Query Rewriting in Horn-$\mathcal{SHIQ}$ (Extended Abstract)

Despoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos Stamou

School of Electrical and Computer Engineering,
National Technical University of Athens, Greece

1 Introduction

Resolution is an important and attractive tool for rewriting a Description Logic ontology into (disjunctive) datalog for the purposes of query answering [3, 5, 6]. This is because there are already many general purpose resolution calculi that can support deduction in much more expressive logics and which have well-defined powerful redundancy elimination criteria like clause subsumption. However, the generality of these procedures implies that the characteristics and structure of DL axioms are not fully exploited during saturation and as a consequence the designed rewriting algorithms often suffer from performance issues [7, 4]. More precisely, resolution algorithms like those designed in [3, 5, 6] usually produce far too many clauses that contain function terms, many of which are never used to derive other function-free clauses that are members of the ontology rewriting. For example, a typical algorithm will always resolve clauses $A(x) \leftarrow R(x, y) \wedge B(y)$ and $R(x, f(x)) \leftarrow C(x)$ to produce $A(x) \leftarrow C(x) \wedge B(f(x))$ even if the function term $f(x)$ cannot be subsequently eliminated.

In our previous work we have defined a novel resolution-based rewriting algorithm for DL-Lite and $\mathcal{ELHI}$ that largely avoids the generation of such redundant clauses [8]. The algorithm uses a macro-inference rule, called *n-shrinking*, which searches for sets of clauses such that when these are used as side-premises in consecutive resolutions, intermediate clauses with function terms can be produced but the final resolvent must be function-free. For example, in the previous scenario, the algorithm will consider resolving the two clauses to produce $A(x) \leftarrow C(x) \wedge B(f(x))$ only if a clause of the form $B(f(x)) \leftarrow C(x)$ also exists in the working set of clauses and hence the function-free clause $A(x) \leftarrow C(x)$ can finally be obtained. In order to reduce the size of the set where such pairs of side premises are looked up, in contrast to previous approaches, the algorithm does not "propagate" function symbols. For example, all algorithms in [3, 5, 6] will resolve clauses $S(x, f(x)) \leftarrow C(x)$ and $R(x, y) \leftarrow S(x, y)$ to produce $R(x, f(x)) \leftarrow C(x)$. In contrast, the algorithm in [8] will first try to *unfold* the clause $R(x, y) \leftarrow S(x, y)$ on some clause of the form $A(x) \leftarrow S(x, y) \wedge B(y)$ to produce $A(x) \leftarrow R(x, y) \wedge B(y)$ and then consider n-shrinking on that clause.

We have considerably extended our previous work and designed a datalog rewriting algorithm for Horn-$\mathcal{SHIQ}$ ontologies that follows the same principles.

Our first extension is to modify the unfolding and n-shrinking rules to be applicable on clauses with equality that are a distinctive feature of Horn-$\mathcal{SHIQ}$.

Example 1. Consider an ontology consisting of the following axioms given also in clausal form:

$$A \sqsubseteq \leq 1R.B \rightsquigarrow y \approx z \leftarrow A(x) \wedge R(x,y) \wedge R(x,z) \wedge B(y) \wedge B(z) \quad (1)$$

$$D \sqsubseteq \exists R.\top \rightsquigarrow R(x, g(x)) \leftarrow D(x) \quad (2)$$

$$C \sqsubseteq \exists S.B \rightsquigarrow S(x, f(x)) \leftarrow C(x), \quad B(f(x)) \leftarrow C(x) \quad (3)$$

$$S \sqsubseteq R \rightsquigarrow R(x,y) \leftarrow S(x,y) \quad (4)$$

Unfolding between (1) and (4) produces $y \approx z \leftarrow A(x) \wedge S(x,y) \wedge R(x,z) \wedge B(y) \wedge B(z)$ on which shrinking with premises clauses (3)(a) and (3)(b) produces $f(x) \approx z \leftarrow A(x) \wedge C(x) \wedge R(x,z) \wedge B(z)$. Notice how shrinking prevents resolving clause (1) with clause (2) which would construct a redundant resolvent. $\diamondsuit$

The biggest challenge in the new algorithm is to deal with equality reasoning. Like in [3] we employ superposition, however, following the ideas set in [8] we try to restrict its application in such a way that unnecessary intermediate clauses are constructed only when necessary. First, due to n-shrinking, only equality clauses with function-free bodies can appear in the working set (see also previous example). Hence, superposition inferences can be restricted to have only such clauses as side premises which significantly reduces the search space. However, superposition inferences can introduce new function terms in the body of a clause. Consider for example clauses $f(g(x')) \approx x' \leftarrow B(x')$ and $R(x, f(x)) \leftarrow A(x)$. The first clause can be superposed into the second with unifier $x \mapsto g(x')$ to obtain the resolvent $R(g(x'), x') \leftarrow A(g(x')) \wedge B(x')$. Now, first, note that by the *basic* strategy of superposition [1] (superposition remains complete if it is only applied into function terms not introduced by previous unification steps) no subsequent superpositions need to be applied on the body of clause $R(g(x'), x') \leftarrow A(g(x')) \wedge B(x')$. Second, according to the ideas in [8] and our previous discussion this intermediate clause is of importance only if $g(x')$ can be eliminated from the body. These two observations combined imply that we can devise the following macro-superposition inferences:

$$\frac{R(x, f(x)) \leftarrow A(x) \quad f(g(x)) \approx x \leftarrow B(x), A(g(x)) \leftarrow C(x)}{R(g(x), x) \leftarrow C(x) \wedge B(x)}$$

$$\frac{B(f(x)) \leftarrow A(x) \quad f(g(x)) \approx x \leftarrow B(x), A(g(x)) \leftarrow C(x)}{B(x) \leftarrow C(x) \wedge B(x)}$$

where $f(x)$ has not been introduced by a previous unification.

A detailed description and definition of the algorithm can be found at `http://www.image.ece.ntua.gr/~despoina/document.pdf`.

2 Evaluation

We have implemented our rewriting algorithm into our prototype system Rapid.[1] We conducted an experimental evaluation and compared it against Clipper [2], to the best of our knowledge, the only available conjunctive query rewriting system for Horn-$\mathcal{SHIQ}$ ontologies. Our test suite included Horn-$\mathcal{SHIQ}$ fragments of the ontologies NASA SWEET 2.3, Periodic, and DOLCE2.1Lite-Plus. We also used the UOBM ontology that is provided in Clipper's test suite. For the UOBM ontology we used the 10 queries that come together with Clipper, while for the rest we manually constructed 5 test queries. All tests were performed on a 2,26GHz Intel Core 2 Duo laptop running OS X 10.9.5 and JVM 1.7. We set a timeout to 2 hours. Table 1 indicates the results. As can be seen regarding UOBM (left sub-table) Rapid is consistently faster than Clipper. The sizes of the computed rewritings are roughly the same and differences are attributed to the different structures of the computed datalog rewritings. Regarding the much larger and relatively real-world ontologies (right sub-table) Clipper failed to terminate within the set time limit whereas Rapid requires at most a few seconds.

Table 1: Evaluation results

(a)

UOBM (207 axioms)			
t (ms)		Rew. size	
Rapid	Clipper	Rapid	Clipper
11	64	3	2
18	56	14	16
65	1415	109	920
25	59	22	45
18	67	16	33
20	117	13	16
21	65	14	15
14	61	10	25
28	55	31	33
23	61	23	23

(b)

$\mathcal{O}$	Axioms	t (ms)	Rew. size
NASA SWEET	11578	56	170
		142	399
		177	551
		414	979
		348	989
Periodic_full	43689	29	23
		1768	533
		1336	631
		129	195
		605	714
DOLCE2.1	1055	1314	1192
		1230	1194
		35	85
		163	205
		1379	1192

Acknowledgements Giorgos Stoilos was funded by a Marie Curie Career Reintegration Grant within EU's FP7 under grant agreement 303914.

[1] `http://www.image.ece.ntua.gr/~achort/rapid/`

References

1. Leo Bachmair, Harald Ganzinger, Christopher Lynch, and Wayne Snyder. Basic paramodulation. *Information and computation*, 121(2):172–192, 1995.
2. Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran, and Guohui Xiao. Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules. In *Proc. of AAAI 2012*, 2012.
3. Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reasoning in description logics by a reduction to disjunctive datalog. *J. of Autom. Reas.*, 39(3):351–384, 2007.
4. Martha Imprialou, Giorgos Stoilos, and Bernardo Cuenca Grau. Benchmarking ontology-based query rewriting systems. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI 2012)*, 2012.
5. Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks. A Comparison of Query Rewriting Techniques for DL-lite. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors, *Proc. of the 22nd Int. Workshop on Description Logics (DL 2009)*, 2009.
6. Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks. Tractable query answering and rewriting under description logic constraints. *Journal of Applied Logic*, 8(2):186–209, 2010.
7. Frantisek Simancik, Yevgeny Kazakov, and Ian Horrocks. Consequence-based reasoning beyond horn ontologies. In *Proc. of IJCAI 2011*, pages 1093–1098, 2011.
8. Despoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos Stamou. Optimising resolution-based rewriting algorithms for owl ontologies. *Journal of Web Semantics*, Accepted, In press, 2015.