be motivated by some very simple

[4] Nevertheless, it is also true that some RDF stores treat terminological triples in special ways, e.g., by keeping them in a dedicated named graph.

observations. Consider a situation where our TBox is guaranteed to contain only axioms of the form $A \sqsubseteq B$ with A and B class names. Such axioms are encoded in RDF using triples of the form A rdfs:subClassOf B. Clearly, one can now query for all (inferred) instances of a class A using the graph pattern

$$\{?X\ (\mathsf{rdf{:}type}\ /\ \mathsf{rdfs{:}subClassOf}^{*})\ A\},$$

which looks for elements ?X that are connected to class A through property rdf:type followed by zero or more uses of property rdfs:subClassOf. The queries used in our work are significantly more involved, since they need to take into account a much larger vocabulary used in OWL, including equivalent classes, subproperties and equivalent properties, inverses, n-ary class intersections, and existential restrictions. Moreover, the queries need to take into account the special semantics of $\top$ and the universal property, as well as the potential inconsistency caused by $\bot$ and the empty property.

An interesting observation here is that syntactic sugar can make schema-agnostic query rewriting more difficult. Clearly, taking into account many possible syntactic encodings must lead to larger queries, which will usually affect execution times. However, the OWL feature owl:SymmetricProperty even makes query rewriting impossible altogether. This is surprising, since property symmetry can easily be expressed using inverses and subproperties, which are fully supported by our approach. Such effects can occur since SPARQL 1.1 is not a universal computing formalism (for its complexity class). Note that these problems vanish if one allows even the most basic kinds of normalisation, but this is not always practical (e.g., when querying the DBpedia SPARQL endpoint).

An interesting side effect of our work is that it provides a simple, worst-case optimal method for terminological reasoning in OWL QL (and thus DL-Lite$_R$). As we treat TBox axioms as data, we can actually formulate queries over terminological knowledge, or even answer conjunctive queries where some class or property names are replaced by variables, with the intended meaning that these "meta-variables" range over vocabulary symbols of the ontology.[5]

Future steps in this line of research include empirical evaluations, where the main challenge is to identify OWL QL benchmarks with non-trivial TBoxes. It should not be assumed that the good theoretical properties of the approach translate directly into good performance, and further optimisations and adjustments might be needed. Implementation techniques such as partial materialisation would lead to a form of *combined rewriting* (cf. [8] for a different approach to combined rewriting). Moreover, it is interesting to extend our work towards more expressive ontology and query languages. On the one hand, one can look towards more expressive DLs, such as $\mathcal{EL}$, which have also been considered in query rewriting [13]. A schema-agnostic approach in this case would resemble Datalog-based reasoning calculi for these logics [10], and indeed one could view Datalog as a query language here. On the other hand, one might consider ontology languages that extend the expressiveness of DL-Lite by using existential rules, e.g., *linear TGDs* [2]. Such cases might actually be somewhat simpler to handle, since one is free to choose a (possibly normalised) database representation of rules, given that there is not standard RDF encoding available.

[5] The term *higher-order query* has been used in this context [11], although true higher-order variables would rather represent arbitrary sets without any relationship to the vocabulary.

Acknowledgements This work has been funded by the Vienna Science and Technology Fund (WWTF, project ICT12-015), and by the DFG in project DIAMOND (Emmy Noether grant KR 4381/1-1).

References

1. Bischof, S., Krötzsch, M., Polleres, A., Rudolph, S.: Schema-agnostic query rewriting in SPARQL 1.1. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C.A., Vrandečić, D., Groth, P.T., Noy, N.F., Janowicz, K., Goble, C.A. (eds.) Proc. 13th Int. Semantic Web Conf. (ISWC'14). LNCS, vol. 8796, pp. 584–600. Springer (2014)
2. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Semantics 14, 57–83 (2012)
3. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. Automated Reasoning 39(3), 385–429 (2007)
4. Di Pinto, F., Lembo, D., Lenzerini, M., Mancini, R., Poggi, A., Rosati, R., Ruzzi, M., Savo, D.F.: Optimizing query rewriting in ontology-based data access. In: Proceedings of the 16th International Conference on Extending Database Technology. pp. 561–572. ACM (2013)
5. Glimm, B., Krötzsch, M.: SPARQL beyond subgraph matching. In: Patel-Schneider, P.F., Pan, Y., Glimm, B., Hitzler, P., Mika, P., Pan, J., Horrocks, I. (eds.) Proc. 9th Int. Semantic Web Conf. (ISWC'10). LNCS, vol. 6496, pp. 241–256. Springer (2010)
6. Glimm, B., Ogbuji, C. (eds.): SPARQL 1.1 Entailment Regimes. W3C Recommendation (21 March 2013), available at http://www.w3.org/TR/sparql11-entailment/
7. Harris, S., Seaborne, A. (eds.): SPARQL 1.1 Query Language. W3C Recommendation (21 March 2013), available at http://www.w3.org/TR/sparql11-query/
8. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: Walsh [16], pp. 2656–2661
9. Kontchakov, R., Rodriguez-Muro, M., Zakharyaschev, M.: Ontology-based data access with databases: A short course. In: Rudolph, S., Gottlob, G., Horrocks, I., van Harmelen, F. (eds.) Reasoning Web, LNCS, vol. 8067, pp. 194–229. Springer, Mannheim, Germany (2013)
10. Krötzsch, M.: Efficient rule-based inferencing for OWL EL. In: Walsh [16], pp. 2668–2673
11. Lenzerini, M., Lepore, L., Poggi, A.: Practical query answering over Hi(DL-Lite$_R$) knowledge bases. In: Bienvenu, M., Ortiz, M., Rosati, R., Simkus, M. (eds.) Proc. 27th Int. Workshop on Description Logics (DL'14). CEUR Workshop Proceedings, vol. 1193, pp. 608–619. CEUR-WS.org (2014)
12. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-profiles/
13. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. J. Applied Logic 8(2), 186–209 (2010)
14. Rodriguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontology-based data access: Ontop of databases. In: Alani, H., Kagal, L., Fokoue, A., Groth, P.T., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N.F., Welty, C., Janowicz, K. (eds.) Proc. 12th Int. Semantic Web Conf. (ISWC'13). LNCS, vol. 8218, pp. 558–573. Springer (2013)
15. Rudolph, S., Krötzsch, M.: Flag & check: Data access with monadically defined queries. In: Hull, R., Fan, W. (eds.) Proc. 32nd Symposium on Principles of Database Systems (PODS'13). pp. 151–162. ACM (2013)
16. Walsh, T. (ed.): Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11). AAAI Press/IJCAI (2011)