Aggregate Queries over Ontologies

Diego Calvanese, Evgeny Kharlamov, Werner Nutt, Camilo Thorne
Free University of Bozen-Bolzano
Piazza Domenicani, 3
39100 Bozen-Bolzano, Italy
{calvanese,kharlamov,nutt,cthorne}@inf.unibz.it

ABSTRACT

Answering queries over ontologies is an important issue for the Semantic Web. Aggregate queries were widely studied for relational databases but almost no results are known for aggregate queries over ontologies. In this work we investigate the latter problem. We propose syntax and semantics for epistemic aggregate queries over ontologies and study query answering for `MAX, MIN, COUNT, CNTD, SUM, AVG` queries for an ontology language *DL-Lite*$_{\mathcal{A}}$.

Categories and Subject Descriptors

H.2.3 [**Languages**]: Query languages; H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Syntax, Semantics, Algorithms

Keywords

Aggregate Queries, Incomplete Information, Ontology Languages, Description Logic

1. INTRODUCTION

The semantic web and current web technologies advocate and make widespread use of information structured by and with ontologies. By an ontology we understand both (i) a conceptual model of a domain of interest, that is, concepts and relations that we may abstract from the domain, plus the constraints that hold over them and (ii) a data set that instantiates this model. In contrast with databases, data set may be incomplete with respect to the conceptual model. Usually ontologies are viewed as logical theories stated in an ontology language such as OWL and that therefore draws it formal foundations from the field of the description logics.

For instance, a conceptual model of an ontology depicted by the ER-diagram in Figure 1. This family ontology characterizes a state of affairs common in Italy, where families own cars and receive children tax reductions (CTR). CTR is calculated according to the salary of a person and is given for each child the person has. On the diagram the relation CTR associates Parents with their TaxReductions, HasChild relation associates Parents with their Children and HasCar associates Parents with their Cars.

Given an ontology, one is typically interested in posing queries to it and retrieving data from its data set. The issue of query answering over ontologies was studied by the community [8, 9, 12] for `SELECT-FROM-WHERE` fragment of SQL, that is, for conjunctive queries. Similar results have been obtained in the area of incomplete databases [6, 5], that is, databases under the open world assumption. Conjunctive queries, however, do not allow us to express many natural and interesting information requests, such as those expressed by *aggregate queries*, like `COUNT`, `MIN`, `AVG` SQL queries. Consider examples of these questions in Table 1 below.

There are almost no results related to querying ontologies with aggregate queries [10, 4, 3]. The main aim and contribution of this paper is to fill this gap.

The structure of the paper is the following. In Sections 2 and 3 we remind the relevant notions from relational databases and Description Logic *DL-Lite*$_{\mathcal{A}}$. Moreover, in Section 2 we propose our syntax and semantics for rule-based conditional aggregate queries. In Section 4 we discuss the problems of adopting certain answer semantics to aggregate queries over ontologies. In Section 5 we introduce epistemic aggregate

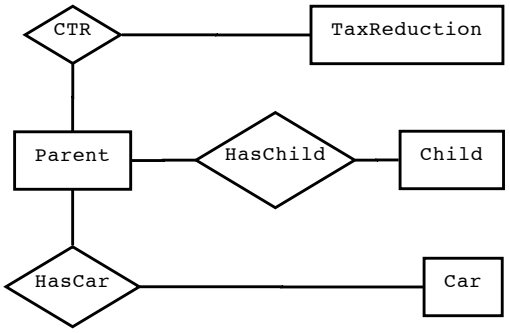

Figure 1: The family ontology.

queries and in Section 6 we propose algorithms for evaluating them over *DL-Lite*$_\mathcal{A}$ ontologies.

2. AGGREGATE QUERIES OVER DATABASES

In this section we remind the reader a theory of relational databases, conjunctive and aggregate queries in SQL and rule-based notation (see [1] for details). We also remind nested conditional SQL queries and introduce an extension of rule-based notation to capture these queries.

2.1 Databases and Conjunctive Queries

We assume as given a countably infinite *domain* $\Delta := \Delta_O \cup \Delta_V$ partitioned into a domain Δ_O of object constants and a domain Δ_V of values (containing numbers, e.g., $\mathbb{N}$). We call *tuple* any finite sequence $\bar{c}$ of domain elements. We will denote the empty tuple as ().

A *term* (like t) is either a variable (like v, w, x, y, z) or a constant (like c). A *relation name* R is a predicate symbol of arity n (a nonnegative integer). Terms and relations are assumed to be conveniently typed. A *database schema* $\mathbf{R}$ is a finite set of relation names. An *atom* is an expression of the form $R(\bar{t})$, where R is a relation name of arity n and $\bar{t}$ is a sequence of n terms. An atom is *ground* when all its terms are constants.

A *condition* ϕ over $\mathbf{R}$ is a conjunction of atoms with relation names from $\mathbf{R}$ which we write in the form of a list. We denote by $Var(\phi)$ the set of variables occurring in condition ϕ.

A *conjunctive query* (CQ) over $\mathbf{R}$ is an expression of the form:

$$q(\bar{x}) \leftarrow \phi,$$

where q is a relation of some arity n that does not occur in $\mathbf{R}$, called *head relation*, and ϕ is over $\mathbf{R}$. The variables in $\bar{x}$ are called *distinguished variables* and ϕ is called condition. Distinguished variables should occur in ϕ. A conjunctive query is *boolean* when the sequence $\bar{x}$ is empty. As usual we identify a conjunctive query with its head relation q and denote $Var(()q)$ the set of all the variable in the condition of q.

A *database instance* (DB) D of $\mathbf{R}$ is a pair $(\Delta, \cdot^D)$ where Δ is the domain and $\cdot^D$ is an *interpretation function* over $\mathbf{R}$, that is, a function mapping each relation symbol R of arity n in $\mathbf{R}$ to a subset R^D of Δ^n, i.e., to a *relation instance*. Observe that DBs are the first order logic interpretation of $\mathbf{R}$. We notice that in this paper we allow DBs to be infinite. This is done for consistency of the presented theory of aggregate queries over ontologies. It does not conflict with the fact that in practice databases are finite because our theory deals with finite representations of the infinite DBs, i.e. with knowledge bases (we introduce knowledge bases in Section 3).

(a)	What is the maximum tax reduction recorded in the ontology?
(b)	How many parents own cars?
(c)	What is the total tax reduction of each parent?

Table 1: Example of questions that involves aggregations.

Example 2.1. Consider the DB schema with relations from Figure 1, $\mathbf{R}_f := \{Parent, CTR, HasCar, HasChild\}$, where f in $\mathbf{R}_f$ stands for "family". A database D_f of $\mathbf{R}_f$ is

Parent

Name
Luisa
Anna

CTR

PName	Amount
Luisa	100
Anna	150

HasCar

PName	CType
Anna	vw golf
Anna	fiat uno

HasChild

PName	CName
Anna	Mario
Luisa	Beppe
Anna	Paolo

For convenience, we use attribute names to denote relation positions. ∎

An *assignment* γ over a condition ϕ is a function that maps $Var(\phi)$ to Δ and each constant in ϕ to itself. Assignments are extended to complex syntactic objects like atoms and conditions in the usual way. Whenever $\gamma(\bar{x}) = \bar{c}$ we say that $\bar{x}$ is *bound* to $\bar{c}$. An assignment γ is *satisfies* ϕ over a DB D if, for each atom $R(\bar{t})$ in ϕ, $\gamma(\bar{t}) \in R^D$. We denote by $Sat_D(\phi)$ the set of satisfying assignments of ϕ over D.

The *set of answers* q^D of a CQ q over a DB D is defined as

$$q^D := \{\bar{c} \mid \bar{c} = \gamma(\bar{x}),\ \gamma \in Sat_D(\phi)\}.$$

2.2 Aggregate Queries

We enrich our syntax by considering now the following standard SQL *aggregation functions*:

$$max,\quad min,\quad count,\quad cntd,\quad sum,\quad avg,$$

In what follows, α will denote an arbitrary aggregation function. We call an *aggregation term* any expression either of the form *count* or $\alpha(y)$, where y is called an *aggregation variable*. We use $\alpha(\bar{y})$ to denote both cases.

An *aggregate query* (AQ) over $\mathbf{R}$ is a query of the form:

$$q(\bar{x}, \alpha(\bar{y})) \leftarrow \phi,$$

where $\bar{x}$ is a (possibly empty) sequence of *grouping variables*, $\alpha(\bar{y})$ is an aggregation term and ϕ is a condition containing $\bar{x}$ and $\bar{y}$. Moreover, $\bar{y}$ does not occur in $\bar{x}$.

Example 2.2. Question (a) in the introduction gives rise to the following *max*-query q_1 over $\mathbf{R}_f$ from Example 2.1:

$$q_1(max(y)) \leftarrow Parent(x), CTR(x, y).$$

The same query in SQL notation is:

```
SELECT MAX(CTR.Amount)
FROM   Parent, CTR
WHERE  Parent.Name = CTR.PName
```

There is no `GROUP BY` clause, which reflects the fact that in q_1 there are no grouping variables. ∎

Aggregation functions in SQL are defined over bags $\{\!|.|\!\}$, called *groups*, of symbolic and numerical values and return a number. Let D be a database and $\bar{c}$ a tuple. Consider an aggregate query $q(\bar{x}, \alpha(\bar{y})) \leftarrow \phi$. The *group* $F_{\bar{c}}$ of tuple $\bar{c}$ is defined as the bag

$$F_{\bar{c}} := \{\!|\gamma(\bar{y}) \mid \bar{c} = \gamma(\bar{x}) \text{ and } \gamma \in Sat_D(\phi)|\!\}.$$

The *set of answers* q^D of an ACQ q over a D is defined as

$$q^D := \{(\bar{c}, \alpha(F_{\bar{c}})) \mid \bar{c} = \gamma(\bar{x}),\ \gamma \in Sat_D(\phi)\}.$$

EXAMPLE 2.3. Consider *max*-query q_1 of Example 2.2 and the database D_f of Example 2.1. There are only two satisfying assignments, namely, $\gamma_1 := [x/Luisa, y/100]$ and $\gamma_2 := [x/Anna, y/150]$. Since q_1 has no grouping variables, the only group we have is for the empty tuple () and $F_{()} = \{\!|100, 150|\!\}$ Therefore $q_1^{D_f} = \{(max(F_{()}))\} = \{(max(\{\!|100, 150|\!\}))\} = \{(150)\}$. ∎

2.3 Conditional Aggregate Queries

The standard syntax of aggregate queries can only express some SQL aggregate queries. For instance, the question (b) from Table 1 corresponds to the SQL query (over $\mathbf{R}_f$):

```
SELECT COUNT(*)
FROM   Parent
WHERE  EXISTS (
         SELECT *
         FROM   HasCar
         WHERE  HasCar.PName = Parent.Name)
```

which contains an SQL (sub)query that is *nested* within an `EXISTS` condition. Nested SQL subqueries of this kind do not return bags, but a truth value that will be `true` if and only if there is a tuple verifying the conditions of the subquery.

To cover this feature we must extend accordingly the syntax and semantics of aggregate queries. This nesting will be expressed by bracketing atoms in the query's condition.

A *conditional aggregate query* q (CAQ) is a query of the form:

$$q(\bar{x}, \alpha(\bar{y})) \leftarrow \phi, [\psi],$$

where ϕ and ψ are conditions. We call ψ a *nested condition*. In addition, $\bar{x}, \bar{y} \subseteq Var(\phi)$ and $Var(\phi) \cap Var(\psi) \neq \emptyset$. As before, we identify queries with their head relation.

We distinguish assignments of $Sat_D(\phi, \psi)$ that give way to bags of tuples and assignments that do not, i.e., assignments that satisfy nested conditions. We define the later ones as

$$Sat_D^{\psi}(\phi) := \{\gamma_{|\,Var(\phi)} \mid \gamma \in Sat_D(\phi, \psi)\}.$$

The *group* of a tuples $\bar{c}$, denoted $G_{\bar{c}}$, is defined similar to groups $F_{\bar{c}}$ but using assignments from $Sat_D^{\psi}(\phi)$:

$$G_{\bar{c}} := \{\!|\gamma(\bar{y}) \mid \bar{c} = \gamma(\bar{x}) \text{ and } \gamma \in Sat_D^{\psi}(\phi)|\!\}.$$

We define *the set of answers* q^D of CAQ q over DB D as

$$q^D := \{(\bar{c}, \alpha(G_{\bar{c}})) \mid \bar{c} = \gamma(\bar{x}), \text{ for some } \gamma \in Sat_D^{\psi}(\phi)\}.$$

EXAMPLE 2.4. With the new syntax we can express the question (b) from Table 1 as a CAQ q_2 over $\mathbf{R}_f$ as follows:

$$q_2(count) \leftarrow Parent(x), [HasCar(x, y)].$$

For the DB D_f from Example 2.1 we obtain $q_2^{D_f} = \{(2)\}$. ∎

3. THE DL-LITE$_A$ ONTOLOGY LANGUAGE

We present now the ontology language on which we base our considerations in the rest of the paper, namely the Description Logic *DL-Lite*$_\mathcal{A}$. Such a language is one of the most expressive variants of Description Logics (DLs) of the *DL-Lite* family [7], which is a family of DLs that have been developed specifically to handle large amounts of data, possibly stored in relational databases [11]. *DL-Lite*$_\mathcal{A}$ is also one of the tractable fragments of the upcoming standard OWL 2[1]. As usual in DLs, in *DL-Lite*$_\mathcal{A}$ the universe of discourse is represented in terms of concepts, denoting sets of objects, and roles, denoting binary relations between objects. Additionally, *DL-Lite*$_\mathcal{A}$ provides mechanisms to deal both with abstract objects (which are instances of concepts) and with data values (such as strings, integers, ...): value-domains denote sets of values, while (concept) attributes[2] denote binary relations between objects and values. The distinction between objects and values is especially important in the ontology-based data access setting, where one needs to deal with the fact that (abstract) objects are maintained at the level of the ontology, while data values are stored in the underlying database.

3.1 The DL-Lite$_A$ language

In providing the specification of *DL-Lite*$_\mathcal{A}$, we use the following notation:

- A denotes an *atomic concept*, B a *basic concept*, C a *general concept*, and $\top_C$ the *universal concept*.
- E denotes a basic value-domain, i.e., the range of an attribute, F a *value-domain expression*, and $\top_D$ the *universal value-domain*.
- P denotes an *atomic role*, Q a *basic role*, and R a *general role*. An atomic role is simply a role denoted by a name.
- U denotes an *atomic attribute* (or simply attribute), and V a *general attribute*.

An atomic concept or an atomic value-domain is simply a unary relation symbol, while an atomic role or an atomic attribute is simply binary relation symbol. The syntax of *basic* and *general* concepts, value-domains, roles, and attributes is given below. Given an attribute U, we call the *domain* of U, denoted by $\delta(U)$, the set of objects that U relates to values, and we call *range* of U, denoted by $\rho(U)$, the set of values that U relates to objects. Note that the domain $\delta(U)$ of an attribute U is a concept, whereas the range $\rho(U)$ of U is a value-domain.

[1] `http://www.w3.org/2007/OWL/`

[2] For simplicity, we do not consider here role attributes.

Formally, the syntax of *DL-Lite*$_\mathcal{A}$ expressions is defined as follows:

$$\begin{array}{lcl} B & ::= & A \mid \exists Q \mid \delta(U) \\ C & ::= & \top_C \mid B \mid \neg B \mid \exists Q.C \\ E & ::= & \rho(U) \\ F & ::= & \top_D \mid T_1 \mid \cdots \mid T_n \end{array} \qquad \begin{array}{lcl} Q & ::= & P \mid P^- \\ R & ::= & Q \mid \neg Q \\ V & ::= & U \mid \neg U \end{array}$$

The semantics of *DL-Lite*$_\mathcal{A}$ is specified, as usual in DLs, in terms of first-order logic interpretations, i.e., database instances, over the domain $\Delta = \Delta_O \cup \Delta_V$ (cf. Section 2). All such database instances agree on the semantics assigned to each value-domain T_i. In particular, we assume that Δ_V coincides with the union of the interpretations of all T_i. Given a database instance $D = (\Delta, \cdot^D)$ interpreting unary and binary atomic relation symbols, the interpretation is extended to complex expressions in the usual way. For lack of space we don't provide here the details of the semantics of the constructs, and refer instead to [11].

3.2 Knowledge bases

A *DL-Lite*$_\mathcal{A}$ *knowledge base* (KB), or ontology, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is constituted by two components: a TBox $\mathcal{T}$, used to represent intensional knowledge, and an ABox $\mathcal{A}$, representing extensional information. The *TBox* is constituted by a set of assertions of the form:

B	$\sqsubseteq$ C	*(concept inclusion assertion)*
Q	$\sqsubseteq$ R	*(role inclusion assertion)*
E	$\sqsubseteq$ F	*(value-domain inclusion assertion)*
U	$\sqsubseteq$ V	*(attribute inclusion assertion)*
(funct Q)		*(role functionality assertion)*
(funct U)		*(attribute functionality assertion)*

A concept inclusion assertion expresses that a (basic) concept B is subsumed by a (general) concept C. Analogously for the other types of inclusion assertions. A role functionality assertion expresses the (global) functionality of an atomic role. Analogously for an attribute functionality assertion.

Formally, a database instance D satisfies an inclusion assertion $X \sqsubseteq Y$ if the corresponding set inclusion $X^D \subseteq Y^D$ holds, and it satisfies a functionality assertion (funct Q) (resp., (funct U)), if the relation Q^D (resp., U^D) is a function.

A *DL-Lite*$_\mathcal{A}$ ABox is a finite set of *membership assertions* of the form $A(a)$, $P(a, b)$, and $U(a, c)$, where a and b are object constants in Δ_O, and c is a value in Δ_V. To define its semantics, we specify when a database instance $D = (\Delta, \cdot^D)$ satisfies a membership assertion α in $\mathcal{A}$, written $D \models \alpha$. Namely, D satisfies $A(a)$ if $a \in A^D$, $P(a, b)$ if $(a, b) \in P^D$, and $U(a, b)$ if $(a, b) \in U^D$.

D is a *model* of a *DL-Lite*$_\mathcal{A}$ KB $\mathcal{K}$ or, equivalently, D *satisfies* $\mathcal{K}$, written $D \models \mathcal{K}$ iff D satisfies all assertions in $\mathcal{K}$. A KB is *satisfiable* if it has at least one model. A KB $\mathcal{K}$ *logically implies* an assertion α if all models of $\mathcal{K}$ are also models of α. (Similar definitions hold for a TBox $\mathcal{T}$ or ABox $\mathcal{A}$.)

3.3 Query Answering in DL-Lite$_\mathrm{A}$

A query over a knowledge base $\mathcal{K}$ is a query whose predicates are the *atomic* concepts, value domains, roles, and attributes of $\mathcal{K}$. The reasoning service we are interested in is *query answering*: given a knowledge base $\mathcal{K}$ and a query $q(\bar{x})$ over $\mathcal{K}$, return the *certain answers* $Cert(q, \mathcal{K})$ to $q(\bar{x})$ over $\mathcal{K}$, i.e., all tuples $\bar{t}$ of elements of $\Delta_O \cup \Delta_V$ such that $\bar{t} \in q^D$ for every model D of $\mathcal{K}$.

Note that query answering over an ontology is a form of reasoning over incomplete DBs, and as such, in general, a more complex task than plain query answering over a database. Indeed, from the results in [7] it follows that, in general, already answering CQs over *DL-Lite*$_\mathcal{A}$ KBs as introduced above, is PTIME-hard in *data complexity* (i.e., the complexity measured w.r.t. the size of the ABox only). As a consequence, to solve query answering over such KBs, we need at least the power of general recursive Datalog.

However, as shown in [7], the data complexity of answering unions of CQs in *DL-Lite*$_\mathcal{A}$ drops to LOGSPACE, provided a suitable restriction is posed on the interaction between role (resp., attribute) inclusion assertions and functionality assertions. Intuitively, the restriction requires that no functional role or attribute can be specialized, i.e., appear in the left-hand side of an inclusion assertion, when in the right-hand side there is a non-negated role or attribute (we refer to [7] for the details). Moreover, if such a condition is satisfied, computing the certain answers of a query with respect to a satisfiable *DL-Lite*$_\mathcal{A}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ can be reduced, through a process called *perfect reformulation*, to the evaluation over $\mathcal{A}$ (now viewed as a complete database) of a suitable union of CQs [7]. Such a query answering technique has also been implemented in the QuOnto system [2], where the ABox is stored directly in a relational database, containing one relation for each atomic symbol. In the following, we will also assume that the ABox of the ontology over which queries are answered is stored in the form of database relations, one for each atomic concept, attribute, and role symbol. As for domains, we do not need to store them explicitly in the database, since they are actually only used to represent the range of attributes.

4. LIMITATIONS OF CERTAIN ANSWER SEMANTICS

As we discussed in the previous section, the problem of answering queries over ontologies is a special case of the one of answering queries over sets of DBs. This problem has not yet been studied systematically for aggregate queries.

Lechtenbörger et al. [10] investigated aggregate queries over conditional tables, which are a formalism to specify incomplete databases, and showed that in this case the answers to an aggregate query can be represented again by a conditional table. Arenas et al. [4] studied aggregate queries over the set of repairs of an inconsistent database. They introduced a so-called *range-semantics*, according to which a query is evaluated over each individual instance (that is, each repair), and returns the minimal and the maximal value thus obtained. Afrati and Kolaitis [3] gave a semantics and algorithms for aggregated queries in data exchange, which crucially exploit the specific characteristics of this setting.

As a preparation for the following sections, we will argue that an approach that generalizes the standard certain answer semantics to aggregate queries will not be satisfactory, since in most case the set of certain answers will be empty.

Recall the question (c) from Table 1, it can be expressed as

$$q_3(x, sum(y)) \leftarrow \mathrm{CTR}(x, y), \mathrm{HasChild}(x, z). \qquad (1)$$

In SQL syntax, the query is

```
SELECT   HasChild.PName, SUM(CTR.Amount)
FROM     HasChild, CTR
WHERE    HasChild.PName = CTR.PName
GROUP BY HasChild.PName
```

EXAMPLE 4.1. Consider the ontology $\mathcal{K}_1$ that has an empty TBox and whose ABox is D_f from Example 2.1. Because of the open-world semantics of ontologies, the knowledge base $\mathcal{K}_1$ says that, for instance, Luisa has at least two children, Beppe and Paolo, and that it is unclear whether she has a car.

The certain answer semantics for queries over knowledge bases requires one first to evaluate a query q over each database (model) of the knowledge base and then to intersect the answer sets obtained. Since the knowledge base $\mathcal{K}_1$ does not "know" all children of Luisa and Anna, in different models of $\mathcal{K}_1$ the numbers of their children differ and, consequently, the reductions for the two women differ. Therefore, the set of certain answers $Cert(q, \mathcal{K}_1)$ is empty. ∎

The proposition below generalizes the example to knowledge bases where either the Tbox or ABox are empty.

PROPOSITION 4.2. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a knowledge base and q an aggregate query. Then $Cert(q, \mathcal{K}) = \emptyset$ if $\mathcal{T} = \emptyset$ or if $\mathcal{A} = \emptyset$.*

Clearly, $Cert(q, \mathcal{K}) = \emptyset$ holds not only in the two extreme cases of the proposition. It holds, intuitively, whenever the ontology does not restrict the instances extending an ABox so much that at least one group has the same aggregate value in each instance.

5. EPISTEMIC AGGREGATE QUERIES OVER ONTOLOGIES

In this section we propose a syntax and semantics of *epistemic aggregate queries* and give intuitions why they solve the problems with the semantics discussed in Section 4.

5.1 Motivation

Reconsidering Example 4.1, we notice that certain answer semantics does not exploit the fact that in every instance D that is a model of $\mathcal{K}_1$, Luisa has two children for sure and Anna has one child for sure. Consequently, Luisa will receive a tax reduction of 300 EUR for the two children named in the ABox and Anna one of 100 EUR for the one child named. Clearly, in a specific instance the total reduction for Anna may vary, but there will always be a component of 300 EUR, due to Mario and Paolo, her two *known* children. Similarly, for Anna, there will always be a component of 100 EUR, due to her known child Beppe. If we could tell our query engine that we only want the sum of the known reductions for the known children, then the engine could return as answers to query q_3 the set $\{(\text{Luisa}, 300), (\text{Anna}, 100)\}$.

Note that in this case the answers to the query could be obtained by first computing the certain answers of the conjunctive query

$$q_3'(x, y) \leftarrow \mathrm{CTR}(x, y), \mathrm{HasChild}(x, z),$$

then grouping the result according to values of x, and finally computing the aggregate $sum(y)$ for each group.

The next example shows this semantics is too restrictive.

EXAMPLE 5.1. Consider the knowledge base $\mathcal{K}_2$, which has the following ontology:

$$\exists\mathrm{CTR} \sqsubseteq \exists\mathrm{HasChild}, \qquad (\mathsf{funct}\ \mathrm{HasChild}), \qquad (\mathsf{funct}\ \mathrm{CTR}).$$

The first constraint is satisfied by a database instance if for everyone who receives a Children Tax Reduction a child is registered in the database. The second constraint is satisfied by a database instance if for everyone at most one child is registered in the database. The last constraint says that at most one CTR is assigned to anyone. One could imagine that the databases reflect the situation of a country with a birth control policy, where only one child would give the parents a tax reduction. Consequently, at most one child needs to be registered in a tax reduction database.

Suppose the ABox of $\mathcal{K}_2$ is as follows:

CTR

Name	Amount
Luisa	150
Anna	100

HasChild

PName	CName
Anna	Mario

According to the knowledge base, there is one known child of Anna, namely Mario, but no known child of Luisa. However, due to the first constraint, it is known that Luisa has a child, although it is not known who that child is. ∎

If we want to identify the tax reductions that necessarily hold for every database instance that is a model of $\mathcal{K}_2$, it is not sufficient to consider only the known children, as in the previous example, but also which children are known to exist for each person.

Reasoning with the information in the ABox and in the ontology, a query processor could find out that Luisa receives a reduction of 150 EUR and Anna a reduction of 100 EUR. Actually, in this case $\{(\text{Luisa}, 150), (\text{Anna}, 100)\}$ is the set of certain answers to the query q_3 over the KB $\mathcal{K}_2$.

To summarize the intuitions of the semantics for aggregate queries we presented in the examples, in the case of $\mathcal{K}_1$ we (1) calculated certain answers for the query q_3', the body of which contains the the same condition as q_3, and the distinguished variables of which are $Var(q)$ and (2) aggregated over the certain answers. In the example with the KB $\mathcal{K}_2$ we (1) used certain answers for the grouping variables $\bar{x}$ and the aggregation variable y and (2) applied reasoning over the ontology for the variable z, which varies over the children of tax reduction holders, in order to perform the aggregation.

5.2 Epistemic Aggregate Queries

In order to capture both of these cases we introduce epistemic aggregate queries. An *epistemic aggregate query* is a query of the form:

$$q(\bar{x}, \alpha(\bar{y})) \leftarrow \mathbf{K}\bar{x}, \bar{y}, \bar{z}.\ \phi, [\psi], \tag{2}$$

where $\bar{z}$ is a possibly empty list of distinct existential variables of ϕ and ψ such that $\bar{z}$ is disjoint from $\bar{x}$ and $\bar{y}$.

We call $\mathbf{K}$ an *epistemic operator* (where the letter "K" stands for "known") and the variables in the list following $\mathbf{K}$ are the *epistemic* variables or $\mathbf{K}$-variables of the query.

EXAMPLE 5.2. We introduce query q_4 as an epistemic variant of query q_3 in (1)

$$q_4(x, sum(y)) \leftarrow \mathbf{K}x, y.\mathrm{CTR}(x, y), \mathrm{HasChild}(x, z), \tag{3}$$

Note that this query has no nested conditions and the only epistemic variables are x and y. ∎

5.3 Semantics of EAQs over Ontologies

We define the semantics of epistemic aggregate queries for arbitrary knowledge bases.

Consider an epistemic aggregate query q as in (2) and a KB $\mathcal{K}$. Let $\bar{w} := \bar{x} \cup \bar{y} \cup \bar{z}$. A tuple of constants $\bar{c}$ is a *known solution* for $\bar{w}$ in ϕ, ψ over $\mathcal{K}$ if $(\bar{c})$ is a certain answer for the query

$$aux_q(\bar{w}) \leftarrow \phi, \psi \tag{4}$$

over $\mathcal{K}$. The semantics of q over $\mathcal{K}$ will be defined in such a way that for every model D of $\mathcal{K}$ the query q is evaluated considering only assignments that map $\bar{w}$ to known solutions. The epistemic certain answers for q over D are then obtained by taking the intersection of the answer sets for all models D of $\mathcal{K}$.

Formally, let $\bar{w} := \bar{x} \cup \bar{y} \cup \bar{z}$ and $\bar{v} := \bar{w} \cap Var(\phi)$, that is, $\bar{w}$ consists of all epistemic variables of q and $\bar{v}$ of all epistemic variables of ϕ. For a model D of $\mathcal{K}$ we define the set $KSat_{D,\mathcal{K}}(\bar{w}; \phi, \psi)$ of *satisfying* $\mathbf{K}$*-assignments* of ϕ, ψ over D as

$$KSat_{D,\mathcal{K}}(\bar{w}; \phi, \psi) := \{\gamma \in Sat_D(\phi, \psi) \mid \gamma(\bar{w}) \in Cert(aux_q, \mathcal{K})\}.$$

Note that all satisfying $\mathbf{K}$-assignments map $\bar{w}$ to known solutions. Analogously to the standard semantics, we define the set $KSat^{\psi}_{D,\mathcal{K}}(\bar{v}; \phi)$ of all $\mathbf{K}$-*assignments* of ϕ over D *that respect* ψ as

$$KSat^{\psi}_{D,\mathcal{K}}(\bar{v}; \phi) := \{\gamma_{|_{Var(\phi)}} \mid \gamma \in KSat_{D,\mathcal{K}}(\bar{w}; \phi, \psi)\}.$$

The multiset $H_{\bar{d}}$, that we call $\bar{d}$*-group*, that consists of satisfying assignments from $KSat^{\psi}_{D,\mathcal{K}}(\bar{z}; \phi)$ we define as:

$$H_{\bar{d}} := \{\!\{\gamma(y) \mid \gamma \in KSat^{\psi}_{D,\mathcal{K}}(\bar{z}; \phi) \text{ and } \gamma(\bar{x}) = \bar{d}\}\!\}. \tag{5}$$

We say that $(\bar{d}, d)$ is a $\mathbf{K}$-*answer* for an epistemic query q over D if there is $\gamma \in KSat^{\psi}_{D,\mathcal{K}}(\bar{v}; \phi)$ such that

$$\bar{d} = \gamma(\bar{x}) \qquad \text{and} \qquad d = \alpha(H_{\bar{d}}).$$

We denote the set of *all* $\mathbf{K}$-answers for q over a model D of $\mathcal{K}$ as $q(D, \mathcal{K})$.

We define *epistemic certain answers for an epistemic aggregate query* q over a knowledge base $\mathcal{K}$, denoted $ECert(q, \mathcal{K})$, as follows:

$$ECert(q, \mathcal{K}) := \bigcap_{D \in \mathcal{K}} q(D, \mathcal{K}).$$

EXAMPLE 5.3. Consider again the epistemic *sum*-query q_4 from the knowledge base represented by $\mathcal{K}_1$ as in Example 4.1. The certain answers $Cert(q'_3, \mathcal{K}_1)$ are

$$\{(\text{Anna}, 100, \text{Mario}), (\text{Luisa}, 150, \text{Beppe}), (\text{Luisa}, 150, \text{Paola})\}$$

Then $KSat_{D,\mathcal{K}}(\bar{w}; \phi, \psi)$ is $\{\gamma_1, \gamma_2, \gamma_3\}$ where

$$\begin{array}{lcl} \gamma_1 & := & [x/\text{Luisa}, y/150, z/\text{Beppe}], \\ \gamma_2 & := & [x/\text{Luisa}, y/150, z/\text{Paola}], \\ \gamma_3 & := & [x/\text{Anna}, y/100, z/\text{Mario}]. \end{array}$$

Since x is q_3's grouping variable, this means we get two $\mathbf{K}$-groups, the group of Anna and the group of Luisa, namely, $G_{Anna} = \{\!\{100\}\!\}$ and $G_{Luisa} = \{\!\{150, 150\}\!\}$. Recall that *sum* returns the sum of the values of the groups. Therefore,

$$ECert(q_3, \mathcal{K}_1) = \{(\text{Anna}, 100), (\text{Luisa}, 300)\}.$$ ∎

6. EVALUATING EAQS OVER DL-LITE$_{\mathrm{A}}$ ONTOLOGIES

Let q be an AC and $\mathcal{T}$ be an TBox. As we discussed in Section 4, with the classical approach, for some q and $\mathcal{T}$, the set of certain answers $Cert(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ is empty for every ABox $\mathcal{A}$. The same can happen for epistemic aggregate queries. We say that an EAC q is *trivial* for a TBox $\mathcal{T}$ if for all ABoxes $\mathcal{A}$ it holds that $ECert(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \emptyset$.

Obviously, every query that is non-coherent with $\mathcal{T}$ is trivial for $\mathcal{T}$. In this section, we consider only coherent pairs of queries and *DL-Lite*$_{\mathcal{A}}$ ontologies and identify conditions under which these queries are non-trivial for the ontologies. We also present algorithms for computing epistemic certain answers for such queries.

In the following we consider a generic EAQ q which has the form: $q(\bar{x}, \alpha(\bar{y})) \leftarrow \mathbf{K}\bar{x}, \bar{y}, \bar{z}.\ \phi, [\psi]$.

6.1 General Algorithm

In this section we introduce an aggregation algorithm that we will use as the basic one for computing epistemic certain answers $ECert(q, \mathcal{K})$ for the various aggregation functions α. We denote the algorithm `GA` (standing for General Algorithm). `GA` takes as input a conditional aggregate query q and a KB $\mathcal{K}$, and computes a set of tuples $O = \mathtt{GA}(q, \mathcal{K})$.

We denote the sequence of non-distinguished epistemic variables of ϕ as $\bar{z}^{\phi}$. In the algorithm, we denote a predicate (view) of arity $|\bar{x} \cup \bar{y} \cup \bar{z}|$ that "stores" the tuples of certain answers $Cert(aux_q, \mathcal{K})$ for aux_q defined in (4) as $\mathtt{Cert}(\mathtt{aux_q}, \mathcal{K})$, consequently, $\mathtt{Cert}(\mathtt{aux_q}, \mathcal{K})(\bar{x}, \bar{y}, \bar{z})$ is an atom with this predicate name. We also denote as $\mathtt{q_0}$ a predicate (view) that "stores" outputs of q_0.

The algorithm `GA` is defined in Figure 2 and it basically applies two steps on the tuples in $Cert(aux_q, \mathcal{K})$: (i) it projects

Input:	epistemic aggregate query	q
	DL-Lite$_\mathcal{A}$ knowledge base	$\mathcal{K}$
Output:	set of tuples	O
Do:	build q_0 :	$q_0(\bar{x}, \bar{y}, \bar{z}^\phi) \leftarrow \mathtt{Cert}(\mathtt{aux_q}, \mathcal{K})(\bar{x}, \bar{y}, \bar{z})$
	build q_1 :	$q_1(\bar{x}, \alpha(y)) \leftarrow \mathtt{q}_0(\bar{x}, \bar{y}, \bar{z}^\phi)$
	compute certain answers:	$D_1 := \mathit{Cert}(\mathit{aux}_q, \mathcal{K})$
	project on **K**-variables:	$D_2 := q_0^{D_1}$
	aggregate:	$O := q_1^{D_2}$

Figure 2: General algorithm

them on components corresponding to epistemic variables of ϕ and (ii) it aggregates over the result.

Notice that GA projects out non-epistemic variables of $\mathit{Var}(\phi)$ and, consequently, GA loses multiplicity of their values. This may obviously lead to an aggregation that is wrong according to the epistemic certain answers semantics. In the following we show that this does not lead to wrong aggregation if (i) the function α is not sensitive to multiplicities and/or (ii) all the projected out variables do not contribute to multiplicities due to involvement in functional dependencies.

6.2 Queries with Restricted Variables

Let v and w be variables from $\mathit{Var}(q)$ and $\mathcal{T}$ be a *DL-Lite*$_\mathcal{A}$ TBox. We say that v *directly (functionally) depends* on w in $\mathcal{T}$ if either

- there is $R(w, v)$ in q and (funct R) in $\mathcal{T}$, or
- there is $R(v, w)$ in q and (funct R^-) in $\mathcal{T}$.

We say that v *(functionally) depends* on w in $\mathcal{T}$ if there is a sequence of variables $v = v_1, \ldots, v_n = w$ such that each $v_i \in \mathit{Var}(q)$ and each v_i directly depends on v_{i+1}.

We say that a variable occurring in ϕ is *restricted* by $\mathcal{T}$ if it is epistemic or depends on an epistemic variable of q in $\mathcal{T}$. We say an epistemic aggregate query q is *restricted* by $\mathcal{T}$ if all the variables in ϕ are restricted by $\mathcal{T}$.

The following theorem says that GA computes *ECert* for restricted queries.

THEOREM 6.1 (RESTRICTED QUERIES). *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{A}$ TBox and q be a coherent epistemic aggregate query for $\mathcal{T}$. Then if q is restricted by $\mathcal{T}$ then*

- *q is not trivial for $\mathcal{T}$, and*
- *$\mathit{ECert}(q, \langle \mathcal{T}, \mathcal{A}\rangle) = \mathtt{GA}(q, \langle \mathcal{T}, \mathcal{A}\rangle)$, for any $\mathcal{A}$.*

6.3 Count Queries

It turns out that restricted epistemic *count*-queries are the only non-trivial *count*-queries for *DL-Lite*$_\mathcal{A}$ ontologies.

THEOREM 6.2 (COUNT QUERIES). *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{A}$ TBox and q be a coherent epistemic count-query for $\mathcal{T}$. Then q is restricted by $\mathcal{T}$ if and only if*

- *q is not trivial for $\mathcal{T}$ and*
- *$\mathit{ECert}(q, \langle \mathcal{T}, \mathcal{A}\rangle) = \mathtt{GA}(q, \langle \mathcal{T}, \mathcal{A}\rangle)$, for any $\mathcal{A}$.*

6.4 Min, Max and Count Distinct Queries

It turns out GA computes all the epistemic certain answers for *min*-, *max*- and *cntd*- queries even if they are non-restricted.

THEOREM 6.3 (MIN, MAX, COUNT DISTINCT QUERIES). *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{A}$ TBox and q be a coherent epistemic min-, max- or cntd-query for $\mathcal{T}$. Then*

- *q is not trivial for $\mathcal{T}$ and*
- *$\mathit{ECert}(q, \langle \mathcal{T}, \mathcal{A}\rangle) = \mathtt{GA}(q, \langle \mathcal{T}, \mathcal{A}\rangle)$, for any $\mathcal{A}$.*

In fact a more general statement holds, that is, $\mathit{ECert}(q, \mathcal{K}) = \mathtt{GA}(q, \mathcal{K})$ holds for any KB $\mathcal{K}$ and any EAC q with α insensitive to multiplicities of the values for y.

6.5 Sum Queries

It turns out that the only case when a non-restricted *sum*-query is non-trivial for a *DL-Lite*$_\mathcal{A}$ TBox $\mathcal{T}$ is when there is an ABox $\mathcal{A}$ such that all the values of y in at least one group $H_{\bar{d}}$ (defined by Equation 5) are 0. For a given q and $\langle \mathcal{T}, \mathcal{A}\rangle$, existence of such a group $H_{\bar{d}}$ can be found out by the following two queries. The first query $\mathit{trash}(\bar{x})$ accumulates grouping values of groups that contain not zero values for y:

$$\mathit{trash}(\bar{x}) \leftarrow \mathtt{Cert}(\mathtt{aux_q}, \mathcal{K})(\bar{x}, \bar{y}, \bar{z}), y \neq 0.$$

The second query $\mathit{ans}(\bar{x}, 0)$ drops the values returned by *trash* from the set of all grouping values in $\mathit{Cert}(\mathit{aux}_q, \mathcal{K})$:

$$\mathit{ans}(\bar{x}, 0) \leftarrow \mathtt{Cert}(\mathtt{aux_q}, \mathcal{K})(\bar{x}, \bar{y}, \bar{z}), \mathtt{not}\ \ \mathit{trash}(\bar{x}). \quad (6)$$

In the queries inequality "$\neq$" and negation not intuitively work as "$\neq$" and MINUS in SQL (see [1] for details).

The following theorem shows how to compute $\mathit{ECert}(q, \mathcal{K})$ for non-restricted epistemic *sum*-queries.

THEOREM 6.4 (SUM QUERIES). *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{A}$ TBox and q be an epistemic sum-query that is coherent but non-restricted for $\mathcal{T}$. Then*

$$\mathit{ECert}(q, \mathcal{K}) = \mathit{ans}^{\mathit{Cert}(\mathit{aux}_q, \mathcal{K})},$$

where ans is defined as in Equation 6.

Next proposition deal with y that varies over naturals.

PROPOSITION 6.5 (SUM QUERIES OVER NATURALS). *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{A}$ TBox and q be a conditional sum query that is non-restricted for $\mathcal{T}$. Then*

$$\mathit{ECert}(q, \mathcal{K}) = \{(\bar{d}, 0) \mid (\bar{d}, 0) \in \mathtt{GA}(q, \mathcal{K})\}.$$

6.6 Average Queries

It turns out that for the epistemic *avg*-queries computation of epistemic certain answers is more involved than in all the previous cases. The reason is that the average of k and l copies of a number k is k in both cases,

Let M be a multi-set and k be a natural number. We denote $k * M$ a multi-set obtained from M by duplicating k times each element in M. For example, if $M = \{\!\{1, 1, 3\}\!\}$ and $k = 2$, then $k * M = \{\!\{1, 1, 1, 1, 3, 3\}\!\}$. We denote the average of all the elements occurring in M as $avg(M)$. The following proposition says that avg is insensitive for uniform enlargements of multisets.

PROPOSITION 6.6. *Let M be a multi-set. Then*

$$avg(M) = avg(k * M), \text{ for any natural number } k.$$

Let q' be as follows:

$$q'(\bar{x}) \leftarrow \texttt{Cert}(\texttt{aux}_\texttt{q}, \mathcal{K})(\bar{x}, \bar{y}, \bar{z}),$$

that is, q' projects the certain answers of the auxiliary query aux_q on the grouping variables.

We say that q has a *uniform grouping* wtr $\mathcal{T}$ if for every ABox $\mathcal{A}$ and every $\bar{d} \in q'^{ECert(q,\langle\mathcal{T},\mathcal{A}\rangle)}$, there exists a multiset M such that for every database instance D compatible with $\langle\mathcal{T}, \mathcal{A}\rangle$, there is a natural number k such that $H_{\bar{d}} = k * M$ (where $H_{\bar{d}}$ defined as in Equation 5).

THEOREM 6.7 (UNIFORM GROUPING). *Let q be an epistemic avg-query, $\mathcal{T}$ be a DL-Lite$_\mathcal{A}$ TBox. If q has a uniform grouping wrt $\mathcal{T}$ then q is non-trivial for $\mathcal{T}$.*

We identify syntactic conditions on q and $\mathcal{T}$ that guarantee that q has a uniform grouping and, hence, non-trivial for $\mathcal{T}$.

We say that an epistemic aggregate query q is *decomposable* for $\mathcal{T}$ if ϕ can be partitioned into ϕ_1, ϕ_2 such that

- y occurs only in ϕ_1,
- all variables non-restricted by $\mathcal{T}$ occur only in ϕ_2, and
- $Var(\phi_1) \cap Var(\phi_2) \subseteq \bar{x}$.

Next theorem says decompositionality implies non-triviality.

THEOREM 6.8 (DECOMPOSABLE AVERAGE QUERIES). *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{A}$ TBox and q be an epistemic avg-query. If q is decomposable over $\mathcal{T}$ then*

- *q has uniform grouping wrt $\mathcal{T}$ and*
- *$ECert(q, \langle\mathcal{T}, \mathcal{A}\rangle) = \texttt{GA}(q, \langle\mathcal{T}, \mathcal{A}\rangle)$, for any $\mathcal{A}$.*

7. CONCLUSIONS AND FUTURE WORK

In the paper we investigated the problem of answering conditional aggregate queries over ontologies. We showed that certain answer semantics is not adequate for this purpose. We illustrated this phenomenon on examples and provided intuitions that motivated an alternative, namely epistemic, semantics for conditional aggregate queries over ontologies. In order to capture our semantics we separated variables into two categories, namely epistemic and non-epistemic variables, and extended the syntax of the queries by introducing an epistemic operator **K** that indicates which variables are epistemic.

We investigated syntactic conditions on an epistemic aggregate query q and conceptualization $\mathcal{T}$ that guarantee existence of an instantiation $\mathcal{A}$ of this conceptualization such that (epistemic) answers for q over the ontology constituted by $\mathcal{A}$ and $\mathcal{T}$ are not empty (non-triviality property of q for $\mathcal{T}$). We developed an algorithm, `GA`, that retrieves the answers of q over $\mathcal{A}$ and $\mathcal{T}$ if these conditions are satisfied. For each of the aggregation functions in the list: `MAX, MIN, COUNT, CNTD, SUM, AVG`, we found specific conditions that guarantee non-triviality of q and we provided ways to retrieve the answers.

In the future we plan to consider aggregate queries over ontologies in different ontology languages. We also plan to implement our algorithms and test them.

8. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995.

[2] A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QUONTO: QUerying ONTOlogies. In *Proc. of AAAI 2005*, pages 1670–1671, 2005.

[3] F. Afrati and P. Kolaitis. Answering aggregate queries in data exchange. In *Proc. of PODS 2008*, 2008.

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

[5] A. Cali, D. Calvanese, G. D. Giacomo, and M. Lenzerini. Data integration under integrity constraints. *Information Systems*, (29), 2004.

[6] A. Cali, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete information. In *PODS '03: Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 2003.

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

[8] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic SHIQ. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 07)*, 2007.

[9] I. Horrocks and S. Tessaris. Querying the semantic web: A formal approach. In *Proceedings of the 13th International Semantic Web Conf. (ISWC 2002)*, 2002.

[10] J. Lechtenbörger, H. Shu, and G. Vossen. Aggregate queries over conditional tables. *J. Intell. Inf. Syst.*, 19(3):343–362, 2002.

[11] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.

[12] R. Rosati. The limits of querying ontologies. In *Proceedings of the Eleventh International Conference on Database Theory (ICDT 2007)*, 2007.