Web Semantics: Science, Services and Agents on the World Wide Web 33 (2015) 94–111

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Complexity of answering counting aggregate queries over *DL-Lite*

Egor V. Kostylev [a,b], Juan L. Reutter [c,*]

[a] *University of Oxford, United Kingdom*
[b] *University of Edinburgh, United Kingdom*
[c] *Pontificia Universidad Católica de Chile, Chile*

ARTICLE INFO

Article history:
Received 30 September 2013
Received in revised form
21 April 2015
Accepted 18 May 2015
Available online 29 May 2015

Keywords:
Description logics
OBDA
DL-Lite
Aggregate queries
Count queries
Complexity

ABSTRACT

The ontology based data access model assumes that users access data by means of an ontology, which is often described in terms of description logics. As a consequence, languages for managing ontologies now need algorithms not only to decide standard reasoning problems, but also to answer database-like queries. However, fundamental database aggregate queries, such as the ones using functions COUNT and COUNT DISTINCT, have received very little attention in this context, and even defining appropriate semantics for their answers over ontologies appears to be a non-trivial task. Our goal is to study the problem of answering database queries with aggregation in the context of ontologies. This paper presents an intuitive semantics for answering counting queries, followed by a comparison with similar approaches that have been taken in different database contexts. Afterwards, it exhibits a thorough study of the computational complexity of evaluating counting queries conforming to this semantics.

Our results show that answering such queries over ontologies is decidable, but generally intractable. However, our semantics promotes awareness on the information that can be obtained by querying ontologies and raises the need to look for suitable approximations or heuristics in order to allow efficient evaluation of this widely used class of queries.

1. Introduction

The growing popularity of ontologies as a paradigm for representing knowledge in the Semantic Web is based on the ability to describe incomplete information in the domain of interest.

Several variations of the *web ontology language* (*OWL*) have been formalised to manage ontologies. Most of these languages correspond to fragments of first order logic, which are called *description logics* (*DLs*) [1]. These fragments allow to define classifications of objects and formulate complex relationships for such classifications. Traditionally, literature has considered the decidability of standard reasoning problems, such as satisfiability, to be the most essential properties of any DL. In fact, a lot of effort has been made over last decades to maximise the expressive power of DLs while still keeping the decidability of these problems [1].

However, it has recently become clear that focusing on the decidability of satisfiability and related problems is not enough for practical applications of ontologies. For example, the concept of *ontology-based data access* (*OBDA*) requires reasoning tasks that are much more complex. The information system of OBDA applications consists of two layers: the *data layer* is formed by several data sources, whose exact structure is not interesting or not known to the clients, and a *conceptual layer*, where the clients can pose queries, and that is linked to the data layer by logical mappings [2,3]. Usually, data sources are relational databases, but the conceptual layer is an ontology formulated in a DL.

Hence, OBDA brings new challenges to system designers. Users should be able to pose database-style queries over ontologies using a reasonable amount of computational resources: the complexity of query answering should not be much higher than over usual relational databases. This presents serious restrictions on which ontology languages can be used in OBDA systems. In fact, this motivates the use of description logics of the *DL-Lite* family, underlying OWL 2 QL profile, which have been designed specifically to maximise expressive power while maintaining good query answering properties [4]. In particular, the computational complexity of answering simple queries such as *conjunctive queries* (*CQ*s) and *unions of conjunctive queries* (*UCQs*) over these DLs is the same as for relational databases [5,6].

It is natural to ask what happens when one moves beyond conjunctive queries. Recently, some attention has been paid to

* Corresponding author.
E-mail addresses: egor.kostylev@cs.ox.ac.uk (E.V. Kostylev), jreutter@ing.puc.cl (J.L. Reutter).

http://dx.doi.org/10.1016/j.websem.2015.05.003
1570-8268/© 2015 Elsevier B.V. All rights reserved.

the problem of answering various standard extensions of CQs and UCQs over ontologies. For example, [7,8], and [9] study path queries over ontologies, while [10,11], and [9] consider adding some form of negation to these simple queries. The general conclusion from these papers is that the complexity of evaluating such queries is usually higher than for CQs and UCQs, and even higher than for similar problems in relational databases. In some cases this difference in complexity is surprisingly high: e.g., while answering CQs with inequalities is known to be efficiently computable for relational databases, the problem is undecidable when such queries are posed over *DL-Lite* ontologies.

Yet there is another extension of CQs that has received little attention in the context of OBDA—*aggregate queries*. These queries answer questions such as "How many children does Ann have?" or "What is the average salary over each department in the Pandidakterion?" Usually, they combine various aggregation functions, such as `MIN`, `MAX`, `SUM`, `AVERAGE`, `COUNT`, and `COUNT DISTINCT` [12], together with a *grouping* functionality, as in the usual `GROUP BY` clause of SQL.

Aggregate queries are an important and heavily used part of almost every relational database query language, including SQL. Consequently, in the context of the Semantic Web we expect the need for answering queries with aggregates in OBDA settings, with applications such as SPARQL under entailment regimes [13]. But despite their importance the study of aggregate queries over ontologies has been lacking, save for a few exceptions [14].

The main reason for the lack of research in this direction is the difficulty of defining a semantics for aggregate queries over ontologies. The complication is that, unlike relational databases, in ontologies one assumes that every knowledge base instance is incomplete and describes a part of the infinite number of models of the knowledge base (i.e., the *open world assumption*, or *OWA*, is adopted), and a query may have a different answer on each of these models. For standard queries like CQs and UCQs one usually looks for the *certain answers* of queries, that is, the tuples that are answers in all possible models [5]. This approach, however, is not suitable for aggregate queries, as the following shows.

Consider a knowledge base where Ann is a parent and the ontology asserts that every parent has at least one child. If nothing else is assumed then for every positive integer n there exists a model where Ann has n children. Thus, the answer to the simple query "How many children does Ann have?" in different models of the knowledge base can be any number greater than or equal to 1. The syntactic intersection of these answers (i.e., applying standard certain answers semantics) trivially gives us the empty set, which is clearly not satisfactory. As a different approach, Calvanese et al. introduce *epistemic* semantics for aggregate queries [14]. In a nutshell, the idea is to apply the aggregation function only to known values. For example, the epistemic answer to the query above is 0, because we do not know anybody who is definitely a child of Ann. But this is clearly not the desired answer: since Ann is a parent we know that she has at least one child. Hence, the epistemic semantics does not always give a correct answer to `COUNT` queries.

We embark on the task of defining a suitable semantics for answering what we call *counting aggregate queries*, which are queries that use `COUNT` or `COUNT DISTINCT` functions. Motivated by the original idea of certain answers, we seek to find the maximal information that is common in the answers to such a query for all the models of a knowledge base.

As the first contribution of this paper we develop the notion of *aggregate certain answers* that can be explained as follows: a number n is in the aggregate certain answers of a counting query over a knowledge base if the result of the aggregation function of such query is not less than n in any possible model of the knowledge base. We show that this is a natural and useful semantics for aggregate queries that count. For instance, even if we do not know precisely how many children Ann has in the example above, we know that she has at least one, and thus 1 is an aggregate certain answer to the query.

Having established our semantics, we turn to the study of the algorithmic properties of aggregate certain answers computation for counting queries. We concentrate on ontologies of the *DL-Lite* family, in particular *DL-Lite*$_{core}$ and *DL-Lite*$_{\mathcal{R}}$ [5]. The choice of these DLs is twofold: first, as mentioned above, these formalisms are important in the OBDA settings; second, they are among the simplest DLs and hence are good candidates to begin with.

We start our study under the assumption that the query and the *terminology* (i.e., the *TBox*) are *fixed*, and the only input is the *assertions* (*ABox*). This corresponds to the *data complexity* of the problem in Vardi's taxonomy [15]. Somewhat surprisingly, our results show that the complexity of the problem is resilient to the choice of both DL and counting function and is **coNP**-complete in all cases. As far as we are aware, these are the first tight complexity bounds for answering aggregate queries in the presence of ontologies.

In order to get a further understanding of the computational properties of the problems we then proceed to the study of the *combined complexity* of computing the aggregate certain answers, that is, assuming that the query, ABox, and TBox form the input. Here we have an evidence supporting a difference for the choice of DL, albeit not for the choice of counting function. More precisely, we are able to show that the problem is **coNExpTime**-hard for *DL-Lite*$_{\mathcal{R}}$ and Π_2^p-hard for *DL-Lite*$_{core}$. Unfortunately, we do not have matching upper bounds: we show that the problem is in **coN2ExpTime** for *DL-Lite*$_{\mathcal{R}}$ and in **coNExpTime** for *DL-Lite*$_{core}$. Note that the hardness results are significant: they show that the combined complexity of aggregate query answering is higher than of answering standard conjunctive queries in case of *DL-Lite*$_{core}$, and higher than of answering relational algebra queries over complete databases in case of *DL-Lite*$_{\mathcal{R}}$.

This paper is an extended version of [16], but it contains substantial new material, including definitions, examples, full proofs for all theorems, and additional statements. Furthermore, it contains a revision on the upper bounds for combined complexity of the problem of computing the aggregate certain answers that were not correctly stated in [16].

Organisation. We start with an overview of related work in the following section, and basic background on *DL-Lite* and CQs in Section 3. We formally define the semantics of counting aggregate queries over *DL-Lite*$_{core}$ and *DL-Lite*$_{\mathcal{R}}$, and state corresponding decision problems of answering of these queries in Section 4. We establish *data complexity* of these problems in Section 5, and the bounds for the *combined complexity* in Section 6. We conclude in Section 7.

2. Related work

Aggregate queries have been part of most database query languages, such as SQL, for decades. Their theoretical formalisation can be found in, for example, [12]. Semantics for aggregate queries have been already defined for several database settings that feature incomplete information. For example, an inconsistent database instance (with respect to a set of constraints) describes a set of repairs, each of which satisfies the constraints and can be obtained from the instance by a minimal number of transformations. Aggregate queries over inconsistent databases were explored in [17], where the *range semantics* was defined. Intuitively, this semantics corresponds to the *interval* between the minimal and the maximal possible answers to the query amongst all the repairs of a given instance. The same semantics was adopted by [18,19] in the context of data exchange.