Complexity of Answering Counting Aggregate Queries over *DL-Lite*

Egor V. Kostylev[c], Juan L. Reutter[d]

[c] *University of Oxford and University of Edinburgh*
[d] *Pontificia Universidad Católica de Chile*

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 formalized 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 maximize 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.

Email addresses: `egor.kostylev@cs.ox.ac.uk` (Egor V. Kostylev), `jreutter@ing.puc.cl` (Juan L. Reutter)

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 maximize 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 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 ag-*

gregate 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 $\textit{DL-Lite}_{core}$ and $\textit{DL-Lite}_{\mathcal{R}}$ [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 $\textit{DL-Lite}_{\mathcal{R}}$ and Π_2^p-hard for $\textit{DL-Lite}_{core}$. Unfortunately, we do not have matching upper bounds: we show that the problem is in coN2ExpTime for $\textit{DL-Lite}_{\mathcal{R}}$ and in coNExpTime for $\textit{DL-Lite}_{core}$. 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 $\textit{DL-Lite}_{core}$, and higher than of answering relational algebra queries over complete databases in case of $\textit{DL-Lite}_{\mathcal{R}}$.

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].

Organization 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 $\textit{DL-Lite}_{core}$ and $\textit{DL-Lite}_{\mathcal{R}}$, 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.

However, the techniques from these papers cannot be immediately applied to ontologies because of several specific properties. In particular, these papers consider variations of the *closed world assumption*, whereas in ontologies the open world assumption is assumed. Furthermore, data exchange settings are based on source-to-target dependencies and weakly acyclic target dependencies. These rules out all types of recursion in ontological knowledge thus simplifying the study to a great extent.

In the context of ontologies aggregate queries were studied in [14]. In fact, in this work the range semantics itself was claimed to be trivially meaningless for aggregate queries over ontologies. For instance, for almost any knowledge base we can construct a model such that the aggregate value of an `AVERAGE` query evaluates to any number. Similar examples can be given for all other standard aggregation functions except for `COUNT` and `COUNT DISTINCT`, which are precisely the aggregates in the focus of this paper. As we will show, the computation of the upper bound of the range is almost trivial in these cases as well. But the lower bound of the range, that is, the minimal possible value described above, is completely natural and by no means trivial to compute. In fact, the lower bound of the range semantics is strongly related to our notion of aggregate certain answers as follows: a number is in the aggregate certain answers if and only if it is less than or equal to the lower bound of the range. Thus, this work on aggregate certain answers can be seen as an adaptation of the range semantics of [17] to ontologies.

The solution suggested in [14] is to give *epistemic* semantics to aggregate queries over ontologies, that is, to apply the aggregation function only to whose values which are witnessed in the ABox. We believe that this semantics is much less intuitive and informative than the certain answer semantics defined in this paper. Nevertheless, one can show that the epistemic semantics never gives more answers than aggregate certain answer semantics, so the first can be seen as an under-approximation of the second.

The default language for querying in the Semantic Web is SPARQL. Its very recent version SPARQL 1.1 includes aggregation functionalites, which are quite similar to such functionalities in SQL [20]. It also includes a specification for querying under entailment regimes, in particular, OWL 2 QL, the profile which has *DL-Lite* as logical foundation [13]. The specification adopts *active domain semantics* even for conjunctive queries, which means that the query answering process is decoupled in two steps: first fact entailment is performed with respect to the ontology, and afterwards the query is answered over the resulting set of facts. Unfortunately, fact entailment in OWL 2 QL is very limited. As a result, even conjunctive query answering in the sense of [5] is not possible under such semantics, because the anonymous part of the ontology is essentially ignored. Aggregate queries are to be evaluated using the same active domain semantics, hence this gives even less information than epistemic semantics. We believe that the certain answer semantics, which fully take into account the anonymous part of the ontology, is much more powerful and natural than the active domain semantics.

It is also worth to note that expressive DLs extended with aggregation functions were studied in [21]. Though, the authors did not consider any query language but concentrated on incorporating aggregation into the ontology language itself, by means of special concepts with arguments from so-called *concrete domains* (e.g., numbers). In this paper we consider aggregate query answering over less expressive DLs, which makes our settings incomparable with the settings of [21].

After the first publication of the results of this paper several semantics, including certain answer semantics, were independently defined in [22]. However, no complexity bounds were established.

3. Preliminaries

We start the formal part of this paper with standard notions on the two fragments of *DL-Lite* and conjunctive queries.

Syntax of *DL-Lite* Let $A_0, A_1, \ldots$ be *atomic concepts* and $P_0, P_1, \ldots$ be *atomic roles. Concepts* C and *roles* E of *DL-Lite* languages are formed by the grammar

$$B ::= A_i \mid \exists R, \qquad R ::= P_j \mid P_j^-,$$
$$C ::= B \mid \neg B, \qquad E ::= R \mid \neg R.$$

Concepts B and roles R are called *positive.*

A *TBox* is a finite set of inclusions. In the language of *DL-Lite*$_{core}$ these inclusions are of the form $B \sqsubseteq C$. In *DL-Lite*$_{\mathcal{R}}$ the form $R \sqsubseteq E$ is also allowed.

Let $\mathsf{Ind} = \{a, a_1, a_2, \ldots\}$ be a set of *individual names.* An *ABox* is a set of assertions of the forms $A_i(a)$ and $P_j(a_1, a_2)$ with $a, a_1, a_2 \in \mathsf{Ind}$. Without loss of generality, in what follows we assume that all the individual names from Ind appear in $\mathcal{A}$.

A *knowledge base* (or *KB*) $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ of a *DL-Lite* language contains a TBox $\mathcal{T}$ of the language and an ABox $\mathcal{A}$.

Semantics of *DL-Lite* An *interpretation* $\mathcal{I} = (\mathbb{D}^{\mathcal{I}}, \cdot^{\mathcal{I}})$ contains a (possibly infinite) *domain* of *elements* $\mathbb{D}^{\mathcal{I}}$ and maps each concept C to a subset $C^{\mathcal{I}}$ of $\mathbb{D}^{\mathcal{I}}$ and each role E to a binary relation $E^{\mathcal{I}}$ over $\mathbb{D}^{\mathcal{I}}$ such that

$$\begin{array}{l} (\exists R)^{\mathcal{I}} = \{d \mid \exists d' : (d, d') \in R^{\mathcal{I}}\}, \\ (P_j^-)^{\mathcal{I}} = \{(d_1, d_2) \mid (d_2, d_1) \in P_j^{\mathcal{I}}\}, \\ (\neg B)^{\mathcal{I}} = \mathbb{D}^{\mathcal{I}} \setminus B^{\mathcal{I}}, \\ (\neg R)^{\mathcal{I}} = (\mathbb{D}^{\mathcal{I}} \times \mathbb{D}^{\mathcal{I}}) \setminus R^{\mathcal{I}}, \end{array}$$

as well as each individual name a to an element $a^{\mathcal{I}} \in \mathbb{D}^{\mathcal{I}}$ such that $a_1^{\mathcal{I}} \neq a_2^{\mathcal{I}}$ in case of $a_1 \neq a_2$.

An interpretation $\mathcal{I}$ is a *model* of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ (written $\mathcal{I} \models \mathcal{K}$) if

- $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ for any inclusion $B \sqsubseteq C$ in $\mathcal{T}$,
- $R^{\mathcal{I}} \subseteq E^{\mathcal{I}}$ for any inclusion $R \sqsubseteq E$ in $\mathcal{T}$,
- $a^{\mathcal{I}} \in A_i^{\mathcal{I}}$ for any assertion $A_i(a)$ in $\mathcal{A}$, and
- $(a_1^{\mathcal{I}}, a_2^{\mathcal{I}}) \in P_j^{\mathcal{I}}$ for any assertion $P_j(a_1, a_2)$ in $\mathcal{A}$.

We also use $\models$ as logical implication for (sets of) inclusions and assertions under the semantics given above. For example, $\mathcal{T} \models A_1 \sqsubseteq A_2$ means that the TBox $\mathcal{T}$ implies the inclusion $A_1 \sqsubseteq A_2$.

The definitions above require that $a_1^{\mathcal{I}} \neq a_2^{\mathcal{I}}$ for different individual names a_1 and a_2 in every interpretation $\mathcal{I}$. By this we adopt the *unique name assumption* (*UNA*), which is conventional for *DL-Lite* [5, 6]. However, dropping this assumption does not affect any result of this paper, and after each of our proofs we discuss how to transfer the results if the UNA is not adopted.

Since in this paper we are interested in query answering, in what follows we usually assume all KBs to be *satisfiable*, that is, to have a model. This is justified by the fact that, as we see below, query answering over unsatisfiable knowledge bases is trivial.

Canonical model Next we give some standard notions which will be useful in the proofs of this paper.

The *canonical model* $Can(\mathcal{K})$ of a satisfiable $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is an interpretation with the domain $\mathbb{D}^{Can(\mathcal{K})}$ of all elements $d_{aR_1 \ldots R_m}$, where a is an individual name, $m \geq 0$, and $R_1, \ldots, R_m$ are positive roles such that

- if $m \geq 1$ then there is a positive concept B with $\mathcal{A} \models B(a)$ and $\mathcal{T} \models B \sqsubseteq \exists R_1$ but $\mathcal{A} \not\models R(a, a')$, for all a' and R with $\mathcal{T} \models R \sqsubseteq R_1$;
- $\mathcal{T} \models \exists R_{i-1}^- \sqsubseteq \exists R_i$ but $\mathcal{T} \not\models R_{i-1}^- \sqsubseteq R_i$, for each i, $1 < i \leq m$,

and the interpretation function defined for individual names a, atomic concepts A, and atomic roles P as follows:

$$\begin{array}{l} a^{Can(\mathcal{K})} = d_a, \\ A^{Can(\mathcal{K})} = \{\, a_c \mid \mathcal{K} \models A(a) \,\} \cup \\ \quad \{\, d_{aR_1 \ldots R_m} \mid m \geq 1,\ \mathcal{T} \models \exists R_n^- \sqsubseteq A \,\}, \\ P^{Can(\mathcal{K})} = \{\, (d_{a_1}, d_{a_2}) \mid \mathcal{K} \models P(a_1, a_2) \,\} \cup \\ \quad \{(d_{aR_1 \ldots R_{m-1}}, d_{aR_1 \ldots R_m}) \mid m \geq 1, \mathcal{T} \models R_m \sqsubseteq P\} \cup \\ \quad \{(d_{aR_1 \ldots R_m}, d_{aR_1 \ldots R_{m-1}}) \mid m \geq 1, \mathcal{T} \models R_m \sqsubseteq P^-\}. \end{array}$$

It is well-known that for any model $\mathcal{I}$ there exists a *homomorphism* from $Can(\mathcal{K})$ to $\mathcal{I}$, that is, a mapping $f : \mathbb{D}^{Can(\mathcal{K})} \to \mathbb{D}^{\mathcal{I}}$ such that $f(a^{Can(\mathcal{K})}) = a^{\mathcal{I}}$ for any $a \in \mathsf{Ind}$, and $\mathbf{d} \in S^{Can(\mathcal{K})}$ implies $f(\mathbf{d}) \in S^{\mathcal{I}}$ for any element or pair of elements $\mathbf{d}$ and any atomic concept or role S [5, 6].

Conjunctive queries The main building blocks of aggregate queries are the following most simple queries widely studied in the literature on knowledge bases. Formally, a *conjunctive query* (or *CQ*) is an expression of the form

$$q(\mathbf{x}) \text{ :- } \exists \mathbf{y}\ \phi(\mathbf{x}, \mathbf{y}), \tag{1}$$

where $\mathbf{x}$ is a tuple of *free* variables, $\mathbf{y}$ is a tuple of *existential* variables, and the *body* $\phi(\mathbf{x}, \mathbf{y})$ is a conjunction of *atoms* of the form $A_i(t)$ or $P_j(t_1, t_2)$, where t, t_1, t_2 are *terms*, that is, variables from $\mathbf{x} \cup \mathbf{y}$ or individual names from Ind. If the tuple of free variables $\mathbf{x}$ is empty, then the CQ is *Boolean*. The number of atoms in the body of q is denoted by $|q|$.

Semantics of CQs for a single interpretation is given as follows. Let $q(\mathbf{x})$ be a CQ of the form (1), $\mathcal{I}$ an interpretation, and $\mathbf{a} = (a_1, \ldots, a_n)$ a tuple of individual names from Ind, such that n is the size of $\mathbf{x}$. Let also $\mathbf{a}^{\mathcal{I}}$ be the extension of $\mathcal{I}$ to $\mathbf{a}$, that is, $\mathbf{a}^{\mathcal{I}} = (a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}})$. Then a *match* for $q(\mathbf{x})$ and $\mathbf{a}$ in $\mathcal{I}$ is a mapping $h : \mathbf{x} \cup \mathbf{y} \to \mathbb{D}^{\mathcal{I}}$, such that $h(\mathbf{x}) = \mathbf{a}^{\mathcal{I}}$ and $h'(\mathbf{t}) \in S^{\mathcal{I}}$ for every unary or binary atom $S(\mathbf{t})$ in $\phi(\mathbf{x}, \mathbf{y})$ with $h'(t) = h(t)$ for $t \in \mathbf{x} \cup \mathbf{y}$ and $h'(t) = t^{\mathcal{I}}$ for $t \in \mathsf{Ind}$. If such a match exists, then we also say that the CQ *holds* for $\mathcal{I}$ and $\mathbf{a}$, and write $\mathcal{I} \models q(\mathbf{a})$.

Of course, we are interested not in a single interpretation but in all models of a knowledge base. To this end, the following semantics is usually adopted. A tuple $\mathbf{a}$ is in the *certain answers* to a CQ (1) over a KB $\mathcal{K}$ if and only if $\mathcal{I} \models q(\mathbf{a})$ holds for every model $\mathcal{I}$ of $\mathcal{K}$. It is well-known that a tuple $\mathbf{a}$ of individual names is a certain answer to a CQ $q(\mathbf{x})$ over a satisfiable $\mathcal{K}$ if and only if $Can(\mathcal{K}) \models q(\mathbf{a})$.

4. Counting Queries over Ontologies

The ability to evaluate aggregate queries is a default in every DBMS and is in the standard of SQL. However, as mentioned in the introduction, little attention to this type of queries has been paid in the context of ontologies. Starting to fill this gap in this section we formally define counting aggregate queries over ontologies. We begin with adapting the notion of aggregate conjunctive queries from the database settings into our context and define how to evaluate these queries, first over a particular interpretation and then over a knowledge base. Then we compare our notion with range semantics in inconsistent databases. We note that, although our results in the following sections are limited to ontologies of the *DL-Lite* family, our definitions in this section are general and do not depend on the choice of description logic.

4.1. Syntax and Semantics of Counting Queries

We begin with the syntax of aggregate queries and their semantics with respect to a single interpretation. The last is, essentially, the semantics in the relational database sense, where only a single world is considered. We generally follow standard database theory literature on the topic, such as [12].

Definition 1. An *aggregate conjunctive query* (or *ACQ*) is an expression

$$q(\mathbf{x}, f(\mathbf{z})) \text{ :- } \exists \mathbf{y}\ \phi(\mathbf{x}, \mathbf{y}, \mathbf{z}), \tag{2}$$

where $\mathbf{x}$ is a tuple of *free* variables, $\mathbf{y}$ is a tuple of *existential* variables and $\mathbf{z}$ is a tuple of *aggregation* variables, the *body* $\phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is a conjunction of atoms of the form $A_i(t)$ or $P_j(t_1, t_2)$ with terms t, t_1, t_2 from $\mathbf{x} \cup \mathbf{y} \cup \mathbf{z} \cup \mathsf{Ind}$, and $f(\mathbf{z})$ is an *aggregation function*. Similarly to plain CQs, if $\mathbf{x}$ is empty, then the ACQ is *Boolean*.

Amongst typical aggregation functions considered in database theory and practice we find count $Count()$, count distinct $Cntd(z)$, minimum $Min(z)$, maximum $Max(z)$, average $Avg(z)$, and sum $Sum(z)$. In this paper we concentrate on the first two functions (note that count is nullary while count distinct is unary). We call ACQs using these two functions *counting ACQs* in general, while *count* and *count distinct ACQs* individually. Note that $Count()$ and $Cntd(z)$ are essentially the only standard counting functions in SQL:1999 [23] and SPARQL 1.1 [20].

Before proceeding with the formal semantics of counting aggregate queries, we give a couple of simple examples.

Example 2. Consider a vocabulary with an atomic concept *Parent* and role *HasChild*. The query

$$q^1_{ex}(x, Count()) \text{ :- } \exists y \ Parent(x) \wedge HasChild(x, y)$$

is a count ACQ that is intended to count the children of each parent. The query

$$q^2_{ex}(Cntd(z)) \text{ :- } \exists y \ Parent(y) \wedge HasChild(y, z)$$

is a (Boolean) count distinct ACQ. This query counts all different children having a parent. The graphical representations of these queries in the conventional for this paper way are given in Figure 1.

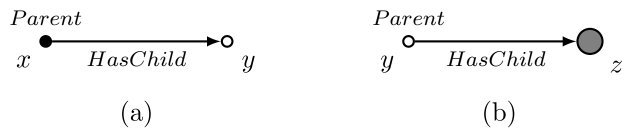

Figure 1: Example queries q^1_{ex} (a) and q^2_{ex} (b): free, existential and aggregation variables are depicted by black, white, and big grey nodes respectively.

For the formal semantics we need the following auxiliary notion. The *core* of an ACQ q of the form (2) is the CQ $\bar{q}(\mathbf{x} \cup \mathbf{z})$:- $\exists \mathbf{y} \ \phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$. In other words, the core of a query has the same body as the original ACQ but considers the aggregation variables as free. Finally, it will be useful to write $\mathbb{N}^\infty$ for the set of natural numbers with 0 and $+\infty$.

Definition 3. A count ACQ $q(\mathbf{x}, Count())$ *holds* for an interpretation $\mathcal{I}$, tuple $\mathbf{a}$ of individual names from Ind, and number $n \in \mathbb{N}^\infty$ (written $\mathcal{I} \models q(\mathbf{a}, n)$) iff n is the number of distinct matches for the core $\bar{q}$ and $\mathbf{a}$ in $\mathcal{I}$.

Informally, for a tuple of individual names a count query returns the number of possible matches of the body to the interpretation which send the free variables to the (interpretations of the) individual names.

Definition 4. A count distinct ACQ $q(\mathbf{x}, Cntd(z))$ *holds* for an interpretation $\mathcal{I}$, tuple $\mathbf{a}$ of individual names, and number $n \in \mathbb{N}^\infty$ (written $\mathcal{I} \models q(\mathbf{a}, n)$) iff n is the number of distinct $a' \in \mathsf{Ind}$ such that $\mathcal{I} \models \bar{q}(\mathbf{a}, a')$ for the core $\bar{q}$ of q.

The intention of this type of queries is different from count ACQs—they count the number of different possibilities of mapping z into the interpretation such that the rest of the query can be properly matched. This difference is illustrated in the following example.

Example 5. Coming back to Example 2, consider an interpretation $\mathcal{I}$ which interprets all names by themselves and satisfies

$$\begin{array}{lcl} Parent^{\mathcal{I}} & = & \{\text{Ann}\}, \\ HasChild^{\mathcal{I}} & = & \{(\text{Ann}, \text{Joe})\}. \end{array}$$

It is not difficult to see that $\mathcal{I} \models q^1_{ex}(\text{Ann}, 1)$ and $\mathcal{I} \models q^2_{ex}(1)$, that is, in $\mathcal{I}$ Ann has one child, and there is precisely one child with a parent. For an interpretation $\mathcal{J}$ which interprets all names by themselves, and satisfies

$$\begin{array}{lcl} Parent^{\mathcal{J}} & = & \{\text{Ann}, \text{Peter}\}, \\ HasChild^{\mathcal{J}} & = & \{(\text{Ann}, \text{Joe}), (\text{Ann}, \text{Rose}), \\ & & \qquad (\text{Peter}, \text{Joe})\}, \end{array}$$

we have $\mathcal{J} \models q^1_{ex}(\text{Ann}, 2)$, $\mathcal{J} \models q^1_{ex}(\text{Peter}, 1)$, and $\mathcal{J} \models q^2_{ex}(2)$.

Since we concentrate on counting aggregate queries, we do not give formal semantics for queries with other aggregation functions. However the intuition behind their semantics is very similar to count distinct ACQs: a value is an aggregate answer if it is the result of applying the aggregation function to the multiset of all elements witnessing the aggregation variable by matches for the core query in the interpretation.

4.2. Certain Answers of Counting Queries over Ontologies

A knowledge base normally describes not a single model but an infinite number of them. This is why one is typically interested in computing the certain answers of queries over a KB, which are usually defined as the intersection of the answers to the query over all possible models of KB [5, 10].

Unfortunately, a definition based on such a syntactical intersection is of little use for ACQs. Indeed, this intersection is almost always empty, as illustrated in the following example.

Example 6. Let $\mathcal{K}_{ex}$ be a knowledge base with the TBox consisting of the inclusion $Parent \sqsubseteq \exists HasChild$, and the ABox consisting of the assertion $Parent(\text{Ann})$. The interpretations $\mathcal{I}$ and $\mathcal{J}$ from Example 5 are models of $\mathcal{K}_{ex}$. Also, in this example we listed all the answers to the queries q^1_{ex} and q^2_{ex}, so it is straightforward to check that the intersections of the relevant sets for only these two models of $\mathcal{K}_{ex}$ are already empty (of course, there are infinitely many other models).

This suggests to avoid using syntactic intersection when defining the semantics of ACQs over ontologies. In the context of OBDA this problem has been identified before by [14]. The proposed solution was to concentrate only on aggregating over *epistemic* knowledge, that is, over the (interpretations of) individual names which are explicitly mentioned in the ABox. Such epistemic answers usually have a non-empty intersection over all the models for the standard aggregate queries, including Max and $Average$. However, for counting queries such answer may be somehow non-satisfactory. For example, the epistemic answer to the ACQ q^1_{ex} over $\mathcal{K}_{ex}$ from Example 6 is $(\text{Ann}, 0)$, because we do not know any individual who is definitely a child of Ann, even if we know for sure about the existence of one of them.

In order to define our semantics we come back to the original intuition behind certain answers: these are the answers that hold in every possible model. For the case of counting ACQs, we cannot be certain of the precise result of the count function. But if the result of a query is greater than or equal to a number n in every model of a knowledge base, then we can say with certainty that the result of a query is always at least n.

This is why we suggest the following definition of certain answers for counting ACQs over DLs: it is the set of all numbers no greater than the *minimum* of the values of the counting function over all the possible models of the KB.

Definition 7. A non-negative number $n \in \mathbb{N}^\infty$ is in the *aggregate certain answers* $Cert(q, \mathbf{a}, \mathcal{K})$ to a counting ACQ q and tuple $\mathbf{a}$ of individual names over a KB $\mathcal{K}$ iff $n \leq \min_{\mathcal{I} \models \mathcal{K}} \{k \mid \mathcal{I} \models q(\mathbf{a}, k)\}$.

The following example illustrates the described intuition.

Example 8. For the KB and queries defined in the previous examples we have that $Cert(q^1_{ex}, \text{Ann}, \mathcal{K}_{ex})$ contains 0 and 1, but not 2 or any greater number. This reflects the fact that in any model of the KB we have at least one child of Ann (and at least 0 as well). By the same reasons $Cert(q^2_{ex}, \mathbf{a}_\emptyset, \mathcal{K}_{ex})$ consists of 0 and 1. (Here and further in the paper $\mathbf{a}_\emptyset$ is the empty tuple of individual names).

Note that under our definition every number that is smaller or equal than the minimum value over all the models is an aggregate certain answer. As mentioned above, this is in line with the idea that certain answers are all those answers that hold in every model: we cannot ask for the precise value of the counting function, but we can ask whether such value is always *at least* n. We focus on the decision problem of answering this question because it resembles usual database decision problems. However, the problem of computing this minimum (and its corresponding decision problem of asking whether n is the minimum) is also important and needs to be studied. We comment on this in the conclusion.

Similarly to the case of CQs, if a KB is unsatisfiable, then by this definition any number is a certain answer to any counting query. Hence, this case is vacuous, and we assume that all KBs are satisfiable.

A definition such as above is non-trivial only for counting standard aggregate queries. Indeed, it relies on their simple property that the minimum above can potentially be any number greater than or equal to 0. For other aggregation functions it is not the case: e.g., such a minimum for $Average$ is trivially almost always $-\infty$.

4.3. Range Semantics of Aggregate Queries

As mentioned in the introduction, aggregate queries have been explored in other settings which deal with many models. In particular, here we compare our notion with that of *range semantics*, defined in the context of inconsistent databases in [17] and later adopted in data exchange [18, 19]. This semantics focuses on the *interval* of possible aggregation values over all models. In the context of counting ACQs over ontologies it can be defined as follows.

The *range of answers* to a counting ACQ q and a tuple $\mathbf{a}$ of individual names over a KB $\mathcal{K}$ is the interval $[m(q,\mathbf{a},\mathcal{K}), M(q,\mathbf{a},\mathcal{K})]$, where

$$m(q,\mathbf{a},\mathcal{K}) = \min_{\mathcal{I}\models\mathcal{K}}\{k \mid \mathcal{I} \models q(\mathbf{a},k)\},$$
$$M(q,\mathbf{a},\mathcal{K}) = \max_{\mathcal{I}\models\mathcal{K}}\{k \mid \mathcal{I} \models q(\mathbf{a},k)\}.$$

It is easy to see that the lower bound of the range interval coincides with the maximal certain answer according to Definition 7. Considering the upper bound, let us come back to the examples in the previous section. We can find a model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \models q^1_{ex}(\text{Ann}, n)$ for any number $n \geq 1$. Hence, in this case the upper bound is $+\infty$. Similar situation is in the case of q^2_{ex}. The following proposition says that this is indeed not unusual.

Proposition 9. *Given a counting ACQ q, a tuple $\mathbf{a}$ of individual names, and a satisfiable DL-Lite KB $\mathcal{K}$, the upper endpoint $M(q,\mathbf{a},\mathcal{K})$ of the range of answers belongs to the set $\{0, 1, +\infty\}$, and can be computed in polynomial time (in the size of q and $\mathcal{K}$).*

Proof. Let $q(\mathbf{x}, f(\mathbf{z}))$:- $\exists \mathbf{y}\ \phi(\mathbf{x},\mathbf{y},\mathbf{z})$ and $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$.

Let us start by considering situations when the upper endpoint is 0. According to the definitions, the only possibility for $M(q,\mathbf{a},\mathcal{K})$ to be 0 is that the core $\bar{q}$ does not have a match sending $\mathbf{x}$ to $\mathbf{a}$ in any model of $\mathcal{K}$. Next we show that this case is possible and can be checked in polynomial time. Consider a fresh individual name a_u for every existential or aggregate variable $u \in \mathbf{y} \cup \mathbf{z}$, and a function g mapping all individual names to themselves, free variables $\mathbf{x}$ to $\mathbf{a}$, and every other variable u of q to a_u. Let $\mathcal{A}_{g(q(\mathbf{a}))}$ be an ABox, which contains the assertion $S(g(\mathbf{t}))$ for every (unary or binary) atom $S(\mathbf{t})$ in $\phi(\mathbf{x},\mathbf{y},\mathbf{z})$, and $\mathcal{K}'$ be a KB $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}_{g(q(\mathbf{a}))} \rangle$. We claim that $\bar{q}$ does not have a match sending $\mathbf{x}$ to $\mathbf{a}$ in any model of $\mathcal{K}$ if and only if $\mathcal{K}'$ is not satisfiable. Indeed, if $\mathcal{K}'$ is satisfiable, then every model $\mathcal{I}$ of $\mathcal{K}'$ is also a model of $\mathcal{K}$, and the function which maps every term t to the interpretation of the individual name $g(t)$ under $\mathcal{I}$ is such a match. On the other hand, if there is a model $\mathcal{I}$ of $\mathcal{K}$ with a match h for $\bar{q}$ and $\mathbf{a}$ in $\mathcal{I}$, then it can be extended to a model of $\mathcal{K}'$ by setting $a_u^{\mathcal{I}} = h(u)$ for every $u \in \mathbf{y} \cup \mathbf{z}$. Such a situation is clearly possible and checking satisfiability can be performed in polynomial time.

Let us proceed to situations when the upper endpoint is 1. Assume that the extended KB $\mathcal{K}'$ is satisfiable, and q is a count ACQ without existential variables, that is, $\mathbf{z}$ and $\mathbf{y}$ are empty. In this case any model of $\mathcal{K}'$ (which is also a model of $\mathcal{K}$) has a match for $\bar{q}$ and $\mathbf{a}$. On the other hand, none of the models can have more, because of the lack of existential variables. Hence, we have $M(q,\mathbf{a},\mathcal{K}) = 1$. Again, such a situation is possible and can be checked in polynomial time.

For the remaining case we show that if the extended KB $\mathcal{K}'$ is satisfiable and $\mathbf{z}\cup\mathbf{y}$ is not empty, then the upper endpoint is $+\infty$. Consider a model $\mathcal{I}$ of $\mathcal{K}'$ (which is also a model of $\mathcal{K}$), a match h for $\bar{q}$ and $\mathbf{a}$ in $\mathcal{I}$, and a variable u, which is either (the only variable) from $\mathbf{z}$, if it exists in case of q is count distinct, or from $\mathbf{y}$ otherwise. Construct an interpretation $\mathcal{I}_{+\infty}$ with the domain extending $\mathbb{D}^{\mathcal{I}}$ by an infinite number of new elements d_i, $i \geq 1$, and interpretation function extending $\cdot^{\mathcal{I}}$ for atomic concepts A and roles P as follows:

$$\begin{array}{lll}
d_i \in A^{\mathcal{I}_{+\infty}} & \text{iff} & h(u) \in A^{\mathcal{I}}, \\
(d_i, d) \in P^{\mathcal{I}_{+\infty}} & \text{iff} & (h(u), d) \in P^{\mathcal{I}}, \\
(d, d_i) \in P^{\mathcal{I}_{+\infty}} & \text{iff} & (d, h(u)) \in P^{\mathcal{I}}.
\end{array}$$

Essentially, $\mathcal{I}_{+\infty}$ extends $\mathcal{I}$ by an infinite number of copies of $h(u)$. On one hand, $\mathcal{I}_{+\infty}$ is a model for both $\mathcal{K}'$ and $\mathcal{K}$. On the other, the mapping h is still a match for $\bar{q}$ and $\mathbf{a}$ in $\mathcal{I}_{+\infty}$, and all the mappings h_i, $i \geq 1$, such that $h_i(u) = d_i$ and $h_i(u') = h(u')$ for all $u' \neq u$ are also matches. Moreover, these matches create new witnesses for the aggregation variable z in case of q is count distinct. So, we have that $\mathcal{I}_{+\infty} \models q(\mathbf{a}, +\infty)$, as required. □

We may thus say that the aggregate certain answers semantics from Definition 7 is an adaptation of the range semantics of [17] to ontologies.

5. Data Complexity of Counting Queries

In this section we begin with the data complexity of the problem of computing aggregate certain answers. We concentrate our study on the defined above variants of description logics from the *DL-Lite* family and counting ACQs. Formally, let $\mathcal{X} \in \{core, \mathcal{R}\}$, $\mathcal{T}$ be a TBox over *DL-Lite*$_\mathcal{X}$, and $q(\mathbf{x}, f(z))$ be a counting ACQ. We are interested in the following family of problems.

f-Aggregate Certain Answers $(\mathcal{T}, q)$	
Input:	ABox $\mathcal{A}$, tuple $\mathbf{a}$, and number $n \in \mathbb{N}^\infty$.
Question:	Does $n \in Cert(q, \mathbf{a}, \langle\mathcal{T}, \mathcal{A}\rangle)$?

The main result of this section is the coNP-completeness of Aggregate Certain Answers for count and count distinct queries over both *DL-Lite*$_\mathcal{R}$ or *DL-Lite*$_{core}$ knowledge bases. In particular, this implies a jump from the complexity of answering standard conjunctive queries over knowledge bases, which is in P (assuming P $\neq$ NP). So why is answering aggregate queries more complex? The main reason is that one can no longer compute aggregate certain answers by simply posing queries over the canonical model: a counting query may yield a given number n when posed over the canonical model, but it is possible to compress the canonical model by identification of elements and obtain a model with a smaller number of witnesses for the counting query.

We thus need a much more involved algorithm for computing these answers. Let $q(\mathbf{x}, f(z))$ be a counting ACQ and $\mathcal{K} = \langle\mathcal{T}, \mathcal{A}\rangle$ a knowledge base. In order to check that $n \notin Cert(q, \mathbf{a}, \langle\mathcal{T}, \mathcal{A}\rangle)$ for a tuple $\mathbf{a}$ of individual names and a number n, we need to find a counterexample in the form of a model $\mathcal{I}_0$ of $\mathcal{K}$ such that $\mathcal{I}_0 \models q(\mathbf{a}, n_0)$, for some $n_0 < n$. Hence, the idea of the proofs below is to show the following small model property: if such a counterexample exists, then there exists a counterexample of polynomial size (with respect to $|\mathcal{A}|$). The coNP algorithm then consists of guessing this small counterexample $\mathcal{I}_0$ and evaluate q over $\mathcal{I}_0$ in polynomial time to verify that $\mathcal{I}_0 \models q(\mathbf{a}, n_0)$, for some $n_0 < n$. We also show that the coNP bound is tight.

5.1. Count Queries

We first establish the upper bound for count ACQs. Let us begin by introducing the required notation.

Let $\mathcal{I}_0$ be a model of a KB $\mathcal{K} = \langle\mathcal{T}, \mathcal{A}\rangle$ and $\mathbb{D}^*$ be a subset of the domain $\mathbb{D}^{\mathcal{I}_0}$ of $\mathcal{I}_0$ which includes all the interpretations $a^{\mathcal{I}_0}$ of individual names a from Ind. Since $\mathcal{I}_0$ is a model, there exists a homomorphism f from the canonical model $Can(\mathcal{K})$ to $\mathcal{I}_0$. Consider a mapping $f' : \mathbb{D}^{Can(\mathcal{K})} \to (\mathbb{D}^* \cup \mathbb{D}^{Can(\mathcal{K})})$ defined as

$$f'(d) = \begin{cases} f(d), & \text{if } f(d) \in \mathbb{D}^*, \\ d, & \text{otherwise.} \end{cases} \tag{3}$$

Then the *interleaving* $\mathcal{I}_0^{\mathbb{D}^*}$ of $\mathcal{I}_0$ for $\mathbb{D}^*$ is the *image* of $Can(\mathcal{K})$ under f', that is the interpretation whose domain is the range of f', the interpretation of individual names coincides with the one in $\mathcal{I}_0$, and where $f'(\mathbf{d})$ is in the interpretation of an atomic concept or role S and a tuple (an element or a pair of elements) $\mathbf{d}$ if and only if $\mathbf{d} \in S^{Can(\mathcal{K})}$. Clearly, $\mathcal{I}_0^{\mathbb{D}^*}$ is a model of $\mathcal{K}$. Essentially, the transformation from $\mathcal{I}_0$ to $\mathcal{I}_0^{\mathbb{D}^*}$ drops everything which is not enforced by f and preserves $\mathcal{I}_0$ only on $\mathbb{D}^*$, while replacing the rest with the corresponding parts of the canonical model.

Next important notion is, essentially, the restriction of the interleaving $\mathcal{I}_0^{\mathbb{D}^*}$ to all the elements which are reachable from a given element by (undirected) paths through role interpretations of bounded length and without intermediate nodes from $\mathbb{D}^*$. Formally, let k be a positive number. Then, for every element d in the domain of $\mathcal{I}_0^{\mathbb{D}^*}$ which is not in $\mathbb{D}^*$, the *k-neighbourhood* $\mathcal{N}_k(d)$ is the set of all domain elements d' such that there exist $d_0, \ldots, d_i$, $i \leq k$, with

- $d_0 = d$, $d_i = d'$,
- $d_j \notin \mathbb{D}^*$ for all j, $0 \leq j < i$, and

- there exists a positive role R_j (that is, an atomic role or its inverse) such that (d_j, d_{j+1}) is in the interpretation of R_j under $\mathcal{I}_0^{\mathbb{D}^*}$ for all $0 \leq j < i$.

Note that according to this definition the last element d_i can be in $\mathbb{D}^*$.

Recall that the elements in the domain of $\mathcal{I}_0^{\mathbb{D}^*}$ which are not in $\mathbb{D}^*$ are the elements of the canonical model, so they have the form d_w with $w = aR_1 \dots R_m$, $m \geq 1$. Moreover, by the construction of the interleaving, for the k-neighbourhood $\mathcal{N}_k(d)$ of an element d there exists an element d_w of the canonical model such that $f'(d_w) \in \mathcal{N}_k(d)$ and w is a prefix of w' for any element $d_{w'}$ with $f'(d_{w'}) \in \mathcal{N}_k(d)$. We call this element the *root* of $\mathcal{N}_k(d)$. Note that $f'(d_w)$ can be either in $\mathbb{D}^*$ or d_w itself.

We define the following equivalence relation on elements in the domain of $\mathcal{I}_0^{\mathbb{D}^*}$ which are not in $\mathbb{D}^*$: $d \sim^k_{\mathbb{D}^*} d'$ for d and d' with roots d_w and $d'_{w'}$ if and only if

1. for any word w_1
   (a) $f'(d_{ww_1}) \in \mathcal{N}_k(d)$ iff $f'(d_{w'w_1}) \in \mathcal{N}_k(d')$,
   (b) $f'(d_{ww_1}) \in \mathcal{N}_k(d) \cap \mathbb{D}^*$ iff $f'(d_{w'w_1}) \in \mathcal{N}_k(d') \cap \mathbb{D}^*$;
2. $|w| \equiv |w'| \pmod{2k+1}$, that is, $|w|$ and $|w'|$ are congruent modulo $2k+1$.

The first requirement essentially says that the neighbourhoods of equivalent elements are isomorphic. The second one guarantees that the "distance" between equivalent elements in the canonical model is large enough.

Having this definitions at hand we are ready to prove the following key lemma.

Lemma 10. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_\mathcal{R}$ KB and $q(\mathbf{x}, Count())$ be a count ACQ. Then there exists a number ℓ depending only on q and $\mathcal{T}$ such that if there is a model $\mathcal{I}_0$ of $\mathcal{K}$ with $\mathcal{I}_0 \models q(\mathbf{a}, n_0)$ for a tuple $\mathbf{a}$ of individual names and a number $n_0 \leq (|\mathsf{Ind}| + |\mathcal{T}|)^{|q|}$, then there is a model $\mathcal{I}'$ of $\mathcal{K}$ with domain of $O(|\mathsf{Ind}|^\ell)$ elements with $\mathcal{I}' \models q(\mathbf{a}, n')$ for some number $n' \leq n_0$.*

Proof. Let $q(\mathbf{x}, Count())$:- $\exists \mathbf{y}\ \phi(\mathbf{x}, \mathbf{y})$.

Let $\mathbb{D}^*$ be all the elements of $\mathbb{D}^{\mathcal{I}_0}$ which are either interpretations of individual names from Ind or images of the variables $\mathbf{y}$ by matches for the core $\bar{q}$ and $\mathbf{a}$ in $\mathbb{D}^{\mathcal{I}_0}$. Let $\mathcal{I}'$ be the interpretation obtained from the interleaving $\mathcal{I}_0^{\mathbb{D}^*}$ of $\mathcal{I}_0$ for $\mathbb{D}^*$ by identifying all elements d and d' such that $d \sim^{|q|}_{\mathbb{D}^*} d'$. First, note that $\mathcal{I}'$ is indeed a model of $\mathcal{K}$, because it is an image of a canonical model and satisfy all negative inclusions of $\mathcal{T}$ by the construction of the equivalence $\sim^{|q|}_{\mathbb{D}^*}$. Furthermore, note that the identification does not create new matches for the core $\bar{q}$ and $\mathbf{a}$, since we are identifying elements with the same $|q|$-neighbourhood, and they are at sufficient distance from each other.

To complete the proof we need to show that $\mathcal{I}'$ has no more than $O(|\mathsf{Ind}|^\ell)$ elements for some ℓ. By definition, every element of the canonical model $Can(\mathcal{K})$ different from d_a for $a \in \mathsf{Ind}$, has at most $|\mathcal{T}| + 1$ immediate neighbours (i.e., elements connected by an atomic role or inverse). Hence, by construction, each element in the interleaving $\mathcal{I}_0^{\mathbb{D}^*}$ which is not in $\mathbb{D}^*$ also has at most $|\mathcal{T}| + 1$ immediate neighbours. It means that the $|q|$-neighbourhood of any element cannot have more than $(|\mathcal{T}| + 1)^{|q|}$ elements. Moreover, since the equivalence $\sim^{|q|}_{\mathbb{D}^*}$ only needs to preserve $\mathbb{D}^*$, and the interpretations of concepts and roles in the neighbourhood are completely defined by its root, the number of different equivalence classes of $|q|$-neighbourhoods generated by $\sim^{|q|}_{\mathbb{D}^*}$ on $\mathcal{I}_{\mathbb{D}^*}$ belongs to $O(|\mathbb{D}^*|^{|\mathcal{T}|^{|q|}})$. Since $|\mathbb{D}^*| \leq n_0|q| + |\mathsf{Ind}|$, the number of elements in the domain of $\mathcal{I}'$ belongs to

$$O((|q| \cdot (|\mathsf{Ind}| + |\mathcal{T}|)^{|q|} + |\mathsf{Ind}|)^{|\mathcal{T}|^{|q|}}).$$

Since $\mathcal{T}$ and q are fixed, we conclude that there exists a number ℓ as required. □

We are ready to state and prove the main result of this section.

Lemma 11. *Let $\mathcal{T}$ be a fixed DL-Lite$_\mathcal{R}$ TBox and $q(\mathbf{x}, Count())$ be a fixed count ACQ. Checking whether $n \in Cert(q, \mathbf{a}, \langle \mathcal{T}, \mathcal{A} \rangle)$, for an ABox $\mathcal{A}$, tuple of individual names $\mathbf{a}$, and number n can be done in* coNP.

Proof. The knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ always has a model with the domain of the size no greater than $|\mathsf{Ind}| + |\mathcal{T}|$. (Recall that we assume $\mathcal{K}$ to be satisfiable.) For example, this is the case for the model constructed the image of $Can(\mathcal{K})$ under the homomorphism h defined as

$$\begin{array}{lcl} h(d_a) & = & d_a, \\ h(d_{aR_1\ldots R_m}) & = & d_{R_m}, \end{array}$$

where d_{R_m} for roles R_m in $\mathcal{T}$ are fresh elements. There exist at most $(|\mathsf{Ind}| + |\mathcal{T}|)^{|q|}$ matches for the core of q in this model. Hence, without loss of generality, we may assume that $n \leq (|\mathsf{Ind}| + |\mathcal{T}|)^{|q|}$, because otherwise the answer to our decision problem is trivially "no". Note that this bound is polynomial in the size of the input, since q is fixed.

By Lemma 10 there exists a number ℓ depending only on q and $\mathcal{T}$ (i.e., fixed in the settings of this lemma), such that if there is a falsifying model $\mathcal{I}_0$, that is a model of $\mathcal{K}$ with $\mathcal{I}_0 \models q(\mathbf{a}, n_0)$ for $n_0 < n$, then there is a model $\mathcal{I}$ of $\mathcal{K}$ over $O(|\mathsf{Ind}|^{\ell})$ elements with $\mathcal{I} \models q(\mathbf{a}, n')$ for some number $n' \leq n_0$. Hence, to verify that n is a certain answer, we need to check whether $\mathcal{J} \models \mathcal{K}$ implies that there are at least n matches for the core of q and $\mathbf{a}$ to $\mathcal{J}$ in all interpretations $\mathcal{J}$ over $O(|\mathsf{Ind}|^{\ell})$ elements. However, such a check is in coNP, since a check for a particular $\mathcal{J}$ can be done in polynomial time (recall, that q and $\mathcal{T}$ are fixed in this lemma). □

Not that the proof of the upper complexity bound does not depend on the UNA, so the result also holds if we do not adopt it. Next we set the matching lower bound.

The proof of the following lemma is by reduction from the complement of the 3-*colouring problem*. The input for the 3-colouring problem is an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges, and the answer is positive if and only if the graph has a 3-colouring. This problem is known to be NP-complete [24]. In this proof, as well as in the proof of the following Lemma 16 which also makes use of a reduction from the 3-colouring problem, it will be convenient to assume that $\mathcal{G}$ does not have isolated vertices. Clearly, this restriction does not change the complexity of the problem.

Lemma 12. *There exist a DL-Lite$_{core}$ TBox $\mathcal{T}$ and a Boolean count ACQ q such that checking whether $n \in Cert(q, \mathbf{a}_\emptyset, \langle \mathcal{T}, \mathcal{A} \rangle)$, for an ABox $\mathcal{A}$, a number n and the empty tuple $\mathbf{a}_\emptyset$, is* coNP*-hard.*

Proof. We begin with the definition of a fixed TBox and count query, then explain how to construct an ABox on the base of an instance of the problem, and finally show that number 4 is a certain answer if and only if the instance has no 3-colouring.

Let $Edge$ and $HasCol$ be atomic roles and Col be an atomic concept. Fix a *DL-Lite$_{core}$* TBox

$$\mathcal{T} = \{\exists Edge \sqsubseteq \exists HasCol, \exists HasCol^- \sqsubseteq Col\},$$

which is intended to assign a colour to every vertex. Fix also a Boolean count ACQ

$$\begin{array}{r} q_{d\text{-}count}(Count()) \text{ :- } \exists y_v, y_u, y_c, y_c' \\ Edge(y_v, y_u) \wedge HasCol(y_v, y_c) \wedge HasCol(y_u, y_c) \\ \wedge\, Col(y_c'). \end{array}$$

The graphical representation of this query is given in Figure 2.

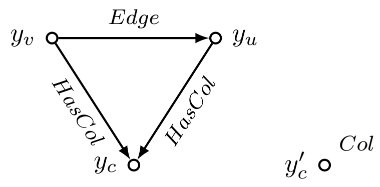

Figure 2: Query $q_{d\text{-}count}(Count())$. It consists of two disconnected parts, one of triangular form, which is intended to detect colourings having an edge assigned with the same colour on both ends, and another being an isolated variable detecting all the colours used in the colouring.

Consider now an instance $\mathcal{G}(\mathcal{V}, \mathcal{E})$ of the complement of the 3-colouring problem and construct the ABox on its base as follows.

Let $\mathsf{Ind} = \mathcal{V} \cup \{r, g, b, a\}$. The ABox $\mathcal{A}$ contains

- assertions $Edge(v, u)$ and $Edge(u, v)$ for each edge $(v, u) \in \mathcal{E}$,
- assertion $Col(c)$ for each $c \in \{r, g, b\}$,

- assertions $Edge(a, a)$ and $HasCol(a, r)$.

The part of the canonical model of $\mathcal{K}$ related to an edge in $\mathcal{G}$ is depicted in Figure 3 (the names of the elements are omitted for brevity). Intuitively, the individual names r, g and b represent colours, and the role $HasCol$ is intended to connect a vertex in the graph with a colour. The individual name a plays an auxiliary role: it guarantees the count to be at least 3 in every model $\mathcal{I}$ of the KB $\mathcal{K}_{\textit{d-count}} = \langle \mathcal{T}, \mathcal{A} \rangle$, that is, $\mathcal{I} \models q_{\textit{d-count}}(\mathbf{a}_\emptyset, 3)$.

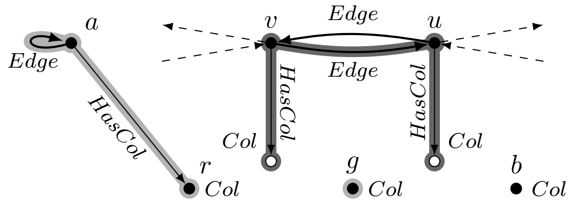

Figure 3: Part of $Can(\mathcal{K}_{\textit{d-count}})$ related to an edge (v, u) of the graph $\mathcal{G}$ with some matches for the core of $q_{\textit{d-count}}(Count())$ highlighted.

Indeed, there are three matches for the core of the query to the part of the canonical model corresponding to the ABox, each of which mapping y_v and y_u to d_a, y_c to d_r, and y'_c to one of d_r, d_g, and d_b. One of these matches, in particular, the match maping y'_c to d_g is highlighted by light grey lines in the figure. Besides these three, for each vertex v there is a match for the core of the query to the canonical model $Can(\mathcal{K}_{\textit{d-count}})$ which maps y_v, y_u and y_c as above, and y'_c to the element connected to d_v by $HasCol^-$. In search of a model with the minimal number of matches we may identify such elements with r, g and b (recall that we generally assume UNA in this paper, that is, interpretations of individual names, such as d_r, d_g, and d_b, cannot be identified; see the discussion after this proof). However, if we do it without care we may introduce new structures which agree with the triangular part of the query, and increase the number of matches, as highlighted by dark grey lines in the figure.

Using these observations, next we formally show that $\mathcal{G}(\mathcal{V}, \mathcal{E})$ has a 3-colouring iff $4 \notin Cert(q_{\textit{d-count}}, \mathbf{a}_\emptyset, \mathcal{K}_{\textit{d-count}})$ (i.e., there is at least four matches for the core query in every model).

($\Leftarrow$) Assume first that 4 does not belong to $Cert(q_{\textit{d-count}}, \mathbf{a}_\emptyset, \mathcal{K}_{\textit{d-count}})$. Thus there exists a model $\mathcal{I}$ of $\mathcal{K}_{\textit{d-count}}$ such that $\mathcal{I} \models q_{\textit{d-count}}(\mathbf{a}_\emptyset, 3)$ (we know from the observation above that the count cannot be any number less than 3).

Since ACQs are monotone, it is safe to assume that for every vertex $v \in \mathcal{V}$ there exists a single element $d \in \mathbb{D}^\mathcal{I}$ such that (d_v, d) in $HasCol^\mathcal{I}$.

Let us define the following colouring $\sigma : \mathcal{V} \to \{\mathsf{red}, \mathsf{green}, \mathsf{blue}\}$ of $\mathcal{G}$: for each vertex $v \in \mathcal{V}$, we set

$$\begin{array}{lll} \sigma(v) = & \mathsf{red} & \text{iff } (d_v, d_r) \in HasCol^\mathcal{I}, \\ \sigma(v) = & \mathsf{green} & \text{iff } (d_v, d_g) \in HasCol^\mathcal{I}, \text{ and} \\ \sigma(v) = & \mathsf{blue} & \text{iff } (d_v, d_b) \in HasCol^\mathcal{I}. \end{array}$$

We now show that σ is indeed a proper 3-colouring.

First, we show that σ assigns a colour to each vertex in $\mathcal{V}$. For the sake of contradiction, assume that it does not hold. Then there must be a vertex v such that $HasCol^\mathcal{I}$ does not contain any of $(d_v, d_r), (d_v, d_g)$, and (d_v, d_b). Since we know that there is $u \in \mathcal{V}$ such that $(d_v, d_u) \in Edge^\mathcal{I}$ (recall, that we assume that $\mathcal{G}$ does not have isolated vertices), and since $\exists Edge \sqsubseteq \exists HasCol$, it follows that there is an element d different from d_r, d_g, and d_b, such that $(d_v, d) \in HasCol^\mathcal{I}$. But then also $d \in Col^\mathcal{I}$ since $\exists HasCol^- \sqsubseteq Col$, and we can construct a fourth match h for the core of $q_{\textit{d-count}}$ to $\mathbb{D}^\mathcal{I}$: $h(y_u) = h(y_v) = d_a$, $h(y_c) = d_r$ and $h(y'_c) = d$. This contradicts the fact that $\mathcal{I} \models q_{\textit{d-count}}(\mathbf{a}_\emptyset, 3)$.

Next we show that σ is indeed a correct colouring. Assume for the sake of contradiction that this is not the case. Then there is an edge $(u, v) \in \mathcal{E}$ such that $\sigma(v) = \sigma(u)$. Without loss of generality let $\sigma(v) = \sigma(u) = \mathsf{red}$. From the definition of σ, it means that the pairs (d_v, d_r) and (d_u, d_r) belong to $HasCol^\mathcal{I}$. We can then construct a fourth match h for the core of $q_{\textit{d-count}}$ in $\mathbb{D}^\mathcal{I}$: $h(y_v) = d_v$, $h(y_u) = d_u$, $h(y_c) = d_r$, and $h(y'_c) = d_r$, which again contradicts the fact that $\mathcal{I} \models q_{\textit{d-count}}(\mathbf{a}_\emptyset, 3)$.

We obtain that σ is a 3-colouring as required.

($\Rightarrow$) Assume that there is a 3-colouring σ of $\mathcal{G}$. To make sure that $4 \notin Cert(q_{\textit{d-count}}, \mathbf{a}_\emptyset, \mathcal{K})$, consider an interpretation $\mathcal{I}$ defined is as follows:

$$\begin{array}{lcl}
\mathbb{D}^{\mathcal{I}} & = & \{d_c \mid c \in \mathcal{V} \cup \{r, g, b, a\}\}, \\
c^{\mathcal{I}} & = & d_c, \text{ for } c \in \mathcal{V} \cup \{r, g, b, a\}, \\
Col^{\mathcal{I}} & = & \{d_r, d_g, d_b\}; \\
Edge^{\mathcal{I}} & = & \{(d_a, d_a)\} \cup \\
 & & \{(d_v, d_u), (d_u, d_v) \mid (v, u) \in \mathcal{E}\}; \\
HasCol^{\mathcal{I}} & = & \{(d_v, d_r) \mid v \in \mathcal{V}, \sigma(v) = \mathsf{red}\} \cup \\
 & & \{(d_v, d_g) \mid v \in \mathcal{V}, \sigma(v) = \mathsf{green}\} \cup \\
 & & \{(d_v, d_b) \mid v \in \mathcal{V}, \sigma(v) = \mathsf{blue}\}.
\end{array}$$

A direct verification shows that $\mathcal{I}$ is a model of $\mathcal{K}_{d\text{-}count}$. Next we show that $\mathcal{I} \models q_{d\text{-}count}(\mathbf{a}_\emptyset, 3)$, We know that there are 3 matching for the core of $q_{d\text{-}count}$ in $\mathbb{D}^{\mathcal{I}}$ which map the variables y_v, y_u to d_a; the variable y_c to d_r; and the variable y_c' to either d_r, or d_b, or d_g. From the definition of $Col^{\mathcal{I}}$ and $HasCol^{\mathcal{I}}$, any other match must send y_v and y_u to some d_v and d_u such that $Edge^{\mathcal{I}}$ contains (d_v, d_u), that is, (v, u) is an edge in $\mathcal{G}$. Hence, if such a forth match exist, then there is an element $d \in \{d_r, d_g, d_b\}$ such that both (d_v, d) and (d_u, d) are in $HasCol^{\mathcal{I}}$, and, therefore, $\sigma(v) = \sigma(u)$. But the last contradicts the fact that σ is a colouring. Hence, $\mathcal{I} \models q_{d\text{-}count}(\mathbf{a}_\emptyset, 3)$ as required. □

Before summarising the results of this section, we make few observations.

First, Lemma 12 continues to hold if one drops the UNA. Indeed, to make the reduction work for the case without the UNA it suffices to guarantee that any pair of individual names from $\{r, g, b\}$ cannot be interpreted by the same element in any model of $\mathcal{K}_{d\text{-}count}$ and that a cannot be interpreted by the same element as any of $v \in \mathcal{V}$. We can do this by extending the ABox by assertions $Red(r)$, $Green(g)$, $Blue(b)$, $Aux(a)$, and $Vertex(v)$ for all $v \in \mathcal{V}$, as well as extending the TBox with negative inclusions for disjointness of the concepts Red, $Green$, and $Blue$ such as $\neg Red \sqsubseteq Green$, and similar inclusions $\neg Aux \sqsubseteq Vertex$, $\neg Vertex \sqsubseteq Aux$.

Second, the proof above makes use of the non-connectivity of the query $q_{d\text{-}count}(Count())$. It is an interesting open problem whether the result holds for connected queries.

The lower bound was shown for *DL-Lite*$_{core}$, while the upper bound holds for any *DL-Lite*$_{\mathcal{R}}$ KB. Since *DL-Lite*$_{\mathcal{R}}$ is more expressive than *DL-Lite*$_{core}$, the lemmas above give us the following complexity result.

Theorem 13. *The problem Count-*Aggregate Certain Answers *$(\mathcal{T}, q)$ is* coNP*-complete in data complexity for DL-Lite$_{\mathcal{X}}$ TBoxes $\mathcal{T}$ with $\mathcal{X} \in \{core, \mathcal{R}\}$.*

Thus the data complexity of count query evaluation rises from P in the relational database case to coNP-complete for *DL-Lite* knowledge bases.

5.2. Count Distinct Queries

In this section we show that the data complexity of computing aggregate certain answers for count distinct queries is the same as for count queries—the problem is coNP-complete.

We again start with the upper bound. The proofs of the following two lemmas are very similar to the proofs of Lemma 10 and Lemma 11, and thus we only summarise the (very few) differences.

Lemma 14. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{\mathcal{R}}$ KB and $q(\mathbf{x}, Cntd(z))$ be a count distinct ACQ. Then there exists a number ℓ depending only on q and $\mathcal{T}$ such that if there is a model $\mathcal{I}_0$ of $\mathcal{K}$ with $\mathcal{I}_0 \models q(\mathbf{a}, n_0)$ for a tuple $\mathbf{a}$ of individual names and a number $n_0 \leq |\mathsf{Ind}| + |\mathcal{T}|$, then there is a model $\mathcal{I}'$ of $\mathcal{K}$ with domain of $O(|\mathsf{Ind}|^\ell)$ elements with $\mathcal{I}' \models q(\mathbf{a}, n')$ for some number $n' \leq n_0$.*

The difference between the proof of this lemma and the proof of Lemma 10 is minor: in this case we include into $\mathbb{D}^*$ the interpretations of all individual names from Ind and the images of the aggregation variable z in $\mathcal{I}_0$ under the matches of the core (but not the images of the existential variables). This leads to the required bound on the size of the domain of $\mathcal{I}'$.

Lemma 15. *Let $\mathcal{T}$ be a fixed DL-Lite$_{\mathcal{R}}$ TBox and $q(\mathbf{x}, Cntd(z))$ be a fixed count distinct ACQ. Checking whether $n \in Cert(q, \mathbf{a}, \langle \mathcal{T}, \mathcal{A} \rangle)$ for an ABox $\mathcal{A}$, tuple of individual names $\mathbf{a}$, and number n can be done in* coNP.

The proof of this lemma goes along the same lines as the proof of Lemma 11 except that we bound n by $|\mathsf{Ind}| + |\mathcal{T}|$ and use Lemma 14 instead of Lemma 10. Similarly to the case of count queries, the proof does not depend on UNA, so the result holds if we drop it.

Next we establish the matching lower bound. The proof of the following lemma is again by a reduction from the complement of the 3-colouring problem. However, the construction is more intricate and requires a separate description.

Lemma 16. *There exist a DL-Lite$_{core}$ TBox $\mathcal{T}$ and a Boolean count distinct ACQ q such that checking whether $n \in Cert(q, \mathbf{a}_\emptyset, \langle\mathcal{T}, \mathcal{A}\rangle)$ for an ABox $\mathcal{A}$ and a number n is* coNP*-hard.*

Proof. Just as in to the proof of Lemma 12 in the previous section we first define a fixed TBox and count distinct query, then explain how to construct an ABox on the base of an instance of the complement of the 3-colouring problem, and finally show that number 4 is a certain answer if and only if the instance has no 3-colouring.

Let *Edge*, *Aux*, and *HasCol* be atomic roles. Fix a *DL-Lite$_{core}$* TBox

$$\mathcal{T} = \{\exists Edge \sqsubseteq \exists HasCol\},$$

intended to assign a colour to every vertex. Fix also a Boolean count distinct ACQ

$$\begin{aligned} q_{d\text{-}cntd}(Cntd(z)) \text{ :- } &\exists y_v, y_u, y_c, y_a \\ Edge(y_v, y_u) \wedge HasCol(y_v, y_c) &\wedge HasCol(y_u, y_c) \\ &\wedge Aux(y_s, y_u) \wedge HasCol(y_s, z). \end{aligned}$$

The graphical representation of this query is given in Figure 4. The triangular part plays a similar role as the triangular part in the count query $q_{d\text{-}count}$ in Lemma 12.

Given an instance $\mathcal{G}(\mathcal{V}, \mathcal{E})$ of the complement of coNP-complete 3-colouring problem, we construct the ABox on its base, as follows.

Consider the set of individual names

$$\begin{aligned} \mathsf{Ind} \;=\; &\{v, v_v, v_u, v_c, v_s, v_z \mid v \in \mathcal{V}\} \cup \\ &\{r, g, b, a, a_v, a_u, a_c\}. \end{aligned}$$

Let $\mathcal{A}$ contain

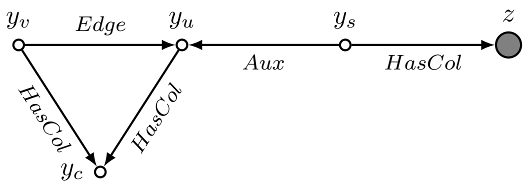

Figure 4: Query $q_{d\text{-}cntd}(Cntd(z))$.

- the assertions $Edge(v, u)$ and $Edge(u, v)$ for each $(v, u) \in \mathcal{E}$;
- the assertions $Aux(v, v_u)$, $HasCol(v_u, v_c)$, $HasCol(v_v, v_c)$, $Edge(v_v, v_u)$, $Aux(v_s, v)$ and $HasCol(v_s, v_z)$ for each vertex v;
- the assertions $Aux(a, a_u)$, $HasCol(a_u, a_c)$, $HasCol(a_v, a_c)$, $Edge(a_v, a_u)$, $HasCol(a, r)$, $HasCol(a, g)$, and $HasCol(a, b)$.

Figure 5 depicts the part of the canonical model of $\mathcal{K}$ that is related to an edge in $\mathcal{G}$. Intuitively, the individual names r, g and b represent colours, and the role *HasCol* is intended to connect vertices in the graph with their colours. Other individual names play auxiliary roles as follows. The individual name a guarantees the count of the images of z to be at least 3 in every model $\mathcal{I}$ of the KB $\mathcal{K}_{d\text{-}cntd} = \langle\mathcal{T}, \mathcal{A}\rangle$, that is, $\mathcal{I} \models q(\mathbf{a}_\emptyset, 3)$ by means of matches mapping the aggregation variable z to one of d_r, d_g, and d_b, the variable y_s to d_a, and the variables in the triangular part of the query to the interpretations of a_v, a_u and a_c. In Figure 5 one of these matches (with z mapped to d_r) is highlighted by thin light grey lines on the left.

Besides these three, for each vertex v there is a match from the core query to the canonical model $Can(\mathcal{K}_{d\text{-}cntd})$ which maps z to the anonymous element connected to d_v by $HasCol^-$, y_s to d_v and the triangular part to such a part corresponding to d_v (this match is depicted by thick light grey lines in the figure). Hence, each of these matches maps z to a separate element, thus increasing the aggregate value for the canonical model. In search of a model with the minimal value we may identify such anonymous elements with d_r, d_g, and d_b. However, if we don't do it with care, we may introduce new structures which match with the triangular part of the query by interpretations of some

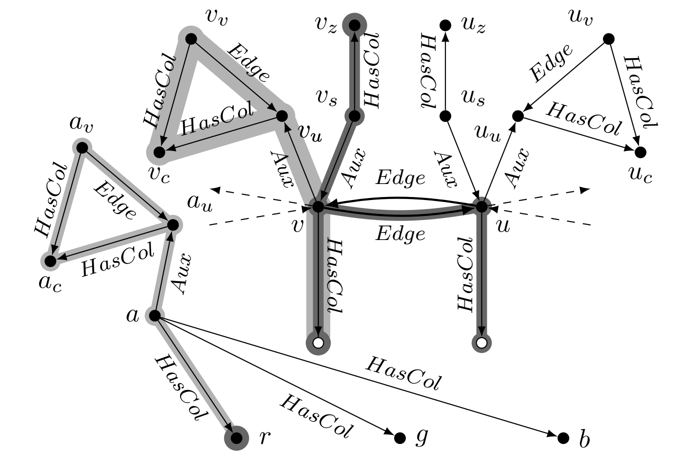

Figure 5: Part of $Can(\mathcal{K}_{d\text{-}cntd})$ related to an edge (v, u) of $\mathcal{G}$ with some matches for the core of $q_{d\text{-}cntd}$ highlighted.

adjacent in $\mathcal{G}$ vertices v, u and a colour vertex, and the rest of the query with, for example, the interpretations of v_s and v_z. In particular, in this case the aggregation variable z is mapped to the interpretation of u_z, which increases the aggregate value. Such a possible match is highlighted in the figure by thin dark grey lines, assuming that the white nodes are identified.

Using these observations, it is possible to formally show that $\mathcal{G}(\mathcal{V}, \mathcal{E})$ has no 3-colouring iff $4 \in Cert(q, \mathbf{t}_\emptyset, \mathcal{K})$ (i.e., for every model there is at least four different images of z in the domain with corresponding matches for the rest of the query). This proof goes the same lines as the second part of the proof of Lemma 12, so we omit it for brevity. □

This lemma again holds for the case when UNA is dropped, and the proof can be adopted in the same way as the proof of Lemma 12.

The following theorem summarises the results of this section.

Theorem 17. *The problem Cntd-*AGGREGATE CERTAIN ANSWERS $(\mathcal{T}, q)$ *is* coNP*-complete in data complexity for TBoxes $\mathcal{T}$ in DL-Lite$_\mathcal{X}$ with $\mathcal{X} \in \{core, \mathcal{R}\}$.*

6. Combined Complexity of Counting Queries

Although data complexity is arguably the most used measure of algorithms in database settings, combined complexity has its own value for understanding fundamental properties of problems. In this section we study the combined complexity of computing aggregate certain answers. Formally, let $\mathcal{X} \in \{core, \mathcal{R}\}$ and f be a counting aggregation function. Now we are interested in the following family of problems.

DL-Lite$_\mathcal{X}$ f-AGGREGATE CERTAIN ANSWERS

Input: *DL-Lite$_\mathcal{X}$* KB $\mathcal{K}$, f ACQ q, tuple $\mathbf{a}$, and number $n \in \mathbb{N}^\infty$.
Question: Does $n \in Cert(q, \mathbf{a}, \mathcal{K})$?

6.1. Count Queries

We start again with count queries. Recall the algorithm to compute the certain answers for count queries explained in the proof of Lemma 11. Note that if one takes into consideration the size of the query and the TBox, then this algorithm naturally gives a coN2ExpTime upper bound; the only difference is that in this case the number of neighbourhoods is of double exponential size (with respect to the size of q and $\mathcal{T}$), and thus the model we need to guess is of double exponential size.

Unfortunately, for the case of *DL-Lite$_\mathcal{R}$*, we are not able either to improve this upper bound or to show that this bound is tight. However, we are able to prove that the problem is coNExpTime-hard. This lower bound is interesting in its own right, because it coincides with the complexity of answering similar queries in other database scenarios that feature incomplete information (see, for example, [25]).

As we see later in this section, the situation is different for the case of *DL-Lite$_{core}$*. Again, we cannot establish tight bounds, but can improve the general algorithm on one exponent, to coNExpTime, and show Π_2^p-hardness.

We start with the coNExpTime-hardness of AGGREGATE CERTAIN ANSWERS for count queries and *DL-Lite$_\mathcal{R}$* ontologies. This lower bound is established by a reduction from the complement of the following version of the tiling problem, that we call here the *NExp-tiling problem.* Its input is a quadruple $(\mathcal{C}, \mathcal{H}, \mathcal{V}, n)$, where $\mathcal{C}$ is a set of colours, $\mathcal{H}, \mathcal{V} \subseteq \mathcal{C} \times \mathcal{C}$ are horizontal and vertical adjacency relations on the colours, and n is a number given in unary; the output is positive if there exist a tiling of a $2^n \times 2^n$ square which use unitary tiles coloured from $\mathcal{C}$ in such a way that each pair of horizontally adjacent tiles satisfies the relation $\mathcal{H}$ and each pair of vertically adjacent tiles satisfies the relation $\mathcal{V}$. This problem was shown to be NExpTime-complete in [26].

Lemma 18. *The decision problem DL-Lite$_\mathcal{R}$ Count-*AGGREGATE CERTAIN ANSWERS *is* coNExpTime-*hard.*

Proof. As mentioned above, the coNExpTime-hardness is established by a reduction from the complement of the NExp-tiling problem. In what follows we first explain how to, given an instance $(\mathcal{C}, \mathcal{H}, \mathcal{V}, n)$ of this problem, construct in polynomial time a *DL-Lite$_\mathcal{R}$* knowledge base $\mathcal{K}_{c\text{-}count}$ and Boolean count query $q_{c\text{-}count}$, and then show that $|\mathcal{C}| + 1 \notin Cert(q_{c\text{-}count}, \mathbf{a}_\emptyset, \mathcal{K}_{c\text{-}count})$ iff the answer for the instance is positive.

Roughly speaking, the idea of the construction is as follows. Each model of $\mathcal{K}_{c\text{-}count}$ represents a possible tiling of the $2^n \times 2^n$ square, and has at least p matches for the core query $\bar{q}_{c\text{-}count}$; in turn, each violation of the adjacent tiles from $\mathcal{H}$ or $\mathcal{V}$ increases the number of matches. In this way, the only possibility for $p+1$ to be not a certain answer is the existence of a model that represents a tiling in which there are no violations.

Let $(\mathcal{C}, \mathcal{H}, \mathcal{V}, n)$ be an instance of the NExp-tiling problem. We start the construction with the vocabulary. It consists of atomic roles

- $H_0^n, \ldots, H_0^1, H_1^n, \ldots, H_1^1$ and $V_0^n, \ldots, V_0^1, V_1^n, \ldots, V_1^1$ used to identify tile positions in the square by horizontal and vertical coordinates represented in binary (reverse numeration $n, \ldots, 1$ of bits is convenient for representation of such binary numbers and is used consistently throughout this proof);
- H and V subsuming the roles above;
- Bit_0 and Bit_1 used to identify the bit values in the coordinates;
- *Bit* subsuming the roles above;
- *Tile* used to connect a position to a particular colour;

and atomic concepts

- *Root* to start the construction;
- *Zero* and *One* for binary values;
- $Colour_m$ for each colour c_m in $\mathcal{C}$; and
- *Colour* subsuming the concepts above.

In what follows we define the set of individual names Ind, ABox $\mathcal{A}$, TBox $\mathcal{T}$ and core query $\bar{q}_{c\text{-}count}$ simultaneously, splitting the description in conceptual parts. The (possibly nested) subqueries of the core query will have form $\phi(\mathbf{y}) = \exists \mathbf{y}'\ \psi(\mathbf{y}, \mathbf{y}')$, and we assume that variables $\mathbf{y}$ are 'globally' existential for the superquery, but the existential variables $\mathbf{y}'$ are 'local' for each subquery, that is in the overall query they should be renamed to names fresh for this subquery (recall that the overall query is Boolean, so all the variables are existential in it).

Let the set Ind contain

- individual names 0 and 1 as the binary values for defining the coordinates of the square,
- all the colours c_m in $\mathcal{C}$ as individual names,

and let the ABox $\mathcal{A}$ contain assertions

- $Zero(0)$ and $One(1)$,
- $Colour(c_m)$ for each colour c_m in $\mathcal{C}$.

Let the core query $\bar{q}_{c\text{-}count}$ contain the subquery

$$\psi'() = \exists y_0, y_1, y_c \quad Zero(y_0) \wedge One(y_1) \wedge Colour(y_c).$$

Note that all the variables in this part of the query are locally existential. Assuming that the rest of the core query has a match in some model of $\mathcal{K}_{c\text{-}count}$, then the overall core query has at least p matches each of which maps y_0 to (the interpretation of) 0; y_1 to 1; and y_c to one of the colours c_m. Moreover, if a model contains some other elements in the interpretations of *Zero*, *One* or *Colour*, then the number of matches is strictly greater that p. Since we want to check whether there is a model with only p matches, in what follows we consider only the models which does not have such other elements.

Let Ind contain individual name

- r as the root of the construction,

and let the ABox $\mathcal{A}$ contain assertions

- $\mathit{Root}(r)$,
- $\mathit{Colour}_m(c_m)$ for each colour c_m in $\mathcal{C}$.

Together with the TBox $\mathcal{T}$ given next, the individual name r generates a tree, each leaf of which represents a position in the square. For all binary values $i, j = 0, 1$ for bits, all $n > k \geq 1$, all $n \geq \ell \geq 1$, and all $1 \leq m \leq p$ the TBox $\mathcal{T}$ contains the inclusions

- $\mathit{Root} \sqsubseteq \exists H_i^n$;
- $\exists (H_i^{k+1})^- \sqsubseteq \exists H_j^k$;
- $\exists (H_i^1)^- \sqsubseteq \exists V_j^n$;
- $\exists (V_i^{k+1})^- \sqsubseteq \exists V_j^k$;
- $H_i^\ell \sqsubseteq H$ and $V_i^\ell \sqsubseteq V$;
- $\exists (H_i^\ell)^- \sqsubseteq \exists \mathit{Bit}_i$ and $\exists (V_i^\ell)^- \sqsubseteq \exists \mathit{Bit}_i$;
- $\exists \mathit{Bit}_0^- \sqsubseteq \mathit{Zero}$ and $\exists \mathit{Bit}_1^- \sqsubseteq \mathit{One}$;
- $\mathit{Bit}_i \sqsubseteq \mathit{Bit}$;
- $\exists (V_i^1)^- \sqsubseteq \exists \mathit{Tile}$;
- $\exists \mathit{Tile}^- \sqsubseteq \mathit{Colour}$.

A (part of a) model of $\mathcal{K}_{\textit{c-count}}$, satisfying the aforementioned restrictions is given on the left of Figure 6. It consists of a tree with r as the root and H's and V's as labels of edges. If the subscript of such a label is 0 then its end has an outgoing edge labelled with Bit_0, which by inclusion $\exists \mathit{Bit}_0^- \sqsubseteq \mathit{Zero}$ and our assumption leads to (the interpretation of) 0. If the subscript is 1, then the outgoing edge is labelled with Bit_1 and leads to 1. By this, each leaf in the tree determines a position in the $2^n \times 2^n$ square, by means of binary representations of horizontal and vertical coordinates along the branch going to the leaf. Each leaf is also connected by Tile to one of the tile colours c_m, according to our assumption (these connections are just sketched).

We continue with the definition of $\bar{q}_{\textit{c-count}}$. As mentioned above, its idea is to capture those

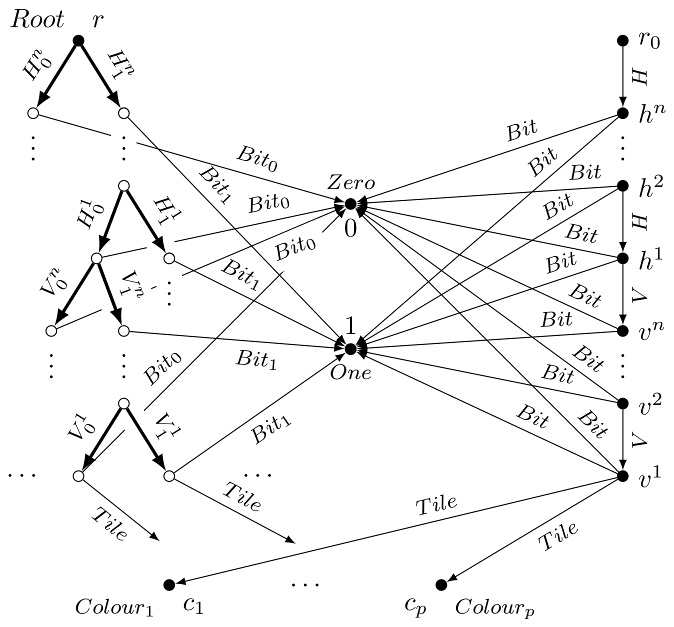

Figure 6: A model of $\mathcal{K}_{\textit{c-count}}$, in which, by means of the subquery ψ', all Bit_0 lead to 0, all Bit_1 lead to 1, and all Tile lead to one of the colours c_m (the ones on the left part are not determined and just sketched). All the indexes of roles are optional (e.g., V_1^n represents also V). The right part is auxiliary and required to guarantee that ψ'' has at least one match in any model.

tilings that have been placed erroneously. To achieve this we make use of several subqueries. For their definition it will be convenient to use the following subquery for variable y_r and tuples of variables $\mathbf{y}_h = y_h^n, \ldots, y_h^1$ and $\mathbf{y}_v = y_v^n, \ldots, y_v^1$:

$$\begin{aligned}
&\phi(y_r, \mathbf{y}_h, \mathbf{y}_v) = \\
&\quad H(y_r, y_h^n) \wedge H(y_h^n, y_h^{n-1}) \wedge \cdots \wedge H(y_h^2, y_h^1) \wedge \\
&\qquad V(y_h^1, y_v^n) \wedge V(y_v^n, y_v^{n-1}) \wedge \cdots \wedge V(y_v^2, y_v^1).
\end{aligned}$$

The subquery $\phi(y_r, \mathbf{y}_h, \mathbf{y}_v)$ matches any branch in the tree of a model of $\mathcal{K}_{\textit{c-count}}$, that is a position in the $2^n \times 2^n$ square.

The next step is to relate all those pairs of leaves that represent horizontally or vertically adjacent positions in the square. In order to do this, we note that the binary representations of horizontally adjacent positions have the form

$$\begin{aligned}
&(w_h \cdot 0 \cdot 1^{k-1})(w_v), \\
&(w_h \cdot 1 \cdot 0^{k-1})(w_v),
\end{aligned} \tag{4}$$

where w_h is a possibly empty binary word of length $n - k - 1$, $n > k \geq 1$, w_v is a binary word of length n, and parenthesis are used to delimit the horizontal and vertical coordinates in the representation.

Using this property we define the subqueries ξ_h^k, $n > k \geq 1$, which are meant to capture such horizontally adjacent positions, for tuples of variables $\mathbf{s}_h = s_h^n, \dots, s_h^{k+1}$ and $\mathbf{s}_v = s_v^n, \dots, s_v^1$:

$$\begin{aligned}\xi_h^k(\mathbf{y}_h, \mathbf{y}_v, \mathbf{u}_h, \mathbf{u}_v, y_0, y_1) = \exists y_r, \mathbf{s}_h, \mathbf{s}_v \\ \phi(y_r, \mathbf{y}_h, \mathbf{y}_v) \wedge \phi(y_r, \mathbf{u}_h, \mathbf{u}_v) \wedge \\ \mathit{Bit}(y_h^n, s_h^n) \wedge \dots \wedge \mathit{Bit}(y_h^{k+1}, s_h^{k+1}) \wedge \\ \mathit{Bit}(y_h^k, y_0) \wedge \mathit{Bit}(y_h^{k-1}, y_1) \wedge \dots \wedge \mathit{Bit}(y_h^1, y_1) \wedge \\ \mathit{Bit}(u_h^n, s_h^n) \wedge \dots \wedge \mathit{Bit}(u_h^{k+1}, s_h^{k+1}) \wedge \\ \mathit{Bit}(u_h^k, y_1) \wedge \mathit{Bit}(u_h^{k-1}, y_0) \wedge \dots \wedge \mathit{Bit}(u_h^1, y_0) \wedge \\ \mathit{Bit}(y_v^n, s_v^n) \wedge \dots \wedge \mathit{Bit}(y_v^1, s_v^1) \wedge \\ \mathit{Bit}(u_v^n, s_v^n) \wedge \dots \wedge \mathit{Bit}(u_v^1, s_v^1).\end{aligned}$$

The structure of this subquery closely corresponds to the binary representations (4) of horizontally adjacent positions—the first $n - k - 1$ variables from $\mathbf{y}_h$ and $\mathbf{u}_h$ by means of the *Bit* role represent an arbitrary but same word w_h (matched by $\mathbf{s}_h$), and the rest of these variables represent $0 \cdot 1^{k-1}$ and $1 \cdot 0^{k-1}$ respectively (as we will see, y_0 should be mapped to 0 and y_1 to 1); similarly, the variables $\mathbf{y}_v$ and $\mathbf{u}_v$ represent a word w_v (matched by $\mathbf{s}_v$), the vertical component of the positions.

The next step is to define subqueries detecting horizontally adjacent positions in which the tiles are not arranged according to $\mathcal{H}$. We do this as follows. For each pair $(c_m, c_\ell) \in (\mathcal{C} \times \mathcal{C}) \setminus \mathcal{H}$, that is an incorrect pair, and each $n > k \geq 1$ we define the query

$$\begin{aligned}\chi_{h,(m,\ell)}^k(y_0, y_1, y_m^c, y_\ell^c) = \exists \mathbf{y}_h, \mathbf{y}_v, \mathbf{u}_h, \mathbf{u}_v \\ \xi_h^k(\mathbf{y}_h, \mathbf{y}_v, \mathbf{u}_h, \mathbf{u}_v, y_0, y_1) \wedge \\ \mathit{Tile}(y_v^1, y_m^c) \wedge \mathit{Tile}(u_v^1, y_\ell^c).\end{aligned}$$

In this subquery the variables y_m^c and y_ℓ^c should be mapped to the (interpretations of the) corresponding colours c_m and c_ℓ. Thus, each $\chi_{h,(m,\ell)}^k$ asks for a horizontal violation of tiles coloured c_m and c_ℓ in the horizontally adjacent positions identified by $\xi_h^k(\mathbf{y}_h, \mathbf{y}_v, \mathbf{u}_h, \mathbf{u}_v, y_0, y_1)$.

Properties, similar to (4) hold for vertically adjacent positions. Hence, we also define subqueries $\chi_{v,(m,\ell)}^k$ for each vertically incorrect pair of colours $(c_m, c_\ell) \in (\mathcal{C} \times \mathcal{C}) \setminus \mathcal{V}$ and each $n > k \geq 1$ on the base of ξ_v^k exactly in the same way as the $\chi_{h,(m,\ell)}^k$ are defined of the base of the ξ_h^k.

We are now in a position to define the main subquery of the core query $\bar{q}_{\textit{c-count}}$, for the tuple of variables $\mathbf{y}^c = y_1^c, \dots, y_p^c$:

$$\begin{aligned}\psi''() = \exists y_0, y_1, \mathbf{y}^c \\ \bigwedge_{(c_m,c_\ell)\in(\mathcal{C}\times\mathcal{C})\setminus\mathcal{H}} \bigwedge_{n>k\geq 1} \chi_{h,(m,\ell)}^k(y_0, y_1, y_m^c, y_\ell^c) \wedge \\ \bigwedge_{(c_m,c_\ell)\in(\mathcal{C}\times\mathcal{C})\setminus\mathcal{V}} \bigwedge_{n>k\geq 1} \chi_{v,(m,\ell)}^k(y_0, y_1, y_m^c, y_\ell^c) \wedge \\ \mathit{Zero}(y_0) \wedge \mathit{One}(y_1) \wedge \\ \mathit{Colour}_1(y_1^c) \wedge \dots \wedge \mathit{Colour}_p(y_p^c).\end{aligned}$$

Finally, the core of query $q_{\textit{c-count}}$ is

$$\bar{q}_{\textit{c-count}} \coloneq \psi'() \wedge \psi''().$$

Recall our convention about fresh variable names in different subqueries—for example, all $\mathbf{y}_h$, $\mathbf{y}_v$, $\mathbf{u}_h$ and $\mathbf{u}_v$ in different subqueries for incorrect pairs are different, as well as y_0 and y_1 are different in ψ' and ψ''.

To complete the construction, we need to define the part of the ABox $\mathcal{A}$ which has a match for ψ'' in any model (this is needed for ψ' to play its role of sending ends of Bit_i to 0 and 1 respectively and ends of *Tile* to colours). The set Ind contains individual names

- r_0, $h^n, \dots, h^1$, $v^n, \dots, v^1$,

and the ABox $\mathcal{A}$ for each $i = 0, 1$, each $n > k \geq 1$, each $n \geq \ell \geq 1$, and each $1 \leq m \leq p$ contains the assertions

- $H(r_0, h^n)$, $H(h^{k+1}, h^k)$,
- $V(h^1, v^n)$, $V(v^{k+1}, v^k)$,
- $\mathit{Bit}(h^\ell, i)$, $\mathit{Bit}(v^\ell, i)$,
- $\mathit{Tile}(v^1, c_m)$.

The graphical representation of the part of a model corresponding to this part of ABox is given on the right of Figure 6. It indeed works as expected: there are exactly p matches for the overall $\bar{q}_{\textit{c-count}}$ in any model, all of them mapping ψ'' to this part, y_0 and y_1 from ψ' to (the interpretations of) 0 and 1 respectively, and y_c from ψ' to one of $c_1, \ldots, c_p$.

Having the construction of the KB $\mathcal{K}_{c\text{-}count}$ and query $q_{c\text{-}count}$ completed, next we formally show the correctness of the reduction. Using the intuition of the construction, we need to show that there is a solution to the instance of the NExp-tiling problem if and only if there is a model $\mathcal{I}$ of $\mathcal{K}_{c\text{-}count}$ with no matches from $\bar{q}_{c\text{-}count}$ to $\mathcal{I}$, except those p.

($\Leftarrow$) Assume that there is no solution of the instance $(\mathcal{C}, \mathcal{H}, \mathcal{V}, n)$ of the NExp-tiling problem, but yet, for the sake of contradiction, there is a model $\mathcal{I}$ of $\mathcal{K}_{c\text{-}count}$ with only those p matches from $\bar{q}_{c\text{-}count}$ to $\mathcal{I}$. Without loss of generality we assume that $\mathcal{I}$ is minimal in the sense that there is no any proper submodel of $\mathcal{I}$ with this property. Note that $\mathcal{I}$ possesses the following properties:

1. the interpretations of relations $Zero$ and One contain only the interpretations of 0 and 1, respectively;
2. interpretations of each $Colour_m$ contains only the interpretation of the individual name c_m, for $c_m \in \mathcal{C}$, and the interpretation of $Colour$ contains all these elements, but nothing else.

Indeed, by the construction of $\bar{q}_{c\text{-}count}$, it is clear that a violation of any of these properties immediately results in more matches for this query in $\mathcal{I}$.

Consider the canonical model of $\mathcal{K}_{c\text{-}count}$. As noted, the part connected to the element d_r in the interpretation of $Root$ (we know there is only one such element because of the minimality of $\mathcal{I}$) forms a tree on interpretations of the roles H and V, each node of which, except the root, has a Bit-successor and each leaf has a $Tile$-successor. We also know that there is a homomorphism from the canonical model to $\mathcal{I}$, and, since $\mathcal{I}$ is minimal, this homomorphism is surjective. Hence, in $\mathcal{I}$ the ends of Bit_0 are (the interpretation of) 0, the ends of Bit_1 are 1, and the ends of $Tile$ are among colours. We can construct a tiling for a $2^n \times 2^n$ square as follows. The image of each branch in the tree identifies coordinates of a position in the square in binary, by means of the ends of the Bit-connections. In this position the tiling has a tile of the colour c_m whose interpretation is connected by $Tile^-$ with the homomorphic image of the leaf of the branch (this c_m is unique since $\mathcal{I}$ is minimal). By the construction, it must be the case that this is indeed a correct tiling, because any violation would result in an additional match for one of the subqueries $\chi^k_{h,(m,\ell)}$ or $\chi^k_{v,(m,\ell)}$, which would lead to more than p matches for the core query $\bar{q}_{c\text{-}count}$ in $\mathcal{I}$, contradicting the assumption.

($\Rightarrow$) Assume that there is a solution to the instance $(\mathcal{C}, \mathcal{H}, \mathcal{V}, n)$. Consider the model $\mathcal{I}$ of $\mathcal{K}_{c\text{-}count}$ which is the image of a homomorphism from the canonical model mapping

- all ends of Bit_0 to the interpretation of 0,
- all ends of Bit_1 to the interpretation of 1,
- all ends of $Tile$ for the leaf of a branch in the tree to the interpretation of the colour of the tile in the position identified in binary by the branch, and
- all other elements to themselves.

It follows from the construction of $\bar{q}_{c\text{-}count}$ that no match exists which maps the variables of this query to some elements in this tree, that is there are exactly p matches for $\bar{q}_{c\text{-}count}$ in $\mathcal{I}$. □

Having the results for count queries over more expressive *DL-Lite*$_\mathcal{R}$, we proceed to *DL-Lite*$_{core}$.

As said in the beginning of this section, in this case we are able to improve the general algorithm and establish a coNExpTime upper bound. The proof is an adaptation of the algorithm in the proof of Lemma 11. Recall that this proof shows that there is always a model of double-exponential size, with respect to all of TBox, query and ABox, that witnesses that a certain n belongs to the aggregate certain answers of a query. For the case of *DL-Lite*$_{core}$ KBs, we are able to show that one can always find such a model of exponential size. This is formalised in the following lemma.

Lemma 19. *There exists a fixed polynomial P such that for every DL-Lite$_{core}$ KB over a set of individual names* Ind *and count ACQ $q(\mathbf{x}, Count())$ if there is a model $\mathcal{I}_0$ of $\mathcal{K}$ with $\mathcal{I}_0 \models q(\mathbf{a}, n_0)$ for a number $n_0 \leq (|\mathsf{Ind}| + |\mathcal{T}|)^{|q|}$, then there is a model $\mathcal{I}'$ over $O(|\mathsf{Ind}|^{P(|q|)})$ elements with $\mathcal{I}' \models q(\mathbf{a}, n')$ for some number $n' \leq n_0$.*

Proof. Let $\mathbb{D}^*$ be all elements of $\mathbb{D}^{\mathcal{I}_0}$ which are either interpretations of the individual names from Ind or images of variables by matches for the core of q in $\mathbb{D}^{\mathcal{I}_0}$. Consider the interleaving $\mathcal{I}_0^{\mathbb{D}^*}$ of $\mathcal{I}_0$ for $\mathbb{D}^*$.

The proof goes along the same lines as the proof for Lemma 10, as we construct from $\mathcal{I}_0^{\mathbb{D}^*}$ a much smaller model by merging elements with similar neighbourhoods. As a base for the neighbourhood for an element d, we could still consider all elements connected to d by paths of length $|q|$ or less. Though, from the properties of *DL-Lite*$_{core}$ one can show that it is safe to consider the equivalence relation which is much more general than for *DL-Lite*$_{\mathcal{R}}$ knowledge bases, which ends up in an exponential increase in the number of elements that can be merged. This is justified by the following property.

First, note that since $\mathcal{K}$ is a *DL-Lite*$_{core}$ knowledge base, every element d in in the domain of the interleaving that is not in $\mathbb{D}^*$ has the following property: for any atomic role or inverse of a role R there exists at most one element d' with (d, d') in the interpretation of R.

Consider now the body $\phi(\mathbf{x}, \mathbf{y})$ of q, and let d be an element in the domain of the interleaving outside $\mathbb{D}^*$. For ease of explanation assume that q is Boolean (i.e., $\mathbf{x}$ is empty) and the $|q|$-neighbourhood $\mathcal{N}_{|q|}(d)$ of d contains only elements outside $\mathbb{D}^*$. Further, assume that there is a match h for q in $\mathcal{N}_{|q|}(d)$ such that d is in the image of h, say, $d = h(y)$ for some y from $\mathbf{y}$. By the properties of canonical models that we have mentioned above, it must be the case that all matches for q in $\mathcal{N}_{|q|}(d)$ mapping y to d (including h) come from automorphisms in q, that is, are such that the subinterpretations of $\mathcal{N}_{|q|}(d)$ induced by them are exactly the same.

Using the property above, we consider the following notion instead of the notion of k-neighbourhood in Lemma 10. A *subquery* q' of the core $\bar{q}$ of q is a CQ whose body is a conjunction of (not necessary all) atoms in the body of $\bar{q}$. Given an element d in the domain of $\mathcal{I}_0^{\mathbb{D}^*}$ which is not in $\mathbb{D}^*$, the *q-neighbourhood* $\mathcal{N}_q^*(d)$ of d is the set of all domain elements d' such that there exist $d_0, \dots, d_i$, $i \leq k$, with

- $d_0 = d$, $d_i = d'$,
- $d_j \notin \mathbb{D}^*$ for all j, $0 \leq j < i$,
- there exists a positive role R_j (that is, an atomic role or its inverse) such that (d_j, d_{j+1}) is in the interpretation of R_j under $\mathcal{I}_0^{\mathbb{D}^*}$ for all $0 \leq j < i$,
- there is a match for a subquery q' of $\bar{q}$ and $\mathbf{a}$ in $\mathcal{I}_0^{\mathbb{D}^*}$ that has all $R_j(d_j, d_{j+1})$ above in the image.

Having this definition, the rest of the proof is the same as the proof of Lemma 10. In particular, the *root* of $\mathcal{N}_q^*(d)$ is the element d_w of the canonical model such that $f'(d_w) \in \mathcal{N}_q^*(d)$ and w is a prefix of w' for any element $d_{w'}$ with $f'(d_{w'}) \in \mathcal{N}_q^*(d)$ (recall that f' is the function from the definition of the interleaving given in (3)). Then, exactly the same as before, we define the following equivalence relation on elements in the domain of $\mathcal{I}_0^{\mathbb{D}^*}$ which are not in $\mathbb{D}^*$: $d \sim_{\mathbb{D}^*}^q d'$ for d and d' with roots d_w and $d'_{w'}$ if and only if

1. for any word w_1
   (a) $f'(d_{ww_1}) \in \mathcal{N}_k(d)$ iff $f'(d_{w'w_1}) \in \mathcal{N}_k(d')$,
   (b) $f'(d_{ww_1}) \in \mathcal{N}_k(d) \cap \mathbb{D}^*$ iff $f'(d_{w'w_1}) \in \mathcal{N}_k(d') \cap \mathbb{D}^*$;
2. $|w| \equiv |w'| \pmod{2k+1}$, that is, $|w|$ and $|w'|$ are congruent modulo $2k + 1$.

Once again the model obtained from $\mathcal{I}_0^{\mathbb{D}^*}$ by identifying the elements d, d' such that $d \sim_{\mathbb{D}^*}^q d'$ does not create new matches for the core and $\mathbf{a}$ in the resulting model $\mathcal{I}_0$. Moreover, from the construction of q-neighbourhoods the model $\mathcal{I}_0$ has $O(\mathbb{D}|^{p(|q|)})$ underlying elements. This finishes the proof of the lemma. □

Using this lemma it is now straightforward to show our upper bound. The idea of the algorithm, once again, is to guess such an instance $\mathcal{I}_0$, and show that the number of matches for $\bar{q}$ and $\mathbf{a}$ in $\mathcal{I}_0$ corresponds to the desired number.

Lemma 20. *The problem DL-Lite$_{core}$ Count-*Aggregate Certain Answers *is in* coNExpTime.

The coNExpTime reduction shown in Lemma 18 uses role inclusions in the TBox, that is, it is applicable only to *DL-Lite*$_\mathcal{R}$, but not to *DL-Lite*$_{core}$. We can show a Π_2^p lower bound using a reduction that is very similar to that of Lemma 24 in the following section. Thus, for space reasons we omit the formal proof of this result. We have the summarizing theorem.

Theorem 21.
(1) The problem DL-Lite$_{core}$ *Count*-AGGREGATE CERTAIN ANSWERS *is in* coNExpTime *and* Π_2^p-*hard.*
(2) The problem DL-Lite$_\mathcal{R}$ *Count*-AGGREGATE CERTAIN ANSWERS *is in* coN2ExpTime *and* coNExpTime-*hard.*

6.2. Count Distinct Queries

In Section 5 we adapt the algorithm from Lemma 11 on count queries to an algorithm in Lemma 15 on count distinct queries. This algorithm naturally gives us a coN2ExpTime upper bound for the combined complexity of count distinct query answering. Similarly to the count queries we can neither improve this bound nor show that it is tight. However, again, we show coNExpTime-hardness of this problem for *DL-Lite*$_\mathcal{R}$-ontologies. In the case of *DL-Lite*$_{core}$ the situation is again similar to count case: we can improve, in exactly the same way, the algorithm by one exponent to coNExpTime, and show Π_2^p hardness of the problem.

Let us begin with the general case for *DL-Lite*$_\mathcal{R}$ knowledge bases. We use a reduction similar to the one in the proof of Lemma 18.

Lemma 22. *The decision problem DL-Lite*$_\mathcal{R}$ *Cntd*-AGGREGATE CERTAIN ANSWERS *is* coNExpTime-*hard.*

Proof. As mentioned above, the coNExpTime-hardness is established by a reduction from the complement of the NExp-tiling problem. In what follows, first we explain how to, given an instance $(\mathcal{C}, \mathcal{H}, \mathcal{V}, n)$ of this problem, construct in polynomial time a *DL-Lite*$_\mathcal{R}$ knowledge base $\mathcal{K}^\mathcal{R}_{c\text{-}cntd}$ and Boolean count distinct aggregate query $q^\mathcal{R}_{c\text{-}cntd}$; and then we show that $p + 3 \notin Cert(q^\mathcal{R}_{c\text{-}cntd}, \mathbf{a}_\emptyset, \mathcal{K}^\mathcal{R}_{c\text{-}cntd})$ for the empty tuple $\mathbf{a}_\emptyset$ of individual names and $p = |\mathcal{C}|$ iff the answer to the instance is positive.

The underlying idea of the reduction is the same as in the proof of Lemma 18. However, this proof is much more technical, since dealing with count distinct instead of count queries creates a series of technical complications.

More precisely, recall that in the proof of Lemma 18 we built a knowledge base such that some of its models represent different possibilities of tiling the $2^n \times 2^n$ square. The knowledge base, and in particular the ABox, was constructed in a way that its models contain (a) a tree identifying tile colours in the positions in the square and (b) a linear structure, which guarantees that every model has at least p matches for the core query (see Figure 6). The query consisted of several disconnected subqueries, which look for (a) models which are not tilings, that is, for example, models in which some positions in the square are not defined, or tile colours are not assigned to some positions, and (b) incorrect tilings, that is those that have violations of horizontally or vertically adjacent tiles from $\mathcal{H}$ and $\mathcal{V}$. By this, a model representing a tiling which is correct has exactly p matches for the core query, and otherwise all the models have at least $p + 1$ matches.

The main technical difficulty of this reduction is that all conjuncts in the count distinct query $q^\mathcal{R}_{c\text{-}cntd}(z)$ have to be connected to the single aggregation variable z which accomplishes all the tasks above.

Let $(\mathcal{C}, \mathcal{H}, \mathcal{V}, n)$ be an instance of the NExp-tiling problem. The vocabulary of $\mathcal{K}^\mathcal{R}_{c\text{-}cntd}$ slightly extends the vocabulary of $\mathcal{K}_{c\text{-}count}$ from Lemma 18: it contains atomic roles H_i^ℓ and V_i^ℓ for $i = 1, 2$ and $n \geq \ell \geq 1$; H and V; Bit_0 and Bit_1; Bit; and $Tile$; as well as atomic concepts $Root$; $Zero$ and One; $Colour_m$ for $1 \leq m \leq p$; and $Colour$, with the same intended meaning as in that proof, and new atomic roles

- Aux to connect subqueries,
- HV which subsumes H and V, and
- $TileBit$ which subsumes $Tile$ and Bit.

The last two roles are required, because some parts of the query need to match conceptually different parts of models. We will see it in more detail later.

In what follows we show how to construct the TBox, query, and ABox. We do it in this particular order, because the first two have minor differences with the construction in Lemma 18 and can be seen as a basis for understanding the construction, but the last one is more complicated.

So, we start the construction with the TBox $\mathcal{T}$. It is almost the same as the TBox in the proof of Lemma 18. We recall the common part here for completeness. For each $i, j = 0, 1$, each $n > k \geq 1$, each $n \geq \ell \geq 1$, and each $1 \leq m \leq p$ the TBox $\mathcal{T}$ contains the inclusions

- $\mathit{Root} \sqsubseteq \exists H_i^n$;
- $\exists (H_i^{k+1})^- \sqsubseteq \exists H_j^k$;
- $\exists (H_i^1)^- \sqsubseteq \exists V_j^n$;
- $\exists (V_i^{k+1})^- \sqsubseteq \exists V_j^k$;
- $H_i^\ell \sqsubseteq H$ and $V_i^\ell \sqsubseteq V$;
- $\exists (H_i^\ell)^- \sqsubseteq \exists \mathit{Bit}_i$ and $\exists (V_i^\ell)^- \sqsubseteq \exists \mathit{Bit}_i$;
- $\exists \mathit{Bit}_0^- \sqsubseteq \mathit{Zero}$ and $\exists \mathit{Bit}_1^- \sqsubseteq \mathit{One}$;
- $\mathit{Bit}_i \sqsubseteq \mathit{Bit}$;
- $\exists (V_i^1)^- \sqsubseteq \exists \mathit{Tile}$;
- $\exists \mathit{Tile}^- \sqsubseteq \mathit{Colour}$.

In addition, we extend $\mathcal{T}$ with

- inclusions $\mathit{Tile} \sqsubseteq \mathit{TileBit}$ and $\mathit{Bit} \sqsubseteq \mathit{TileBit}$;
- inclusions $H \sqsubseteq HV$ and $V \sqsubseteq HV$;
- negative inclusions stating that $\mathit{Colour}_1, \ldots, \mathit{Colour}_m$, Zero and One are pairwise disjoint;
- negative inclusions stating that Colour is disjoint from Zero and One.

The positive inclusions are already announced above. The negative inclusions guarantee, that each colour element is 'coloured' in only one colour, that a tiling does not put binary numbers instead of tiles in square positions and so on.

We continue with the query $q^{\mathcal{R}}_{\textit{c-cntd}}(z)$ and do it in the same way as in Lemma 18. In particular, we split it into subqueries, and adopt the convention on 'local' (listed after $\exists$ before the body of subquery) and 'global' (listed as parameters of subqueries) existential variables in these subqueries—the first ones should be renamed to fresh names when combining the subqueries.

Similar to $q_{\textit{c-count}}$, the query $q^{\mathcal{R}}_{\textit{c-cntd}}$ consists of two subqueries, ψ' and ψ'', fulfilling the same tasks. However, it is more convenient to start the definition with the second of them.

The base building block of the query is exactly the same as in Lemma 18. It is the following subquery, for variable y_r and tuples of variables $\mathbf{y}_h = y_h^n, \ldots, y_h^1$ and $\mathbf{y}_v = y_v^n, \ldots, y_v^1$:

$$\begin{aligned}\phi(y_r, \mathbf{y}_h, \mathbf{y}_v) = {}&\\ H(y_r, y_h^n) \wedge H(y_h^n, y_h^{n-1}) \wedge \cdots \wedge H(y_h^2, y_h^1) \wedge&\\ V(y_h^1, y_v^n) \wedge V(y_v^n, y_v^{n-1}) \wedge \cdots \wedge V(y_v^2, y_v^1).&\end{aligned}$$

Its intention is to match any branch in the tree of a model, that is a position in the $2^n \times 2^n$ square.

The next level of queries is again very similar to the construction in Lemma 18. The only difference in the following subqueries ξ_h^k, $n > k \geq 1$, capturing horizontally adjacent positions on the base of representations (4), is that y_r is 'global'. Formally, for $n > k \geq 1$ and tuples of variables $\mathbf{s}_h = s_h^n, \ldots, s_h^{k+1}$ and $\mathbf{s}_v = s_v^n, \ldots, s_v^1$,

$$\begin{gathered}\xi_h^k(y_r, \mathbf{y}_h, \mathbf{y}_v, \mathbf{u}_h, \mathbf{u}_v, y_0, y_1) = \exists \mathbf{s}_h, \mathbf{s}_v\\ \phi(y_r, \mathbf{y}_h, \mathbf{y}_v) \wedge \phi(y_r, \mathbf{u}_h, \mathbf{u}_v) \wedge\\ \mathit{Bit}(y_h^n, s_h^n) \wedge \cdots \wedge \mathit{Bit}(y_h^{k+1}, s_h^{k+1}) \wedge\\ \mathit{Bit}(y_h^k, y_0) \wedge \mathit{Bit}(y_h^{k-1}, y_1) \wedge \cdots \wedge \mathit{Bit}(y_h^1, y_1) \wedge\\ \mathit{Bit}(u_h^n, s_h^n) \wedge \cdots \wedge \mathit{Bit}(u_h^{k+1}, s_h^{k+1}) \wedge\\ \mathit{Bit}(u_h^k, y_1) \wedge \mathit{Bit}(u_h^{k-1}, y_0) \wedge \cdots \wedge \mathit{Bit}(u_h^1, y_0) \wedge\\ \mathit{Bit}(y_v^n, s_v^n) \wedge \cdots \wedge \mathit{Bit}(y_v^1, s_v^1) \wedge\\ \mathit{Bit}(u_v^n, s_v^n) \wedge \cdots \wedge \mathit{Bit}(u_v^1, s_v^1).\end{gathered}$$

Similar modification is done to the subqueries matching violations from $\mathcal{H}$. In particular, for each pair $(c_m, c_\ell) \in (\mathcal{C} \times \mathcal{C}) \setminus \mathcal{H}$, that is an incorrect pair of colours, and each $n > k \geq 1$ we define the query

$$\begin{aligned}\chi^k_{h,(m,\ell)}(y_{aux}, y_0, y_1, y_m^c, y_\ell^c) = \exists y_r, \mathbf{y}_h, \mathbf{y}_v, \mathbf{u}_h, \mathbf{u}_v&\\ \mathit{Aux}(y_{aux}, y_r) \wedge \xi_h^k(y_r, \mathbf{y}_h, \mathbf{y}_v, \mathbf{u}_h, \mathbf{u}_v, y_0, y_1) \wedge&\\ \mathit{Tile}(y_v^1, y_m^c) \wedge \mathit{Tile}(u_v^1, y_\ell^c).&\end{aligned}$$

Note that here we use a new 'global' variable y_{aux} to join by role Aux the roots y_r of all such subqueries, as we see below.

As before, we define subqueries $\chi^k_{v,(m,\ell)}$ for each vertically incorrect pair of colours $(c_m, c_\ell) \in (\mathcal{C} \times \mathcal{C}) \setminus \mathcal{V}$ and each $n > k \geq 1$ on the base of ξ^k_v exactly in the same way as $\chi^k_{h,(m,\ell)}$.

We are now in a position to define the main subquery of the core query $\bar{q}^{\mathcal{R}}_{\textit{c-cntd}}(z)$, for the tuple of variables $\mathbf{y}^c = y^c_1, \dots, y^c_p$:

$$\begin{aligned}
\psi''(y_{aux}) = \exists y_0, y_1, \mathbf{y}^c \\
\bigwedge_{(c_m,c_\ell)\in(\mathcal{C}\times\mathcal{C})\setminus\mathcal{H}} \ \bigwedge_{n>k\geq 1} \chi^k_{h,(m,\ell)}(y_{aux}, y_0, y_1, y^c_m, y^c_\ell) \wedge \\
\bigwedge_{(c_m,c_\ell)\in(\mathcal{C}\times\mathcal{C})\setminus\mathcal{V}} \ \bigwedge_{n>k\geq 1} \chi^k_{v,(m,\ell)}(y_{aux}, y_0, y_1, y^c_m, y^c_\ell) \wedge \\
\mathit{Zero}(y_0) \wedge \mathit{One}(y_1) \wedge \\
\mathit{Colour}_1(y^c_1) \wedge \cdots \wedge \mathit{Colour}_p(y^c_p).
\end{aligned}$$

Finally, the core of the query $q^{\mathcal{R}}_{\textit{c-cntd}}(z)$ is

$$\bar{q}^{\mathcal{R}}_{\textit{c-cntd}}(z) \text{ :- } \exists y_{aux}\ \psi'(z, y_{aux}) \wedge \psi''(y_{aux}),$$

where for tuples of variables $\mathbf{y}_h = y^n_h, \dots, y^1_h$ and $\mathbf{y}_v = y^n_v, \dots, y^1_v$,

$$\begin{aligned}
\psi'(z, y_{aux}) = \exists \mathbf{y}_h, \mathbf{y}_v \\
H(y_{aux}, y^n_h) \wedge H(y^n_h, y^{n-1}_h) \wedge \cdots \wedge H(y^2_h, y^1_h) \wedge \\
HV(y^1_h, y^n_v) \wedge HV(y^n_v, y^{n-1}_v) \wedge \cdots \wedge HV(y^2_v, y^1_v) \wedge \\
\mathit{TileBit}(y^1_v, z).
\end{aligned}$$

The subquery ψ' is similar to ϕ, except that it matches both of the roles H and V in the second half (by means of the role HV subsuming both H and V), and requires $\mathit{TileBit}$ in the end. As we see later, even if ψ' has quite a different form from the one in Lemma 18, it still accomplishes the same task to guarantee that each branch is indeed a binary representation of a position in the square and has an assigned tile colour.

While the query and TBox are almost the same as in the previous proof, we need to adapt the ABox considerably.

We separate the definition into two conceptual parts. The first part is required to guarantee that every model has at least $p + 2$ witnesses of the aggregate variable, which are (the interpretations of) 0, 1, and the colours, and that every model with just those witnesses must represent a tiling, in the same way as in Lemma 18. The second part increases the number of witnesses for z in each tiling violating $\mathcal{H}$ or $\mathcal{V}$. So, overall, there is a model with only $p + 2$ witnesses if and only if there is a correct tiling.

Part 1. We start this part with assertions identifying binary constants and colours. Let Ind contain

- individual names 0 and 1 for binary coordinates,
- all the colours c_m in $\mathcal{C}$ as individual names,

and let the ABox $\mathcal{A}$ contain assertions

- $\mathit{Zero}(0)$ and $\mathit{One}(1)$,
- $\mathit{Colour}(c_m)$ and $\mathit{Colour}_m(c_m)$ for each colour c_m in $\mathcal{C}$.

Next we guarantee that the interpretations of these individual names witness the aggregate variable z, so in all potential models with just $p + 2$ witnesses all other witnessing elements are identified with these. For example, leaves in the position tree should be indeed colours. The set Ind contains individual names

- r_0, $h^n, \dots, h^1$, $v^n, \dots, v^1$,
- r_{aux}, $h^n_{aux}, \dots, h^1_{aux}$, $v^n_{aux}, \dots, v^1_{aux}$,

and the ABox $\mathcal{A}$ for each $i = 0, 1$, each $n > k \geq 1$, each $n \geq \ell \geq 1$, and each $1 \leq m \leq p$ contains the assertions

- $H(r_0, h^n)$, $H(h^{k+1}, h^k)$,
- $V(h^1, v^n)$, $V(v^{k+1}, v^k)$,
- $\mathit{Bit}(h^\ell, i)$, $\mathit{Bit}(v^\ell, i)$,
- $\mathit{Tile}(v^1, c_m)$,
- $\mathit{Aux}(r_{aux}, r_0)$,
- $H(r_{aux}, h^n_{aux})$, $H(h^{k+1}_{aux}, h^k_{aux})$,
- $V(h^1_{aux}, v^n_{aux})$, $V(v^{k+1}_{aux}, v^k_{aux})$,
- $\mathit{TileBit}(v^1_{aux}, c_m)$,
- $\mathit{TileBit}(v^1_{aux}, 0)$, and $\mathit{TileBit}(v^1_{aux}, 1)$.

This part of ABox indeed performs its task—all subqueries of ψ'' are mapped to (the part of the model corresponding to) the assertions of the first five items, all atoms of ψ' but last mapped to the next two, but this last atom of ψ', that is

$\mathit{TileBit}(y^1_v, z)$ has exactly $p+2$ options, mapping z to either binary values or colours. Note that here we use the fact that by the TBox *TileBit* matches both *Tile* and *Bit*, as well as *HV* matches both H and V.

Next step is to generate a tree of positions, same as in Lemma 18. The set Ind contains

- the root individual name r,

and the ABox $\mathcal{A}$ contains assertion

- $\mathit{Root}(r)$.

In the canonical model the *Colour*-leaves of this tree are anonymous elements. However, we want them to be identified with colours. To this end, we enforce them to match the aggregate variable z, by this guaranteeing that in a potential model with only $p+2$ witnesses they are identified with the (interpretations of) the colours (note that they cannot be identified with 0 or 1 because *Colour* is disjoint with *Zero* and *One*). In fact, for each leaf the subquery ψ' is already matched by the branch in the tree ending with this leaf, and, moreover, all the atoms of the subquery ψ'', except *Aux*, are matched by the interpretation of the part of the ABox on h^k and v^k, constructed above. So we just need to connect these two parts by the role *Aux*. The ABox $\mathcal{A}$ contains assertion

- $\mathit{Aux}(r, r_0)$.

The last thing left in Part 1 is to guarantee that *Zero*-ends of Bit_0 are indeed (the interpetation of) 0 and *One*-ends of Bit_1 are 1 in all the intermediate nodes of the tree, that is that each branch indeed represents a position in the square. We can do it in the same way as for colours, by enforcing them to be witnesses of z. It guarantees that in a model with $p+2$ witnesses these ends are identified with appropriate binary constants (note that, due to the disjointness inclusions, identifications with colours and wrong constants are not possible). In fact, the *Bit*'s starting in the ends of V^1_i, that is, the last level of the tree, already have such matches, almost the same as for colours. However, for other levels we need an extra construction. The set Ind contains individual names

- $r^{2n}, \dots, r^1$,

and the ABox $\mathcal{A}$ for each $2n > k \geq 1$ and each $2n \geq \ell \geq 1$ contains assertions

- $H(r^{k+1}, r^k)$, $H(r^1, r)$,
- $\mathit{Aux}(r^\ell, r_0)$.

These assertions indeed perform their task—the ψ'' subquery is matched in the same way as for colours, the lower part of ψ' is matched by the branch of the tree, but the rest of ψ', depending on the level checked, is matched by the interpretation of this extra ABox.

Part 2. The aim of this part of ABox is to check that each model representing a tiling which is incorrect, that is, violates either $\mathcal{H}$ or $\mathcal{V}$, has an element witnessing z, different from the interpretations of colours and binary constants. The number of assertions in this part is quite large, so we do not write all of them explicitly, but instead describe in detail how to construct them.

Each horizontal violation $(c_m, c_\ell) \notin \mathcal{H}$ in adjacent square positions identified by a k-bit in the binary representation is witnessed by the subquery $\chi^k_{h,(m,\ell)}(y_{aux}, y_0, y_1, y^c_m, y^c_\ell)$. This did not require more construction in the proof of Lemma 18, because this subquery was essentially disconnected from the rest of the query. However, now it is not the case, because y_{aux} is a common variable with other such subqueries. To this end, for each $\chi^k_{h,(m,\ell)}$ we may construct ABox assertions which match all *other* such subqueries, as well as the subquery ψ'. So, the element witnessing z from ψ' increases the aggregation value if there is a violation detected by $\chi^k_{h,(m,\ell)}$ in the main tree of positions. In fact, such an independent construction for each $\chi^k_{h,(m,\ell)}$ is not necessary, and most of assertions can be shared among them.

We start with the ABox on fresh individual names isomorphic to the body of the query $q^{\mathcal{R}}_{\textit{c-cntd}}(z)$. In particular, let Ind contain a fresh individual name e_y for each variable y (including an individual name e_z for the aggregation variable z), and $\mathcal{A}$ contain the assertion $R(e_y, e_{y'})$ for each binary atom $R(y, y')$, and the assertion $A(e_y)$ for each unary atom $A(y)$. Next we do several modifications in it.

1. Replace in all the assertions the individual names e_{y_0} and e_{y_1}, that is the names corresponding to variables for binary values, with 0 and 1, accordingly. Also replace the individual names $e_{y^c_m}$, corresponding to the colour variables, with the colours c_m.
2. Remove the individual name $e_{y_{aux}}$, that corresponds to the common variable y_{aux} of all the subqueries χ and ϕ', together with all assertions with this name.
3. For each subquery $\chi^k_{h,(m,\ell)}$ (or, similarly, $\chi^k_{v,(m,\ell)}$), detecting horizontal (or vertical) violation, introduce a fresh individual name e in Ind and add to $\mathcal{A}$ the assertions
   - $Aux(e, e_{y_r})$, where y_r is the root variable of $\chi^k_{h,(m,\ell)}$,
   - $Aux(e, e_{y'_r})$, for the root variable y'_r of each $\chi^{k'}_{h,(m',\ell')}$ and each $\chi^{k'}_{v,(m',\ell')}$ different from $\chi^k_{h,(m,\ell)}$,
   - $H(e, e_{y^n_h})$, where y^n_h is the first 'local' existential variable of ψ'.

This modification essentially replaces the individual name for the common variable y_{aux} of the query with many copies, each of which corresponding to one of the χ subqueries. Each such copy is connected to the parts of the ABox matching all the subqueries of $\bar{q}^{\mathcal{R}}_{c\text{-}cntd}(z)$ except the χ corresponding to this copy. However, it is connected by Aux to the main tree, so if there is a match of χ in this tree, then the overall query matches as well, that is z is witnessed by e_z. This happens when there is a violation from $\mathcal{H}$ or $\mathcal{V}$ witnessed by this particular χ.

So overall, by this construction there is a model of the knowledge base in which all *Colour*-leaves of the tree identified with colours c_m, all *Zero* and *One* nodes identified with 0 and 1 respectively, and there is no match mapping z to e_z if and only if there is a tiling of the square without violating $\mathcal{H}$ or $\mathcal{V}$. The formal proof of the fact that $p+3 \in Cert(q^{\mathcal{R}}_{c\text{-}cntd}, \mathbf{a}_\emptyset, \mathcal{K}^{\mathcal{R}}_{c\text{-}cntd})$ if and only if there is a correct tiling for the instance $(\mathcal{C}, \mathcal{H}, \mathcal{V}, n)$ of the NExp-tiling problem, can be done in exactly the same way as such a proof in Lemma 18. We omit it for brevity. □

For *DL-Lite*$_{\mathcal{R}}$ ontologies the combined complexity of count distinct problem is between coNExpTime and coN2ExpTime, the same as for count queries. Regarding *DL-Lite*$_{core}$, the situation is again very similar to count case. First, we have the following lemma.

Lemma 23. *The problem DL-Lite*$_{core}$ *Cntd-*Aggregate Certain Answers *is in* coNExpTime.

The proof of this lemma is exactly the same as the proof of Lemma 20, so we omit it for brevity.

The last problem in this paper is the lower bound for the combined complexity of the count distinct problem for the *DL-Lite*$_{core}$ case.

We do it by reduction from $\forall\exists$ *3-SAT*, the problem of verifying, given a formula in 3-CNF with variables partitioned into tuples $\mathbf{u}$ and $\mathbf{v}$, whether it is true that for every truth assignment of the variables $\mathbf{u}$, there exists a truth assignment of the variables $\mathbf{v}$ so that the formula is satisfied with the overall assignment. This problem is well known to be Π^p_2-complete [24].

The proof of the following lemma recalls the proof of Lemma 16, in the sense that the query and knowledge base are carefully designed to have possible matches of two different types, and reducing the number of matches for the aggregate variable for the first type may cause increasing this number for the second.

Lemma 24. *The problem DL-Lite*$_{core}$ *Cntd-*Aggregate Certain Answers *is* Π^p_2*-hard.*

Proof. As it is said above, the proof is by reduction from $\forall\exists$ *3-SAT*. To this end, first we show how to, given a formula ψ of the described form, construct in polynomial time a *DL-Lite*$_{core}$ knowledge base $\mathcal{K}_{c\text{-}cntd} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a Boolean count distinct ACQ $q_{c\text{-}cntd}$, and then prove that $3 \in Cert(q_{c\text{-}cntd}, \mathbf{a}_\emptyset, \mathcal{K}_{c\text{-}cntd})$ (where $\mathbf{a}_\emptyset$ is the empty tuple), if and only if ψ is valid.

Let ψ be a formula of the form

$$\forall \mathbf{u}\, \exists \mathbf{v} \bigwedge_{1 \le k \le \ell} \xi_k,$$

where each ξ_k ($1 \leq k \leq \ell$) is a clause (i.e., disjunction) of exactly three literals (i.e., variables from $\mathbf{u} \cup \mathbf{v}$ or their negations). We denote the variables of each ξ_k by w_k^1, w_k^2 and w_k^3. Let also $\mathbf{u} = u_1, \ldots, u_n$ and $\mathbf{v} = v_1, \ldots, v_m$. Without loss of generality we assume that every variable appears at least in one clause.

We start the description of the construction with the vocabulary. For every clause ξ_k it contains an atomic role $Clause_k$. Each clause ξ_k makes use of three variables, and to indicate them the vocabulary contains atomic concepts $CVar_k^1$, $CVar_k^2$ and $CVar_k^3$. To indicate all the universally qualified variables $\mathbf{u}$ it contains an atomic concept $UVar$. To connect the (elements representing the) variables to their values the vocabulary contains an atomic role Val. Finally, to specify to validating assignments of the three variables of each clause, the vocabulary contains atomic roles Asn_1, Asn_2, and Asn_3.

Let us begin with defining the Boolean count distinct ACQ as follows:

$$q_{c\text{-}cntd}(Cntd(z)) \text{ :- } \exists y_\psi, \mathbf{y}_\xi, \mathbf{y}_u^{val}, \mathbf{y}_v^{val}, \mathbf{y}_u, \mathbf{y}_v \quad \phi,$$

where

- y_ψ corresponds to the overall ψ,
- $\mathbf{y}_\xi = y_{\xi_1}, \ldots, y_{\xi_\ell}$ correspond to the clauses ξ_k,
- $\mathbf{y}_u^{val} = y_{u_1}^{val}, \ldots, y_{u_n}^{val}$ and $\mathbf{y}_v^{val} = y_{v_1}^{val}, \ldots, y_{v_m}^{val}$ correspond to values of the variables of ψ,
- $\mathbf{y}_u = y_{u_1}, \ldots, y_{u_n}$ and $\mathbf{y}_v = y_{v_1}, \ldots, y_{v_m}$ correspond to these variables,

and ϕ is the following conjunction:

$$Val(y_\psi, z) \wedge \bigwedge_{1 \leq k \leq \ell} \Big(Clause_k(y_\psi, y_{\xi_k}) \wedge$$
$$Asn_1(y_{\xi_k}, y_{w_k^1}^{val}) \wedge Asn_2(y_{\xi_k}, y_{w_k^2}^{val}) \wedge Asn_3(y_{\xi_k}, y_{w_k^3}^{val}) \wedge$$
$$CVar_k^1(y_{w_k^1}) \wedge CVar_k^2(y_{w_k^2}) \wedge CVar_k^3(y_{w_k^3}) \Big) \wedge$$
$$Val(y_{u_1}, y_{u_1}^{val}) \wedge \cdots \wedge Val(y_{u_n}, y_{u_n}^{val}) \wedge$$
$$Val(y_{v_1}, y_{v_1}^{val}) \wedge \cdots \wedge Val(y_{v_m}, y_{v_m}^{val}).$$

The graphical representation of this query is given in Figure 7. The concepts $CVar$ and ends of roles Asn depend on ψ, so they are just sketched. Note that the query mentions the role Val in two different positions, which allows to have two different types of matches in the models of the knowledge base, as previously mentioned.

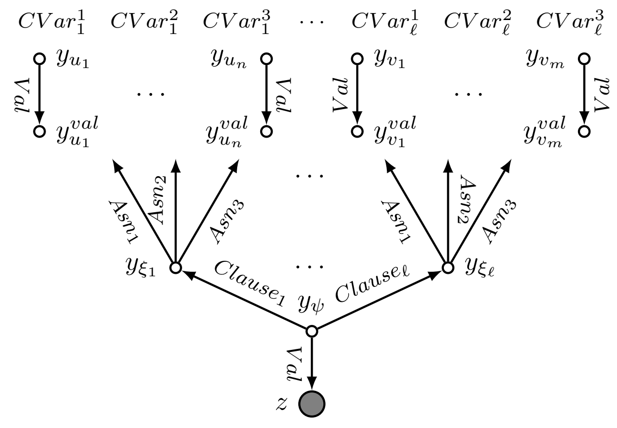

Figure 7: Query $q_{c\text{-}cntd}(Cntd(z))$. The ends of Asn_1, ..., Asn_3 starting at each y_{ξ_k}, are those of $y_{u_i}^{val}$ and $y_{v_j}^{val}$ which correspond to the variables in ξ_k; and those of y_{u_i} and y_{v_j} are labelled with $CVar_k^1$, ..., $CVar_k^3$ which correspond to the variables in ξ_k.

We continue with defining the knowledge base $\mathcal{K}_{c\text{-}cntd} = \langle \mathcal{T}, \mathcal{A} \rangle$, splitting the description in two conceptual parts. Essentially, the first part is responsible for assignment values to the variables $\mathbf{u}$, and the second one for checking whether all such assignments lead to a validation of the existential part of ψ. In fact, only the second part will depend on ψ.

The subset of Ind corresponding to the first part contains

- individual names **true** and **false**, representing the Boolean values,
- all the variables in $\mathbf{u}$ and $\mathbf{v}$ of ψ as individual names,
- individual names a_w, a_w^{val}, a_ξ, and a_ψ, playing auxiliary roles similar to the roles of names a in Lemma 16.

The ABox of the first part of the KB contains the assertions

- $CVar_k^1(a_w), \ldots, CVar_k^3(a_w)$ for all clauses ξ_k;
- $Val(a_w, a_w^{val})$;
- $Asn_1(a_\xi, a_w^{val}), \ldots, Asn_3(a_\xi, a_w^{val})$;
- $Clause_k(a_\psi, a_\xi)$ for all clauses ξ_k;

- $Val(a_\psi, \mathbf{true})$, $Val(a_\psi, \mathbf{false})$;
- $Clause_k(u_i, a_\xi)$ for all clauses ξ_k and all u_i;
- $UVar(u_i)$ for all variables u_i;
- $Val(v_j, \mathbf{true})$, $Val(v_j, \mathbf{false})$ for all v_j.

Having the first part of ABox defined, consider the TBox $\mathcal{T}$ of $\mathcal{K}_{c\text{-}cntd}$. It consists of the single inclusion

$$UVar \sqsubseteq \exists Val.$$

The aim of this inclusion is to assign some value (either **true** or **false**) to each variable u_i as we will see next.

The canonical model of this part of $\mathcal{K}_{c\text{-}cntd}$ is depicted in the upper half of Figure 8. Note that the interpretations of the individual names **true** and **false** witness the aggregate variable already by the ABox. The justifying matches for $\bar{q}_{c\text{-}cntd}$ map y_ψ to a_ψ, all y_{ξ_k} to a_ξ, all $y^{val}_{u_i}$ and $y^{val}_{v_j}$ to a^{val}_w, and, finally, y_{u_i} and y_{v_j} to a_w. The one for **true** is highlighted in the figure by thin light grey lines. Hence, $2 \in Cert(q_{c\text{-}cntd}, \mathbf{a}_\emptyset, \mathcal{K}_{c\text{-}cntd})$. Besides these two, for each variable u_i there is a match for $\bar{q}_{c\text{-}cntd}$ in the canonical model which maps the aggregate variable z to the anonymous element connected to (the interpretation of) u_i by Val^-, y_ψ to u_i and all other variables in the same way as the two matches above. Such a match for u_1 is highlighted by thick light grey lines in the figure. In search of a model with the minimal number of witnesses for z, one needs to identify these anonymous elements with the iterpretations of **true** and **false**. However, such identifications may lead to matches for the second part of the ABox which we describe next.

Given a clause ξ_k with variables w^1_k, w^2_k, and w^3_k, let $\sigma^1_k, \dots, \sigma^7_k$ be all the assignments of these variables which satisfy ξ_k. The subset of Ind corresponding to the second part of the ABox $\mathcal{A}$ contains

- all satisfying assignments σ^p_k, $1 \leq p \leq 7$, of all clauses ξ_k as individual names,
- auxiliary individual names b_ψ and b_z.

The second part of ABox contains the assertions

- $CVar^1_k(w^1_k), \dots, CVar^3_k(w^3_k)$ for all clauses ξ_k with variables $w^1_k, \dots, w^3_k$;

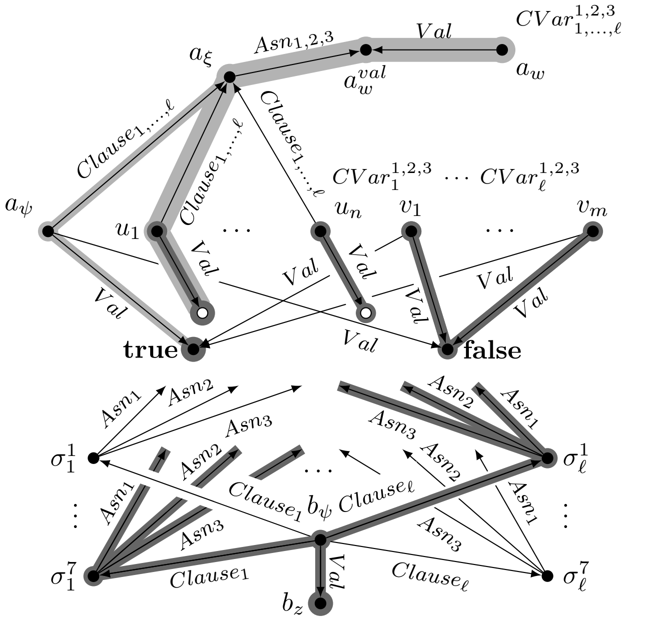

Figure 8: Canonical model of $\mathcal{K}_{c\text{-}cntd}$. The ends of Asn_1, …, Asn_3 starting at each σ^p_k, are the Boolean values of the variables in ξ_k under σ^p_k as assignment; and those of u_i and v_j are labelled with $CVar^1_k$, …, $CVar^3_k$ which correspond to the variables in ξ_k.

- $Asn_1(\sigma^p_k, \sigma^p_k(w^1_k)), \dots, Asn_3(\sigma^p_k, \sigma^p_k(w^3_k))$ for all satisfying assignments σ^p_k of all clauses ξ_k (here $\sigma^p_k(w^i_k)$ as the first argument is an individual name, but $\sigma^p_k(w^i_k)$ is one of the individual names **true** or **false** which is the value of σ^p_k as assignment on w^i_k);
- $Clause_k(b_\psi, \sigma^p_k)$ for all satisfying assignments σ^p_k of all clauses ξ_k;
- $Val(b_\psi, b_z)$.

The second part of $\mathcal{A}$ is depicted in the lower half of Figure 8. Note that it substantially depends on ψ, so some pieces are just sketched, but described in the caption. This part serves the following aim. As said above, in search of a model with the minimal number of witnesses of the aggregation variable z one needs to identify the anonymous elements corresponding to u_i with either **true** or **false**; however, such identification may lead to a match for the query with b_z witnessing the aggregation variable z, by this increasing the aggregation value. Such a possible match is highlighted in the figure by thin dark grey lines. Next we will see, that a possibility to find an identification which does not lead to a

	Data complexity		Combined complexity	
	Count	*Cntd*	*Count*	*Cntd*
DL-Lite$_{core}$	coNP-complete	coNP-complete	Π_2^p-hard and in coNExp	Π_2^p-hard and in coNExp
DL-Lite$_{\mathcal{R}}$	coNP-complete	coNP-complete	coNExp-hard and in coN2Exp	coNExp-hard and in coN2Exp

Table 1: A summary of the complexity results.

match is equivalent to invalidation of ψ.

Formally, we need to show that $3 \in Cert(q_{c\text{-}cntd}, \mathbf{a}_\emptyset, \mathcal{K}_{c\text{-}cntd})$ (i.e., that every model of $\mathcal{K}_{c\text{-}cntd}$, which connects each u_i to either **true** or **false** by Val, has a match with b_z witnessing z) holds for the empty tuple $\mathbf{a}_\emptyset$ if and only if ψ is valid.

($\Leftarrow$) Let for every truth assignment of the variables $\mathbf{u}$ there exist a truth assignment of the variables $\mathbf{v}$ so that each ξ_k is satisfied with the overall assignment, yet assume for the sake of contradiction that there is a model $\mathcal{I}$ of $\mathcal{K}_{c\text{-}cntd}$ such that **true** and **false** are the only witnesses for the varibale z in the matches from $\bar{q}$ to $\mathcal{I}$.

From the construction of $\mathcal{K}_{c\text{-}cntd}$, for each pair (d_{u_i}, d) in $Val^{\mathcal{I}}$, where d_{u_i} is the interpretation of u_i, it must be the case that d is either **true** or **false** (otherwise it violates the observation previously mentioned, since this would give an extra witness for the variable z). Without loss of generality let each u_i have only one such d, since dropping extra Val-connections would not make $\mathcal{I}$ to be not a model. Construct the following valuation $\sigma_{\mathbf{u}}$ for the variables $\mathbf{u}$: for each u_i, let

$$\sigma_{\mathbf{u}}(u_i) = \begin{cases} \textbf{true}, & \text{if } (d_{u_i}, d_{\textbf{true}}) \in Val^{\mathcal{I}}, \text{ and} \\ \textbf{false}, & \text{otherwise,} \end{cases}$$

where $d_{\textbf{true}}$ is the interpretation of **true**. From the original assumption, there must be an assignment $\sigma_{\mathbf{v}}$ of the variables in $\mathbf{v}$ such that $\sigma_{\mathbf{u}} \cup \sigma_{\mathbf{v}}$ satisfies ϕ. We show that $\bar{q}_{c\text{-}cntd}(b_z)$ must hold in $\mathcal{I}$. To that extent, construct the following mapping h from the variables of q to elements in $\mathcal{I}$: h maps the variables y_{u_i} and y_{v_j} to the interpretations of u_i and v_j respectively, as well as each of $y_{u_i}^{val}$ and $y_{v_j}^{val}$ to $d_{\textbf{true}}$ if $\sigma_{\mathbf{u}} \cup \sigma_{\mathbf{v}}$ assigns the value **true** to the corresponding variable u_i or v_j, or to the interpretation of **false** otherwise. Moreover, for each clause ξ_k, it maps each variable y_{ξ_k} to the interpretation of the individual name σ_k^p such that the p-th satisfying assignment for ξ_k is the one witnessed by $\sigma_{\mathbf{u}} \cup \sigma_{\mathbf{v}}$. Finally, it maps y_ψ to the interpretation of b_ψ and z to the interpretation of b_z. It is a matter of technicality to check that h is a match for $\bar{q}_{c\text{-}cntd}$ in $\mathcal{I}$, that is the interpretation of b_z is the third image of z in $\mathcal{I}$. This violates our original assumption.

($\Rightarrow$) Let $3 \in Cert(q_{c\text{-}cntd}, \mathbf{a}_\emptyset, \mathcal{K}_{c\text{-}cntd})$, but assume for the sake of contradiction that there is a truth assignment $\sigma_{\mathbf{u}}$ for the variables $\mathbf{u}$ such that ϕ is not satisfiable under any assignment for the variables $\mathbf{v}$.

Construct the following model $\mathcal{I}$ for $\mathcal{K}$.

- The interpretation of any individual name a is an element d_a.
- The interpretation of all atomic roles and concepts except for Val corresponds precisely to the ABox $\mathcal{A}$. That is, for each atomic role R different from Val, we have that $(d_a, d_b) \in R^{\mathcal{I}}$ if and only if $R(a, b)$ is an assertion in $\mathcal{A}$, and likewise for all the atomic concepts.
- For each u_i, the pair $(d_{u_i}, d_{\sigma_{\mathbf{u}}(u_i)})$ belongs to $Val^{\mathcal{I}}$. Here we assume that u_i is an individual name in the first argument, but a variable in the second; also, $\sigma_{\mathbf{u}}(u_i)$ valuates to one of the individual names **true** or **false**.

It is a technicality to check that $\mathcal{I}$ is indeed a model of $\mathcal{K}_{c\text{-}cntd}$. From the assumption that $3 \in Cert(q_{c\text{-}cntd}, \mathbf{a}_\emptyset, \mathcal{K}_{c\text{-}cntd})$ and the construction of $\mathcal{I}$, it must be the case that $\bar{q}_{c\text{-}cntd}(b_z)$ holds in $\mathcal{I}$. It follows that there must be a match h for $q_{c\text{-}cntd}$ in $\mathcal{I}$ such that the mapping h sends the variable z to the interpretation of b_z. We can, however, determine more properties of h from the construction of $\mathcal{I}$ and $\mathcal{K}_{c\text{-}cntd}$. In fact, it follows that each variable y_{u_i}, is indeed mapped by h to (the interpretation of) u_i in $\mathcal{I}$, and that $y_{u_i}^{val}$ is mapped to the corresponding valuation of u_i according to $\sigma_{\mathbf{u}}$. It follows from the construction of ϕ that the following assignment $\sigma_{\mathbf{v}}$ of the variables $\mathbf{v}$ is such that $\sigma_{\mathbf{u}} \cup \sigma_{\mathbf{v}}$ satisfies ϕ: $\sigma_{\mathbf{v}}$ assigns the value **true** to v_j if the variable y_{v_j} is mapped to (the interpretation of) **true**, according to h, and the value **false** otherwise. This contradicts the original assumption and completes the proof of the lemma. $\square$

Summing up, we have our last theorem.

Theorem 25.
*(1) The problem DL-Lite$_{core}$ Cntd-*AGGREGATE CERTAIN ANSWERS *is in* coNExpTime *and* Π_2^p*-hard.*
*(2) The problem DL-Lite$_{\mathcal{R}}$ Cntd-*AGGREGATE CERTAIN ANSWERS *is in* coN2ExpTime *and* coNExpTime*-hard.*

7. Conclusion

In this paper we have defined an intuitive semantics for counting aggregate queries over ontologies and explored the computational complexity of the corresponding problems. The results, summarized in Table 1, show that the problems are decidable, but intractable. Hence, heuristics and approximations for answering ACQs are on high demand from the practical point of view, with applications, for instance, in the definition of general aggregation in SPARQL under entailment regimes. We consider epistemic and active domain semantics as such approximations, since they have lower data complexities but do not always provide the desired answer. Our work provides the theoretical foundations for further discussion.

There are several directions for future work. First, it is important to close the gaps in the combine complexities. Second, in Section 4 we mentioned the following natural problem closely related to the problems studied in this paper: for a query, knowledge base, and a tuple of individual names, how to compute the minimum value of the counting aggregation function over all the models of the knowledge base? The corresponding decision problem is whether a given number n is such a minimum? In fact, we can easily derive a DP-upper bound for the data complexity of this problem from the results of this paper: first one needs to check whether n belongs to the aggregate certain answers, and then check that $n+1$ is not. However, it remains to see whether the problem is indeed DP-hard. Similar situation is for combine complexity problems. Finally, in this paper the count distinct function has exactly one argument, as it is done in SQL and SPARQL. It is interesting to study the generalisation of this function to arbitrary number of aggregation variables.

References

[1] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, New York, NY, USA (2003)

[2] Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The MASTRO system for ontology-based data access. Semantic Web **2**(1) (2011) 43–53

[3] Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: IJCAI. (2011) 2656–2661

[4] Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. Web Semant. **6**(4) (November 2008) 309–322

[5] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning **39**(3) (2007) 385–429

[6] Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artif. Intell. Res. (JAIR) **36** (2009) 1–69

[7] Bienvenu, M., Ortiz, M., Simkus, M.: Answering expressive path queries over lightweight DL knowledge bases. In Kazakov, Y., Lembo, D., Wolter, F., eds.: Description Logics. Volume 846 of CEUR Workshop Proceedings., CEUR-WS.org (2012)

[8] Bienvenu, M., Ortiz, M., Simkus, M.: Conjunctive regular path queries in lightweight description logics. In Rossi, F., ed.: IJCAI, IJCAI/AAAI (2013)

[9] Kostylev, E.V., Reutter, J.L., Vrgoč, D.: XPath for DL ontologies. In: AAAI. (2015)

[10] Rosati, R.: The limits of querying ontologies. In Schwentick, T., Suciu, D., eds.: ICDT. Volume 4353

of Lecture Notes in Computer Science., Springer (2007) 164–178

[11] Gutiérrez-Basulto, V., Ibáñez-García, Y.A., Kontchakov, R., Kostylev, E.V.: Conjunctive queries with negation over DL-Lite: A closer look. In: Proceedings of the 7th international conference on Web Reasoning and Rule Systems. RR'13 (2013)

[12] Cohen, S., Nutt, W., Sagiv, Y.: Deciding equivalences among conjunctive aggregate queries. Journal of the ACM **54**(2) (2007)

[13] : SPARQL 1.1 entailment regimes (2013) W3C Recommendation 21 March 2013, `http://www.w3.org/TR/ sparql11-entailment/`.

[14] Calvanese, D., Kharlamov, E., Nutt, W., Thorne, C.: Aggregate queries over ontologies. In Elmasri, R., Doerr, M., Brochhausen, M., Han, H., eds.: ONISW, ACM (2008) 97–104

[15] Vardi, M.Y.: The complexity of relational query languages (extended abstract). In: STOC. (1982) 137–146

[16] Kostylev, E.V., Reutter, J.L.: Answering counting aggregate queries over ontologies of the DL-Lite family. In: Proc. of the 27th AAAI Conf. on Artificial Intelligence (AAAI). (2013)

[17] Arenas, M., Bertossi, L., Chomicki, J., He, X., Raghavan, V., Spinrad, J.: Scalar aggregation in inconsistent databases. Theor. Comput. Sci. **296**(3) (March 2003) 405–434

[18] Libkin, L.: Data exchange and incomplete information. In Vansummeren, S., ed.: PODS, ACM (2006) 60–69

[19] Afrati, F., Kolaitis, P.G.: Answering aggregate queries in data exchange. In: Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. PODS '08, New York, NY, USA, ACM (2008) 129–138

[20] W3C SPARQL Working Group: SPARQL 1.1 Query language. W3C Recommendation (21 March 2013) Available at `http://www.w3.org/TR/sparql11-query/`.

[21] Baader, F., Sattler, U.: Description logics with aggregates and concrete domains. Information Systems **28**(8) (2003) 979 – 1004

[22] Kostov, B., Kremen, P.: Count aggregation in semantic queries. In Liebig, T., Fokoue, A., eds.: SSWS@ISWC. Volume 1046 of CEUR Workshop Proceedings., CEUR-WS.org (2013) 1–16

[23] Ramakrishnan, R., Gehrke, J., Gehrke, J.: Database management systems. Volume 3. McGraw-Hill New York (2003)

[24] Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)

[25] Arenas, M., Barceló, P., Reutter, J.L.: Query languages for data exchange: Beyond unions of conjunctive queries. Theory Comput. Syst. **49**(2) (2011) 489–564

[26] Johnson, D.S.: A catalog of complexity classes. In: Handbook of theoretical computer science (vol. A), MIT Press (1991) 67–161

Response to the reviewers

We want to thank both reviewers first for their careful reading and all suggestions for the paper. This time we put a lot of energy into proofreading. We have corrected the typos and addressed the grammatical suggestions of the reviewers, and we believe that the language is now up to the standard.

Specific responses to Reviewer 1.

We will first comment on the main points raised by this reviewer, and then proceed with the minor issues.

- Regarding the definitions of count and count distinct functions. The observation that these two functions are particular cases of a general counting function is absolutely correct. However, our choice of these specific functions is motivated by practice: both SQL and SPARQL allow only for a general Count function (as in the query COUNT(*) in SQL) and a Count distinct function over values of just one argument (as in COUNT (DISTINCT name)). Note that though syntactically SQL allows for an arbitrary number of attributes of the COUNT function, but it has the same meaning as COUNT(*) (as long as there are no nulls in the database). Hence, we studied these particular functions in the paper in order to make a strong case of the practical background of our work.

  From a theoretical point of view, of course, nothing prevents us from studying the general function. However, after considerable thought it is still not clear for us how to extend any of the upper bounds to account for all the intermediate cases. This would require the notion of neighbourhoods for arbitrary sets of elements, and it is not clear at all how these should be defined and used. We have mentioned the study of this generalisation in the conclusion as part of the future work.

- Regarding the definition of Count and Ctnd as the maximum number that is an aggregate certain answer of a query, we have added a discussion in the paper explaining why we think that our definition is more natural. The main point in question is being able to answer the query "Is the answer to this ACQ at least n?", since it is in line with OWA. Another argument is similarity with range semantics in inconsistent databases. In the conclusion we added a paragraph explaining why we can not just switch back and forth between two definitions (the complexity for the problem under the definitions suggested by the reviewer is probably higher than in our case).

- Regarding the partial complexity results, we wanted to make sure in the paper that the problems do not seem to be solvable by any other technique developed in the DL community. We have spent a considerable amount of time thinking about these results, unfortunately still without being able to close the complexity gaps. We still believe that our results are worthy of publication, as the proofs introduce new techniques for the toolbox of the community.

- We have gone through the entire paper making our notation and terminology standard for DL community. We also formally described how our proofs can be adapted for non-UNA cases.
- We extended the related work section with the discussion on the paper suggested by the reviewer. We also made the section self-contained. The only references to literature left in the introduction are important for the exposition.

We have addressed all of the technical comments/errors pointed by the reviewer. Below we comment on those that require discussion not in the paper.

- We added the correct reference for SPARQL 1.1 and a remark that non-distinguished variables in SPARQL queries are those that are not mentioned in the SELECT or CONSTRUCT clause.
- The problem with some of the figures is that we consider appropriate to highlight three matches, so we used three tones of grey. The tones are darker now, and we believe it should be distinguishable in any decent quality printing. However, if the reviewer can suggest a better way for highlighting, we will be happy to follow the suggestion.
- In the beginning of the proof of Lemma 22 we explained in more detail what was the intention of the query in the proof of Lemma 18. In particular, the TBox guarantees that the leaf of the branch corresponding to each position in the square is in the domain of the role Tile, but the query is needed to have only the colours in the range of Tile.