OBDA Using $\mathcal{RL}$ Reasoners and Repairing*

Giorgos Stoilos

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

1 Introduction

Rewriting the input TBox $\mathcal{T}$ (and query $\mathcal{Q}$) into a datalog program, called *$\mathcal{T}$-rewriting* (*$(\mathcal{Q},\mathcal{T})$-rewriting*), is a prominent approach to ontology-based data access [1]. It was used as early as the KAON2 system [5] and it currently consists of (perhaps) the standard approach to answering queries over ontologies expressed in the languages DL-Lite [2, 12] and $\mathcal{ELHI}$ [8, 16].

Apart from computing a rewriting, a problem that has been proven very difficult is how to subsequently (effectively) evaluate it over the data. In many cases, due to its size and/or complexity, additional techniques need to be devised [11, 9, 10]. Unfortunately, all of them have so far been applied only on DL-Lite while for more expressive languages, apart from the work in [3], to the best of our knowledge, no other integrated query answering system has been reported or evaluated with large and complex ontologies.

An interesting observation is that $\mathcal{T}$-rewritings are usually of a particular form—that is, they can be translated back into an $\mathcal{RL}$ TBox [14].[1] Hence, scalable $\mathcal{RL}$ system, like OWLim, can be used to evaluate them over the data. More precisely, it was shown how, given a $\mathcal{SROIQ}$ TBox $\mathcal{T}$ and an $\mathcal{RL}$ system ans, a rewriting of $\mathcal{T}$ can be used to compute a set of axioms $\mathcal{R}$ (called *repair*) such that for *every* CQ $\mathcal{Q}$ with only distinguished variables (i.e., SPARQL CQs) and *every* ABox $\mathcal{A}$ we have $\mathsf{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A}) \subseteq \mathsf{ans}(\mathcal{Q}, \mathcal{T} \cup \mathcal{R} \cup \mathcal{A})$. That is, after repairing, ans is indistinguishable from a $\mathcal{SROIQ}$ reasoner w.r.t. SPARQL CQs.

Although, the experimental results in [14] provided with encouraging results, these were quite preliminary. First, the implementation used an arguably obsolete system (Requiem) and had no optimisations. Second, the evaluation was based on LUBM (an arguably trivial TBox) and a small fragment of Galen. Hence, it was unclear whether repairing can be applied to large and complex TBoxes. Third, the approach could only handle SPARQL CQs.

In the current paper we attempt to extensively study repairing as an approach to OBDA over expressive TBoxes. First, we propose several optimisations and refinements to the first prototype in order to handle large and complex TBoxes. Second, we show how arbitrary queries can be supported. Third, we perform an extensive experimental evaluation which showed that we can handle very large and complex TBoxes (e.g., Galen, GO, Fly) and answer arbitrary CQs over one of them (Fly) within milliseconds. This is an extended abstract of paper [13].

* Funded by an EU FP7 Marie Curie Career Reintegration Grant (No.: 303914).

[1] In OWL and Semantic Web jargon that is an OWL 2 RL TBox [4].

2 Computing Repairs Efficiently

Let $\mathcal{T}$ be a TBox and let ans be an $\mathcal{RL}$ system. Roughly speaking, a repair $\mathcal{R}$ of ans for $\mathcal{T}$ is computed by first computing a $\mathcal{T}$-rewriting (Rew) for $\mathcal{T}$, then removing from Rew the parts that ans can already handle, and finally, minimising the resulting set. For example, if $\mathcal{T} = \{A \sqsubseteq \exists R, \exists R \sqsubseteq B\}$, then $\mathsf{Rew} = \{A(x) \rightarrow B(x), R(x,y) \rightarrow B(x)\}$ is a $\mathcal{T}$-rewriting of $\mathcal{T}$, while $\mathcal{R} = \{A \sqsubseteq B\}$ is the desirable repair for ans. We call $\textsc{Repair}(\mathcal{T})$ the overall procedure.

Compared to [14] we have performed two important refinements. First, according to [14] a repair $\mathcal{R}$ can only be computed by $\mathcal{T}$-rewritings Rew such that $\mathcal{T} \models \mathsf{Rew}$. This is quite restrictive as it prohibits the use of many efficient rewriting algorithms which normalise the input TBox $\mathcal{T}$ into $\mathcal{T}'$ by introducing fresh symbols. To be able to use such approaches to compute repairs we notice that, for $\mathcal{T}'$ the normalised TBox used internally by the rewriting system to compute Rew, we have $\mathsf{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A}) \subseteq \mathsf{ans}(\mathcal{Q}, \mathcal{T}' \cup \textsc{Repair}(\mathcal{T}') \cup \mathcal{A})$. Second, the minimisations steps applied by $\textsc{Repair}$ are based on expensive FOR-loops over the initial $\mathcal{T}$-rewriting in which HermiT is invoked. Despite how optimised a $\mathcal{SROIQ}$ reasoner is, in large ontologies, these entailment checks could simply be too many. To improve the behaviour we have interveaned in the internals of HermiT to implement a form of incremental entailment checking. First, we exhaustively apply the calculus over $\mathcal{T}$ to construct an initial model and we mark the completion of the execution. Then, to check $\mathcal{T} \models \alpha$ we resume the execution from the previous point and after completion we backtrack to the marked point.

3 Supporting non-SPARQL Queries

Let $\mathcal{Q}$ be an arbitrary CQ with query predicate Q, let $\mathcal{T}$ be a TBox, and let $\mathsf{Rew}_D \uplus \mathsf{Rew}_Q$ be a $(\mathcal{Q}, \mathcal{T})$-rewriting ($\mathsf{Rew}_D$ is a datalog program not mentioning Q and Rew_Q a UCQ that mentions Q). Since Rew_D does not mention Q it captures only ground entialments of $\mathcal{T}$ over some $\mathcal{A}$. But, after repairing, ans is complete w.r.t. *all* ground entailments for any $\mathcal{A}$; hence $\mathsf{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A}) \subseteq \mathsf{ans}(\mathsf{Rew}_Q, \mathcal{T} \cup \textsc{Repair}(\mathcal{T}) \cup \mathcal{A})$. The following summarises our approach:

1. Compute a repair $\mathcal{R}$ of $\mathcal{T}$ for ans using procedure $\textsc{Repair}$.
2. Load the dataset $\mathcal{A}$, the input TBox $\mathcal{T}$, and the repair $\mathcal{R}$ to ans.
3. For a CQ $\mathcal{Q}$, if $\mathcal{Q}$ is SPARQL then evaluated it over ans; otherwise compute a $(\mathcal{Q}, \mathcal{T})$-rewriting $\mathsf{Rew}_D \uplus \mathsf{Rew}_Q$ and evaluate Rew_Q over ans.

Note that most steps, i.e., steps 1 and 2, can be done only once as a pre-processing (changes in $\mathcal{A}$ can be handled incrementally).

4 Implementation and Evaluation

We have implemented a prototype ontology repair and query answering tool called Hydrowl.[2] The tool uses Rapid [15] to compute $\mathcal{T}$-rewritings and HermiT [7] and OWLim [6] to minimise it into the final repair.

[2] `http://www.image.ece.ntua.gr/~gstoil/hydrowl/`

Table 1. $|\mathcal{T}|$ ($|\mathcal{R}|$): number of axioms of the input TBox (repair), t: time in seconds.

$\mathcal{T}$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{R}\rvert$	t	$\mathcal{T}$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{R}\rvert$	t
Not-Galen	5471	3015(4153)	298(42)	Galen-doc	4229	6051(6176)	1152(28)
Fly	19845	10361(12368)	2884(178)	Galen	4229	3012(3062)	257(24)

Table 2. Loading times for Fly and UOBM for the various ABoxes.

	Universities				
	1	2	5	10	20
UOBM	4.1	6.8	16.2	31.9	73.2
UOBM$\cup\mathcal{R}$	4.4	8.3	24.3	44.9	108.1

(a) UOBM

	$\mathcal{A}$	$2\times\mathcal{A}$	$3\times\mathcal{A}$	$4\times\mathcal{A}$	$5\times\mathcal{A}$
Fly	14.0	21.9	22.7	27.9	31.5
Fly$\cup\mathcal{R}$	31.9	55.1	68.5	93.0	119.3
Fly$\cup\mathcal{R}^-$	33.2	62.1	70.1	100.6	118.2

(b) Fly

Using Hydrowl we managed to compute repairs for 151 out of the 152 ontologies of our dataset, which contained some large and highly complex ones. In the vast majority of cases a repair could be computed in less than a second, only a handfull required up to a few minutes, and only the very large ones several minutes; Table 1 presents results for the latter. Despite their size and complexity we see that we can compute repairs for them in a reasonable amount of time (recall this is usually done only once). Actually, if we discard a very expensive minimisation step, then we can compute some (non-minimal) repair very efficiently while its size is not considerably larger than the minimal one (see Table 1 numbers in parenthesis). The system in [14] could not handle any of these ontologies.

Next, Table 2 presents loading experiments to OWLim using UOBM (1 to 20 universities) and Fly (ABox multiplied up to 5 times). As can be seen, the overhead introduced by repairing is significantly noticeable only in Fly, mostly due to the size of the computed repair ($\mathcal{R}$), however, recall that loading is usually performed once. In Fly we have also loaded the non-minimal repair ($\mathcal{R}^-$) and as it turns out there is no significant difference compared to the minimal one.

Table 3. Results for Answering the Fly Queries

$\mathcal{Q}_1$			$\mathcal{Q}_2$			$\mathcal{Q}_4$			$\mathcal{Q}_5$		
$\lvert\mathsf{Rew}_Q\rvert$	t_{rew}	t_{ans}	$\lvert\mathsf{Rew}_Q\rvert$	t_{rew}	t_{ans}	$\lvert\mathsf{Rew}_Q\rvert$	t_{rew}	t_{ans}	$\lvert\mathsf{Rew}_Q\rvert$	t_{rew}	t_{ans}
64	0.31	0.31	2880	0.90	1.28	91	0.07	0.04	6	0.05	0.02

The Fly ontology comes with 4 non-SPARQL queries. Table 3 presents the results of our technique from Section 3. As we can see in most cases we were able to compute and evaluate a rewriting almost instantaneously. This greatly outperformes previous approaches on Fly [17, 18] which require several seconds *per* query. The good behaviour of our approach can be attributed to the fact that most hard work is pushed to a pre-processing step while Rew_Q, the only thing computed on-line, is usually small and of simple structure.

References

1. Diego Calvanese De Giacomo Giuseppe Antonella Poggi, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *Journal on Data Semantics*, X:133–173, 2008.
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. *Journal of Automated Reasoning*, 39(3):385–429, 2007.
3. 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.
4. 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.
5. Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Deciding expressive description logics in the framework of resolution. *Information & Computation*, 206(5):579–601, 2008.
6. Atanas Kiryakov, Barry Bishoa, Damyan Ognyanoff, Ivan Peikov, Zdravko Tashev, and Ruslan Velkov. The Features of BigOWLIM that Enabled the BBCs World Cup Website. In *Workshop on Semantic Data Management (SemData)*, 2010.
7. Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research*, 36:165–228, 2009.
8. 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.
9. Mariano Rodriguez-Muro and Diego Calvanese. High performance query answering over DL-Lite ontologies. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2012)*, 2012.
10. Mariano Rodríguez-Muro, Roman Kontchakov, and Michael Zakharyaschev. Ontology-based data access: Ontop of databases. In *Proceedings of the International Semantic Web Conference (ISWC 2013)*, pages 558–573, 2013.
11. Riccardo Rosati. Prexto: Query rewriting under extensional constraints in dl-lite. In *9th Extended Semantic Web Conference (ESWC 2012)*, pages 360–374, 2012.
12. Riccardo Rosati and Alessandro Almatelli. Improving query answering over DL-Lite ontologies. In *Proceedings of the 12th International Conference on the Principles of Knowledge Representation and Reasoning (KR-10)*, 2010.
13. 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.
14. Giorgos Stoilos, Bernardo Cuenca Grau, Boris Motik, and Ian Horrocks. Repairing ontologies for incomplete reasoners. In *Proceedings of the 10th International Semantic Web Conference (ISWC-11), Bonn, Germany*, pages 681–696, 2011.
15. Depoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos Stamou. Optimising resolution-based rewriting algorithms for DL ontologies. In *Proceedings of the 26th Workshop on Description Logics (DL 2013)*, 2013.
16. Despoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos Stamou. Optimising resolution-based rewriting algorithms for dl ontologies. In *Proceedings of the 26th Workshop on Description Logics (DL 2013), Ulm, Germany*, 2013.

17. Yujiao Zhou, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, and Jay Banerjee. Making the most of your triple store: Query answering in OWL 2 using an RL reasoner. In *Proc, WWW 2013*, pages 1569–1580, 2013.
18. Yujiao Zhou, Yavor Nenov, Bernardo Cuenca Grau, and Ian Horrocks. Complete query answering over horn ontologies using a triple store. In *Proc. of the 12th International Semantic Web Conference (ISWC)*. Springer LNCS, 2013.