Efficient Query Answering in DL-Lite through FOL Reformulation (Extended Abstract)

Damian Bursztyn, François Goasdoué, Ioana Manolescu

Efficient Query Answering in DL-Lite through FOL Reformulation (Extended Abstract)

Damian Bursztyn[1], François Goasdoué[2] and Ioana Manolescu[1]

[1] INRIA & U. Paris-Sud, France   [2] U. Rennes 1 & INRIA, France

Abstract. We propose a general query optimization framework for formalisms enjoying FOL reducibility of query answering, for which it reduces to the evaluation of a FOL query against facts. This framework allows searching within a set of alternative equivalent FOL queries, i.e., FOL reformulations, one with minimal evaluation cost when evaluated through a relational database management system. We provide two algorithms, an exhaustive and a greedy, for exploring the optimization space. This framework is applied to the lightweight description logic DL-Lite$_\mathcal{R}$ underpinning the W3C's OWL2 QL profile, for which an experimental evaluation validates the interest and applicability of our technique.

1 Introduction

Query answering in the lightweight DL-Lite$_\mathcal{R}$ description logic [1] has received significant attention in the literature, as it provides the foundations of the W3C's OWL2 QL standard for Semantic Web applications. In particular, query answering techniques based on FOL reducibility, e.g., [1,2,5,6,7], which reduce query answering against a knowledge base (KB) to FOL query evaluation against the KB's facts only (a.k.a. ABox) by compiling the KB's domain knowledge (a.k.a. TBox) into the query, hold great potential for performance. This is because FOL queries can be evaluated by a highly optimized Relational Database Management System (RDBMS) storing the KB's facts.

The goal of our study is to identify efficient techniques for query answering in description logics enjoying FOL reducibility, with a focus on DL-Lite$_\mathcal{R}$. Notably, we reduce query answering to the evaluation of alternative FOL queries, a.k.a. FOL *reformulations*, belonging to richer languages than those considered so far in the literature; in particular, this may allow *several* (equivalent) FOL reformulations for a given input query. This contrasts with related works, e.g., the aforementioned ones, which aim at a *single* FOL reformulation (modulo minimization). Allowing a variety of reformulations is crucial for efficiency, as such alternatives, while computing the same answers, may have very different performance (response time) when evaluated through an RDBMS. Therefore, instead of having a single fixed choice that may or may not be performant, we select the one with lowest estimated evaluation cost among possible alternatives.

2 Cover-based query answering optimization

RDBMS query optimizers consider a set of *evaluation alternatives* (a.k.a. logical and physical plans), and select the one minimizing a *cost estimation function.*

Since the number of alternatives is in $\mathcal{O}(2^n \times n!)$ for a conjunctive query (`CQ`) of n atoms [4], modern optimizers rely on heuristics to explore only a few alternatives; this works (very) well for *small-to-moderate size* `CQ`s. However, `FOL` reformulations go *beyond* `CQ`s in general, and may be *extremely large*, leading the RDBMS to perform poorly.

To work around this limitation, we introduce the cover-based query answering technique to define a space of equivalent `FOL` reformulations of a `CQ`. A *cover* defines how the query is split into subqueries, that may overlap, called *fragment queries*, such that substituting each subquery with its `FOL` reformulation (obtained from any state-of-the-art technique) and joining the corresponding (reformulated) subqueries, *may* yield a `FOL` reformulation for the query to answer. Not every cover of a query leads to a `FOL` reformulation; but every cover which does, yields an alternative *cover-based* `FOL` *reformulation* of the original query. Crucially for our problem, *a smart cover choice may lead to a cover-based reformulation whose evaluation is more efficient.* Thus, the cover-based technique amounts to *circumventing* the difficulty of modern RDBMSs to efficiently evaluate `FOL` reformulations in general.

Problem 1 (Optimization problem). Given a `CQ` q and a description logic KB $\mathcal{K}$, the *cost-driven cover-based query answering problem* consists of finding a cover-based reformulation of q based on $\mathcal{K}$ with lowest (estimated) evaluation cost.

We solve this problem for DL-Lite$_\mathcal{R}$ in two steps. First, we provide a sufficient condition for a cover to be *safe for query answering*, i.e., to lead to a cover-based `FOL` reformulation. The main idea for this condition is to have a cautious approximation of the *query atoms* which are interdependent w.r.t. reformulation, i.e., which (directly or after specialization) unify through state-of-the-art reformulation techniques, and keep them in the same cover fragment. The space of all covers of a query q satisfying this condition is denoted $\mathcal{L}_q$; all $\mathcal{L}_q$ covers turn out to correspond to some fusion of fragments from a certain *root cover* we denote C_{root}. We also refine our sufficient condition to identify an extended space of covers $\mathcal{E}_q$, which includes $\mathcal{L}_q$ and also leads to `FOL` reformulations of q.

Second, based on a function ϵ *estimating the evaluation cost of a given* `FOL` *query through an RDBMS*, we devise two cover search algorithms. The first one, termed **EC-DL (Exhaustive Covers)**, starts from C_{root} and explores all $\mathcal{E}_q$ covers in the case of DL-Lite$_\mathcal{R}$. The second one, named **GC-DL (Greedy Covers)**, also starts from C_{root} but explores $\mathcal{E}_q$ partially, in greedy fashion. It uses an *explored cover set* initialized with $\{C_{\text{root}}\}$, from which it picks a cover C inducing a $q^{\texttt{FOL}}$ reformulation with minimum cost $\epsilon(C)$, and attempts to build from C a cover C', by fusing two fragments, or adding (copying) an atom to a fragment. GC-DL only adds C' to the explored set if $\epsilon(C') < \epsilon(C)$, thus it only explores a small part of the search space. Both algorithms return a cover-based reformulation with the minimum estimated cost w.r.t. the explored space. When fusing two fragments into one, or adding an atom to a fragment, $\epsilon(C)$ decreases if the new fragment is more selective than the fragment(s) it replaces. Therefore, the RDBMS may find a more efficient way to evaluate the query of this new fragment, and/or its result may be smaller, making the evaluation of $q^{\texttt{FOL}}$ based on the new cover C' faster.

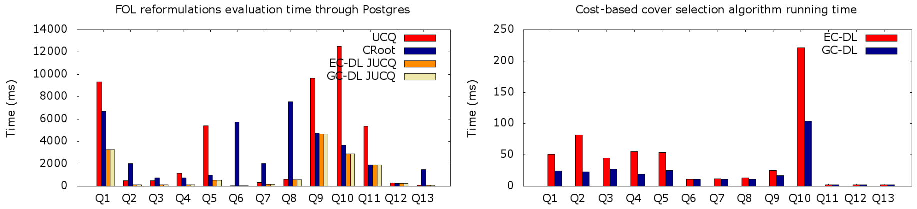

Fig. 1: (a) Evaluation time for FOL reformulations. (b) Cover search running time.

3 Experimental validation

We implemented our cover-based approach in Java 7, on top of PostgreSQL v9.3.2. We used the LUBM$^{\exists}_{20}$ DL-Lite$_{\mathcal{R}}$ TBox and associated EUDG data generator [3]: LUBM$^{\exists}_{20}$ consists of 34 roles, 128 concepts and 212 constraints; the generated ABox comprises 15 million facts. We chose RAPID [2] for CQ-to-UCQ (unions of CQs) reformulation. For ϵ, we used Postgres' own estimation, obtained using the explain directive. We devised a set of 13 CQs, ranging from 2 to 10 atoms (5.77 on average); their UCQ reformulations have 35 to 667 CQs (290.2 on average).

Figure 1(a) depicts the evaluation time through Postgres, of four FOL reformulations: (*i*) the UCQ produced by RAPID [2]; (*ii*) the JUCQ (joins of UCQs) reformulation based on C_{root}; (*iii*) the JUCQ reformulation corresponding to the best-performing cover found by our algorithm EC-DL, and (*iv*) the JUCQ reformulation based on the best-performing cover found by GC-DL. First, the figure shows that fixed FOL reformulations are not efficiently evaluated, e.g., UCQ for Q_1, Q_5 and Q_9-Q_{11}, and the one based on C_{root} for Q_6-Q_8 and Q_{13}. This poor performance correlates with the large size of the UCQ reformulations: such very large unions of CQs are very poorly handled by current RDBMS optimizers, which are designed and tuned for small CQs. Second, the reformulation based on the cover returned by EC-DL is always more efficient than UCQ reformulation (more than one order of magnitude for Q_5), respectively, C_{root}-based reformulation (up to a factor of 230 for Q_6). Third, in our experiments, the GC-DL-chosen cover leads to a JUCQ reformulation as efficient as the EC-DL one, demonstrating that even a partial, greedy cover search leads to good performance (this cannot be guaranteed in general). For Q_7 and Q_9-Q_{13}, the best cover we found is safe; for all the others, this is not the case, confirming the interest of the larger space $\mathcal{E}_q$.

Figure 1(b) depicts the running time of the EC-DL and GC-DL algorithms, which can be seen as the overhead of our cover-based technique. The time is very small, between 2 ms (Q_{11}-Q_{13}, with just 2 atoms) and 221 ms (EC-DL on Q_{10}, of 10 atoms). The time is higher for more complex queries, but these are precisely the cases where our techniques are most beneffician, e.g., for Q_{10}, EC-DL runs in less than 2% of the time to evaluate the UCQ reformulation, while the cover we recommend is more than 4 times faster than UCQ. As expected, GC-DL is faster than EC-DL due to the exploration of less covers. Together, Figure 1(a) and 1(b) confirm the benefits and practical interest of our cost-based cover search.

Acknowledgements This work has been partially funded by the *Programme Investissement d'Avenir* Datalyse project and the ANR PAGODA project.

References

1. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. JAR 39(3), 385–429 (2007)
2. Chortaras, A., Trivela, D., Stamou, G.B.: Optimized query rewriting for OWL 2 QL. In: CADE (2011)
3. Lutz, C., Seylan, I., Toman, D., Wolter, F.: The combined approach to OBDA: taming role hierarchies using filters. In: ISWC. pp. 314–330 (2013)
4. Ono, K., Lohman, G.M.: Measuring the complexity of join enumeration in query optimization. In: VLDB (1990)
5. Pérez-Urbina, H., Horrocks, I., Motik, B.: Efficient query answering for OWL 2. In: ISWC (2009)
6. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: KR (2010)
7. Venetis, T., Stoilos, G., Stamou, G.B.: Incremental query rewriting for OWL 2 QL. In: Description Logics (2012)