Open Access

Open Journal of Semantic Web (OJSW)
Volume 5, Issue 1, 2018
http://www.ronpub.com/ojsw
ISSN 2199-336X

Count Distinct Semantic Queries over Multiple Linked Datasets

Bogdan Kostov, Petr Křemen

Department of Cybernetics, Czech Technical University in Prague,
Technicka 2, 16000 Prague, Czech Republic, {bogdan.kostov, petr.kremen }@fel.cvut.cz

Abstract

In this paper, we revise count distinct queries and their semantics over datasets with incomplete knowledge, which is a typical case for the linked data integration scenario where datasets are viewed as ontologies. We focus on counting individuals present in the signature of the ontology. Specifically, we investigate the Certain Epistemic Count (CEC) and the Possible Epistemic Count (PEC) interval based semantics. In the case of CEC semantics, we propose an algorithm for its evaluation and we prove its correctness under a practical constraint of the queried ontology. We conduct and report experiments with the implementation of the proposed algorithm. We also prove decidability of the PEC semantics.

Type of Paper and Keywords

Regular research paper: *semantic queries, aggregate queries, conjunctive queries, distinct count, linked data, ontology, incomplete knowledge, certain epistemic count, possible epistemic count*

1 Introduction

Reflecting knowledge incompleteness in aggregate queries with their different interpretations (different strategies employed when counting incomplete data) over linked data sets is in its early stages of research, lacking appropriate algorithms, tools as well as benchmark support. We believe that such queries are vital in business intelligence and data quality assessment use-cases where integrated incomplete data is analyzed. We attempt to bring some practical results to this interesting research area.

In this paper, we extend the work [12]. We investigate the possibility to implement different interpretations of count distinct queries over ontologies where the *Unique Name Assumption*[1] (UNA) does not hold, which is a typical case for the linked data integration scenario. We focus on counting individuals present in the signature of the ontology.

Specific contributions of the paper include:

- a novel interpretation labeled *possible epistemic count* (PEC) for count distinct queries and we prove its decidability, see Section 4,
- an algorithm implementing the *certain epistemic count* (CEC) interpretation and proof of its correctness under a practical constraint on the queried ontology, see Section 5,
- experimental evaluation of the algorithm over artificial data, see Section 6.

This paper is organized as follows. In Section 2 we present a motivational use-case for semantic count distinct queries over multiple Linked Data (LD) datasets in the domain of aviation safety. We also discuss the

[1] The unique Name Assumption (UNA) is a convenient assumption in which differently named individuals are interpreted as different domain elements.

Table 1: Safety event report dataset D_1 from Prague Airport

ID	Location	Incident Time	Report Time	FN
1	Prague	12h	12h	A
2	Prague	11h	11h	B
3	-	Not-sure	15h20	C
4	-	Not-sure	10h	D

types of incompleteness that we found in the data and conclude by identifying two suitable interpretations of count distinct queries. In Section 3 we formally define the syntax and semantics of OWL 2 - DL ontologies, semantic conjunctive queries and count distinct queries with basic semantics. In Section 4 we define the PEC semantics of count distinct queries and prove its decidability. In Section 5 we propose an algorithm for the CEC semantics of count distinct queries and we prove its soundness and completeness under a practical constraint on the queried ontology. In Section 6 we report the results of experiments done with the proposed algorithm. We experiment with artificial data samples. In Section 7 we present the state of art. Section 8 summarizes our results and discusses future work.

2 Motivational Use-Cases

We start this section with a discussion about a simple use-case scenario for semantic counting in the domain of aviation safety. We also show how two types of data incompleteness affect the evaluation of the interpretation of the count distinct aggregate function in different scenarios.

One of the strategies in managing aviation safety is through monitoring of reactive indicators. An indicator is typically defined as a statistic in a time frame (a month) over a data set of reported safety events and investigation results. The simplest and also one of the most common indicators are based on count distinct statistics, e.g., the number of bird strike events (e.g., a collision of a bird with an airplane) per month. The precision of this statistics is essential for the efficient and effective performance of safety departments. We identify errors when evaluating these statistics over data sets caused by incompleteness of the collected data. Consider the following example data. Table 1 and 2 show reports of bird strike events collected respectively at two fictive organizations – Prague Airport and Brno Airline. Table 3 shows information about individual flights. The column with name **FN** refers to the flight number.

Each table represents an autonomous dataset. Note that the datasets do not contain events but their reports. Based on our experience with the industry we assume that in an autonomous dataset of event reports the identity of reports is used to represent the identity of events and that each event is reported only once. These conditions are achieved by data collection procedures and data management processes. We can see the first incompleteness in the second and third columns, (“Location” and “Incident Time”) in Table 1 where the values are not always specified. This incompleteness of attribute values occurs naturally. A bird strike event is typically reported by the pilot. The pilot may not perceive the strike at the moment of the occurrence. Hence the time and location of the strike might not be known. As a result, the data collected in the dataset contains incomplete information about the time and location of the incidents. Note also that in case of a bird strike event the time and location of the event are bound and given the time we know the location and vice versa. Consider the indicator defined in Example 1.

Table 2: Safety event report dataset D_2 from Brno Airline

ID	Location	Incident Time	Report Time	FN
1	Prague	12h	12h20	A
4	-	Not-sure	9h10	D

Example 1. ***Bird strike indicator***

- *Number of bird strikes over Prague airport for a month.*

Commonly, this indicator is evaluated by counting the number of bird strike reports in the dataset. Evaluating the indicator in this way over the dataset from Table 1 gives a result of 4 and over the dataset obtained by merging the data from Tables 1 and 2 gives 5. Let’s investigate and identify the errors which occur using this evaluation strategy. To compute this indicator correctly, the queried dataset needs to comply with the following conditions:

- unique identification for events and
- a definite classification of events.

Let’s first consider evaluation over Table 1. The first condition is satisfied because the identity of events is complete, as discussed earlier. The second condition, however, definite classification, is not met. In this case classifying a bird strike as ”over Prague airport” or not is based on the attribute ”Location”. Assuming that the time and location are dependent attributes and the location of the incident is not known, then given that the incident occurred within a certain interval before the landing, the event can satisfy the condition to be

Table 3: Dataset describing flights

FN	Departure	Arrival	DepTime	ArrTime	Airline
A	Prague	Brno	10h	10h30	Brno Airlines
C	Paris	Prague	13h	15h	—
D	Brno	Prague	8h	8h30	Brno Airlines

classified as ”over Prague airport”. The problem is, as mentioned earlier, that neither the time nor the location of some reports are known. For example, the incident reported at row 3 in Table 1 may have occurred in Paris and incident reported at row 4 may have occurred in Brno. In this case, the indicator should be evaluated as a range $\langle 2..4 \rangle$ which represents the incompleteness present in the data as opposed to the conventional evaluation which would result in a definite answer – 4.

Let's investigate the evaluation over the merge of datasets in Tables 1 and 2. The second condition is violated because the incompleteness of the attributes ”Location” and ”Incident Time” is inherited from the merged data. Furthermore, the first condition is also violated because as opposed to the autonomous datasets the merged dataset is not managed in the same manner and thus may contain more than one report for a given event. For example, the event reported at first and forth rows in Table 1 are the same events as the ones reported at first and second rows in Table 2. Note that the merged datasets does not contain this information. In this case, a correct evaluation should return the range $\langle 2..6 \rangle$ as opposed to the definite value 3 returned by the common evaluation strategy.

Instance reconciliation may be used to improve data quality. This process produces a new dataset containing information about which records report the same event. This mapping is implemented by a discriminative attribute which enables grouping of reports of the same event in one group or using relational statements (table1/report#4 owl: sameAs table2/report#1). In the first approach, we can use distinct count with UNA interpretation to evaluate correctly the indicator from Example 1. The second case is quite typical for the semantic web. However, the correct implementation of the indicator using distinct count with UNA interpretation needs to be extended to calculate the partitions defined by the owl: sameAs statements. In both cases, SPARQL 1.1 can be used to evaluate this indicator correctly. However, in the second case, the query will be more complicated because it will need to calculate the equality partitions induced by the owl: sameAs axioms. Because of the added information and the use of a suitable evaluation, this method will return the value 2.

3 Preliminaries

In this section, we will define basic terms and notions used in the rest of the paper. We will start with the definition of an ontology followed by the definition of conjunctive queries and aggregate queries with *distinct count* function.

3.1 Ontology

In this paper we consider the description logic language $\mathcal{SROIQ}(\mathcal{D})$ for ontology representation. We will use the term *ontology* as a shorthand of *a* $\mathcal{SROIQ}(\mathcal{D})$ *ontology*. Next, we will introduce part of the syntax and semantics of the language, relevant to our work. We will also introduce some syntactic sugar used in OWL 2 Web Ontology Language [17, 19], a syntactic variant of $\mathcal{SROIQ}(\mathcal{D})$. For a full description of the syntax and semantics of different description logic formalisms see [1]. Also, note that we assume that the $\mathcal{SROIQ}(\mathcal{D})$ ontologies should comply with all the necessary syntactic restrictions for which inference problems remain decidable.

Definition 1 (Ontology, Selected Syntax and Semantics)**.** *An ontology $\mathcal{O}$ is a pair $\langle S, A \rangle$, where S is a signature and A is a set of axioms. The semantics of ontologies uses a first order interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is an interpretation domain and $\cdot^{\mathcal{I}} : S \rightarrow \Delta^{\mathcal{I}}$ is an interpretation function mapping elements from the ontology signature S to elements from the interpretation domain $\Delta^{\mathcal{I}}$. An ontology $\mathcal{O}$ is satisfied by an interpretation $\mathcal{I}$, denoted by $\mathcal{I} \models \mathcal{O}$, if all of its axioms are satisfied by the interpretation $\mathcal{I}$, such interpretation $\mathcal{I}$ is called a model of $\mathcal{O}$. We say that a set of axioms A is entailed by the ontology $\mathcal{O}$, denoted by $\mathcal{O} \models A$, if every model $\mathcal{I}$ of the ontology $\mathcal{O}$, is also a model of A, $\mathcal{I} \models A$.*

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation. Let R, $\mathsf{R}_1, \mathsf{R}_2, \cdots, \mathsf{R}_n$ be roles, C, D be concepts, a, b, $\mathsf{a}_1, \mathsf{a}_2, \cdots, \mathsf{a}_n$ be individuals (all previous part of S) and k is a natural number. Selected role axioms and their interpretations are:

- $\mathcal{I} \models \mathsf{Func(R)}$ *if* $\langle \mathsf{x}^{\mathcal{I}}, \mathsf{y}^{\mathcal{I}} \rangle \in \mathsf{R}^{\mathcal{I}}$ *and* $\langle \mathsf{x}^{\mathcal{I}}, \mathsf{z}^{\mathcal{I}} \rangle \in \mathsf{R}^{\mathcal{I}}$ *implies* $\mathsf{y}^{\mathcal{I}} = \mathsf{z}^{\mathcal{I}}$.

- $\mathcal{I} \models \mathsf{InvFunc(R)}$ *if* $\langle \mathsf{y}^{\mathcal{I}}, \mathsf{x}^{\mathcal{I}} \rangle \in \mathsf{R}^{\mathcal{I}}$ *and* $\langle \mathsf{z}^{\mathcal{I}}, \mathsf{x}^{\mathcal{I}} \rangle \in \mathsf{R}^{\mathcal{I}}$ *implies* $\mathsf{y}^{\mathcal{I}} = \mathsf{z}^{\mathcal{I}}$.

Note that these axioms are only syntactic sugar and in $\mathcal{SROIQ(D)}$ they are represented using TBox axioms.

Selected TBox and ABox axioms and their interpretations are:

- $\mathcal{I} \models C \sqsubseteq D$ *if* $\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$
- $\mathcal{I} \models \mathsf{HasKey(C, R_1, R_2, \cdots, R_n)}$ *if* $\forall \mathsf{x}^{\mathcal{I}}, \mathsf{y}^{\mathcal{I}} \in C^{\mathcal{I}}$ *and* $\forall \mathsf{w}_1^{\mathcal{I}}, \mathsf{w}_2^{\mathcal{I}}, \cdots, \mathsf{w}_n^{\mathcal{I}}$ *if* $\langle \mathsf{x}^{\mathcal{I}}, \mathsf{w_i}^{\mathcal{I}} \rangle \in \mathsf{R}_i^{\mathcal{I}}$ *and* $\langle \mathsf{y}^{\mathcal{I}}, \mathsf{w_i}^{\mathcal{I}} \rangle \in \mathsf{R}_i^{\mathcal{I}}$ *then* $\mathsf{x}^{\mathcal{I}} = \mathsf{y}^{\mathcal{I}}$.
- $\mathcal{I} \models \mathsf{C(a)}$ *if* $\mathsf{a}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$
- $\mathcal{I} \models \mathsf{R(a, b)}$ *if* $\langle \mathsf{a}^{\mathcal{I}}, \mathsf{b}^{\mathcal{I}} \rangle \in \mathsf{R}^{\mathcal{I}}$

We consider the following concept expressions and their interpretations. Top concept $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$*, bottom concept* $\bot^{\mathcal{I}} = \emptyset$*, intersection* $(\mathsf{C} \sqcap \mathsf{D})^{\mathcal{I}} = \mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}}$*, union* $(\mathsf{C} \sqcup \mathsf{D})^{\mathcal{I}} = \mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}}$*, one of* $(\{\mathsf{a}_1, \mathsf{a}_2, \cdots, \mathsf{a}_n\})^{\mathcal{I}} = \{\mathsf{a}_1^{\mathcal{I}}, \mathsf{a}_2^{\mathcal{I}}, \cdots, \mathsf{a}_n^{\mathcal{I}}\}$*, minimum number restrictions* $(\geq k\,\mathsf{RC})^{\mathcal{I}} = \{\mathsf{x} \mid \sharp\{\mathsf{y} \mid \langle \mathsf{x}, \mathsf{y} \rangle \in \mathsf{R}^{\mathcal{I}}$ *and* $\mathsf{y} \in \mathsf{C}^{\mathcal{I}}\} \geq k\}$*, maximum number restrictions* $(\leq k\,\mathsf{RC})^{\mathcal{I}} = \{\mathsf{x} \mid \sharp\{\mathsf{y} \mid \langle \mathsf{x}, \mathsf{y} \rangle \in \mathsf{R}^{\mathcal{I}}$ *and* $\mathsf{y} \in \mathsf{C}^{\mathcal{I}}\} \leq k\}$*. Where* $\sharp S$ *stands for the size of* S*.*

Examples 2, 3 and 4 show how we can use ABox axioms to construct an ontology for the datasets from Tables 1, 2 and 3. The ontologies capture the type of the incident (e.g., BirdStrike), its location and during which flight it happened. Other data from the tables is ignored. Furthermore, for the sake of space we abbreviate some of the resources: Location – loc; Departure – dep; Arrival – arr; Prague – Prg. In examples 2 and 3, reports are represented as individuals i_k and r_k and the k corresponds to the identifier from column **ID**. Flights are represented as individuals using their flight number (i.e., column **FN**) and are associated with reports using the duringFlight property.

Note, how the lack of location information in tables 2 and 3 is captured by the ontologies in examples 2 and 3 by omitting assertion axioms of the loc property.

Example 2. ***Ontology (ABox) of Prague Airport Report dataset from Table 1***

$\mathsf{BirdStrike}(i1)$, $\mathsf{loc}(i1, Prg)$, $\mathsf{duringFlight}(i1, A)$
$\mathsf{BirdStrike}(i2)$, $\mathsf{loc}(i2, Prg)$, $\mathsf{duringFlight}(i2, B)$
$\mathsf{BirdStrike}(i3)$, $\mathsf{duringFlight}(i3, C)$
$\mathsf{BirdStrike}(i4)$, $\mathsf{duringFlight}(i4, D)$

Example 3. ***Ontology (ABox) of Brno Airlines Report dataset from Table 2***

$\mathsf{BirdStrike}(r4)$, $\mathsf{duringFlight}(r4, D)$

Example 4. ***Ontology (ABox) of the Flight dataset from Table 3***

$\mathsf{Flight}(C)$, $\mathsf{dep}(C, Paris)$, $\mathsf{arr}(C, Prg)$,
$\mathsf{Flight}(D)$, $\mathsf{dep}(D, Brno)$, $\mathsf{arr}(D, Prg)$,

Definition 2 (Inference Problems)**.** *Let $\mathcal{O}$ be an ontology, A be a set of axioms and* C, D *be concepts.*

Consistency checking (CC)**.** *$\mathcal{O}$ is consistent, $CC(\mathcal{O}) = true$, if there is a model $\mathcal{I}$ of $\mathcal{O}$,*

Subsumption. D *subsumes* C*, denotes as* $\mathsf{C} \sqsubseteq \mathsf{D}$*, w.r.t. $\mathcal{O}$, if $\mathcal{O} \models \mathsf{C} \sqsubseteq \mathsf{D}$.*

3.2 Conjunctive Queries

To define *distinct count* queries we first need to define conjunctive queries.

Definition 3 (Conjunctive Query)**.** *We will denote conjunctive queries using the following rule like notation*

$$Q(\bar{x}) \leftarrow \phi(\bar{x}, \bar{z}). \tag{1}$$

The head of the query $Q(\bar{x})$ denotes the name of the query and the result variables $\mathcal{R}_{var}(Q) = \bar{x}$. The body of the query $\phi(\bar{x}, \bar{z})$ is a comma-separated list of query atoms interpreted as a conjunctive query. The allowed atoms are $\mathcal{SROIQ(D)}$ atoms where we allow for distinguished and non-distinguished variables in places of individuals.

The variables $\bar{z}$ are called non-result variables. Result variables must be distinguished. By $\mathcal{V}_{var}(Q)$ we denote the list of all variables in the query. By $\mathcal{V}_d(Q)$ we denote all distinguished variables. A binding $\mu : \mathcal{V}_{var}(Q) \to S$ is a mapping of the variables of the query to elements in the ontology's signature and $Q|_{\mu}$ is the substitution of the variables in Q by the binding μ. Binding of a tuple of variables is denoted by $\mu(\bar{v}) = (\mu(v_1), \ldots, \mu(v_k))$. $M_{Q,\mathcal{O}}$ is the set of all possible bindings of Q w.r.t. $\mathcal{O}$, $\sharp M_{Q,\mathcal{O}} = n^k$ where n is the number of individuals in the signature S of $\mathcal{O}$ and k is the number of distinguished variables of Q.

We call Q a semi-ground query if there are no distinguished variables in the query. A solution to the query Q w.r.t. the ontology $\mathcal{O}$ is a binding μ for which the substitution $Q|_{\mu}$ is a semi-ground query, the body of which is entailed by the ontology (denoted by $\mathcal{O} \models Q|_{\mu}$).

The set of all solution bindings of query Q w.r.t. $\mathcal{O}$ is denoted by $Sat_Q^{\mathcal{O}} = \{\mu \mid \mathcal{O} \models Q|_{\mu}, \mu \in M_{Q,\mathcal{O}}\}$. The result of query Q w.r.t. ontology $\mathcal{O}$ denoted by $Q_{\mathcal{O}}$, is a set of bindings of the result variables $\mathcal{R}_{var}(Q)$, formally $Q_{\mathcal{O}} = \{\mathcal{R}_{var}(\mu) \mid \mu \in Sat_Q^{\mathcal{O}}\}$.

The set of all solution bindings of a query Q w.r.t. an interpretation $\mathcal{I}$ is denoted by $Sat_Q^{\mathcal{I}} = \{\mu \mid \mathcal{I} \models Q|_{\mu}, \mu \in M_{Q,\mathcal{O}}\}$. Finally, the result of query Q

w.r.t. an interpretation $\mathcal{I}$ denoted by $Q_{\mathcal{I}}$, is a set of bindings of the result variables $\mathcal{R}_{var}(Q)$, formally $Q_{\mathcal{I}} = \{\mathcal{R}_{var}(\mu) \mid \mu \in Sat^{\mathcal{I}}_{Q}\}$.

Note that there are certain syntactic limitations of the query body concerning undistinguished variables. For more detail see [11].

Examples 5 shows a conjunctive query using the notation defined above. which will retrieve all bird strike incidents which occurred over the Prague airport.

Example 5. ***Conjunctive Query to Select Bird Strike Incidents Over Prague Airport,***
$Q_1(?inc) \leftarrow \mathsf{BirdStrike}(?inc), \mathsf{loc}(?inc, Prg)$.

3.3 Count Distinct Queries and Semantics

In this section, we define the notion and semantics of the count distinct queries. Particularly, we focus on count distinct queries which retrieve the number of distinct tuples of the result of a conjunctive query Q. We also define some additional terminology and symbols used later in the paper.

First, we define the notation of count distinct queries and their conventional semantics.

Definition 4 (Count Distinct Query)**.** *Let $\mathcal{O}$ be an ontology, Q be an conjunctive query with no negation as failure atoms. We denote a distinct count query over $\mathcal{O}$ as $\mathsf{CD}(Q(\bar{x}), \mathcal{O})$, where CD is the count distinct function. Q is the conjunctive query the results of which are counted. The result variables $\bar{x}$ of Q also specifies the counting variables. Conventionally, count distinct queries are evaluated relying on the UNA principle:*

- *basic semantics* $\mathsf{CD}(Q(\bar{x}), \mathcal{O}) = \sharp(Q_{\mathcal{O}})$

Example 6 shows a representation of the indicator introduced in Example 1 using the notion defined above.

Example 6. ***Count Distinct Query with CEC Semantics***

$\mathsf{CD}(Q_1(?inc) \leftarrow \mathsf{BirdStrike}(?inc), \mathsf{loc}(?inc, Prg), \mathcal{O})$.

Note that in the Definition 6 we count the distinct result bindings $Q_{\mathcal{O}}$. Counting requires comparing bindings. We can view a binding as a set of variable-symbol mapping pairs which can be compared. For example, let binding $\mu_1 = \{(x, a)\}$ is different from binding $\mu_2 = \{(x, b)\}$ and equal to binding $\mu_3 = \{(x, a)\}$. We denote interpretation of bindings as $\mu_1^{\mathcal{I}} = \{(x, a)\}^{\mathcal{I}} = \{(x^{\mathcal{I}}, a^{\mathcal{I}})\}$. When interpreted bindings may compare differently. For example, if $a^{\mathcal{I}} = b^{\mathcal{I}}$ then $\mu_1^{\mathcal{I}} = \mu_2^{\mathcal{I}}$.

Next we define certain epistemic count (CEC) semantics of count distinct queries.

Definition 5 (Certain Epistemic Count)**.** *A count distinct query with CEC semantics is represented as $\mathsf{CD}^{\mathcal{C}}(Q(\bar{x}), \mathcal{O})$.*

The $\mathcal{C}$ in the superscript denotes the semantics mode, i.e., CEC. We define the CEC semantics as follows:

- $\mathsf{CD}^{\mathcal{C}}(Q, \mathcal{O}) = \langle min(\mathbb{N}^{\mathcal{C}}_{Q,\mathcal{O}}), max(\mathbb{N}^{\mathcal{C}}_{Q,\mathcal{O}})\rangle$

where $\mathbb{N}^{\mathcal{C}}_{Q,\mathcal{O}}$ is the set of different sizes of certain answers of Q over $\mathcal{O}$ in different models $\mathcal{I}$ of $\mathcal{O}$, $\mathbb{N}^{\mathcal{C}}_{Q,\mathcal{O}} = \{\sharp((Q_{\mathcal{O}})^{\mathcal{I}}) \mid \mathcal{I} \models \mathcal{O}\}$.

Example 7 shows the representation and the result of the indicator introduced in Example 1 using the notion and semantics defined above.

Example 7. ***Count Distinct Queries***

$\mathsf{CD}^{\mathcal{C}}(Q_1(?inc) \leftarrow \mathsf{BirdStrike}(?inc), \mathsf{loc}(?inc, Prg), \mathcal{O})$.

Evaluating the query with the CEC semantics over the merged datasets from Tables 1 and 2 will result in $\mathsf{CD}^{\mathcal{C}}(Q_1, \mathcal{O}) = \langle 2, 3\rangle$. This is because there are at most three incidents reported over Prague but two of them might be the same.

4 Possible Epistemic Count Semantics

In this section, we will define the Possible Epistemic Count (PEC) semantics for distinct count queries.

Definition 6 (Possible Epistemic Count)**.** *A count distinct query with PEC semantics is represented as $\mathsf{CD}^{\mathcal{P}}(Q(\bar{x}), \mathcal{O})$.*

The $\mathcal{P}$ in the superscript denotes the semantics mode, i.e., PEC. We define the PEC semantics as follows:

- $\mathsf{CD}^{\mathcal{P}}(Q, \mathcal{O}) = \langle min(\mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}}), max(\mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}})\rangle$,

where $\mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}}$ is the set of different sizes of certain answers of Q over $\mathcal{O}$ in different models $\mathcal{I}$ of $\mathcal{O}$, $\mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}} = \{\sharp((Q_{\mathcal{O}})^{\mathcal{I}}) \mid \mathcal{I} \models \mathcal{O}\}$.

Example 8 shows the representation and the result of the indicator introduced in Example 1 using the notion and semantics defined above.

Example 8. ***Count Distinct Query with PEC Semantics***

$\mathsf{CD}^{\mathcal{P}}(Q_1(?inc) \leftarrow \mathsf{BirdStrike}(?inc), \mathsf{loc}(?inc, Prg), \mathcal{O})$.

Evaluating the query with the PEC semantics over the merged datasets from Tables 1 and 2 will result in $\mathsf{CD}^{\mathcal{C}}(Q_1, \mathcal{O}) = \langle 2, 6\rangle$. This is because there are potentially at most 6 incidents reported over Prague. Three of those reports are certainly over Prague. However, two of them might be the same, therefore the minimum is again 2.

Certain and possible answers correspond respectively to the Semantic Tuple Count and Semantic Count introduced in [12], where we also prove decidability of the former. Here we prove that $\mathsf{CD}^{\mathcal{P}}(Q,\mathcal{O})$ is decidable. To prove this we define an alternative to $\mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}}$ which we denote as $\mathbb{N}'$ and then we prove that it is decidable and equivalent to $\mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}}$.

The definition of the PEC interpretation requires to evaluate the number of distinct bindings in each model of the ontology. This is a problem because the ontology may have infinitely many models. Each model $\mathcal{I}$ contributes to the set $\mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}}$ with a number $\sharp((\mathcal{R}_{var}(Sat^{\mathcal{I}}_Q))^{\mathcal{I}})$. The evaluation of that number relies solely on the results of the query Q over the model $\mathcal{I}$, and the equivalence relation of the symbols in the query result encoded in the model.

Note that although there are infinite number models of the ontology the results of the query Q in any of those models belongs to the powerset of the set of all bindings $M_{Q,\mathcal{O}}$, that is there finitely many result sets being counted. The equality relation can also be thought of as a partitioning of the elements in the signature S. Note that the set $E_{\mathcal{O}}$ of all different partitionings encoded in all the models of the ontology is also finite.

Before we define the set $\mathbb{N}'$, we introduce some auxiliary notions. Given M, a potential set of result bindings, we can encode them as a set of axioms as follows $A_M = \mathcal{O} \cup \{Q|_\mu \mid \mu \in M\}$. Given E, a potential partitioning of elements in S we can encode them as a set of axioms as follows $A_E = \bigcup_{e\in E}\{a \doteq b \mid a,b \in e\} \cup \bigcup_{e_1,e_2\in E}\{a \neq b \mid a \in e_1, b \in e_2\}$. Given M and E, we define the ontology extension $\mathcal{O}(M,E)$ as follows $\mathcal{O}(M,E) = \mathcal{O} \cup A_M \cup A_E$.

Let $\mathcal{L} = \{(M,E) \mid M \in M_{Q,\mathcal{O}}, EinE_{\mathcal{O}}\}$ be the set of all pairs of M (potential result bindings composed of elements from the signature S) and E (potential partitioning of those elements). We define the alternative definition of $\mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}}$ as follows.

Definition 7 (Alternative Definition of $\mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}}$). $\mathbb{N}' = \{\sharp(\mathcal{R}_{var}(M)^E) \mid \mathcal{O}(M,E)$ *is consitent* $\wedge\, Sat^{\mathcal{O}(M,E)}_Q = M, (M,E) \in \mathcal{L}\}$.

Lemma 1. *Calculating* $\mathbb{N}'$ *is decidable.*

Proof. To calculate $\mathbb{N}'$ we need to iterate over the finite set $\mathcal{L}$. For each element , $(M,E) \in \mathcal{L}$, we construct $\mathcal{O}(M,E)$, an extension of the original ontology. $\mathcal{O}(M,E)$ is checked for consistency and whether it satisfies the condition $Sat^{\mathcal{O}(M,E)}_Q = M$. If it does, we count the number of interpreted bindings. Each of these steps is decidable. Therefore, calculating the set $\mathbb{N}'$ is decidable. □

Lemma 2. *The set* $\mathbb{N}'$ *is equivalent to the set* $\mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}}$.

Proof. First, we show that $\mathbb{N}' \subseteq \mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}}$. Let n be a number in $\mathbb{N}'$, then $n = \sharp((\mathcal{R}_{var}(M))^E)$. We know that the ontology $\mathcal{O}(M,E)$ is consistent and that it satisfies the condition $Sat^{\mathcal{O}(M,E)}_Q = M$. We can deduce that there exists a model $\mathcal{I} \models \mathcal{O}(M,E)$, such that $M^E = (Sat^{\mathcal{I}}_Q)^{\mathcal{I}}$. Therefore, $n \in \mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}}$.

Second, we show that $\mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}} \subseteq \mathbb{N}'$. Each number n in $\mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}}$ is computed from the interpreted binding set $(Sat^{\mathcal{I}}_Q)^{\mathcal{I}}$ w.r.t a model $\mathcal{I}$ of the ontology. Note that $M' = Sat^{\mathcal{I}}_Q$ belongs to the set $M_{Q,\mathcal{O}}$. Also, note that the partitioning E' of elements in the signature of ontology $\mathcal{O}$ induced by $\mathcal{I}$ is an element from $E_{\mathcal{O}}$. Then, the extended ontology $\mathcal{O}(M',E')$ is consistent and it satisfies the condition $Sat^{\mathcal{O}(M,E)}_Q = M$ which implies that $n \in \mathbb{N}'$.

Therefore, $\mathbb{N}' = \mathbb{N}^{\mathcal{P}}_{Q,\mathcal{O}}$ □

Theorem 1. *Let* $\mathcal{O}$ *be an ontology,* Q *be a conjunctive query. Then calculating the result of* $\mathsf{CD}^{\mathcal{P}}(Q,\mathcal{O})$ *is decidable.*

Proof. The decidability of $\mathsf{CD}^{\mathcal{P}}(Q,\mathcal{O})$ is a direct consequence of Lemmas 1, 2. □

5 Implementation of CEC Semantics for Count Distinct Queries

In this section, we present the implementation of the CEC semantics of count distinct queries described in Section 3.3. The implementation takes in an ontology $\mathcal{O}$ and the underlying conjunctive query Q with result variables as counting variables. The implementation relies on a reasoner which supports conjunctive query answering and consistency checking. Counting over multiple datasets is possible if the datasets are merged, and the schema (ontology) is explicitly included. The resulting dataset, i.e., ontology with TBox, RBox, and ABox, can be used as an input for the counting procedure. The idea behind the implementations is to look for numbers which belong to the result by restricting the possible models of the input ontology and checking whether the restriction is satisfiable.

The algorithm is split into two procedures. First, in Algorithm 1 we define a procedure $\mathsf{CDCSingleColumn}(T,\mathcal{O})$ which calculates the count distinct function with CEC semantics over a set of single variable (single column) bindings.

The input T is a list of elements from the signature of $\mathcal{O}$. The procedure looks for the minimum and maximum by iteratively adding axioms to the ontology. We construct the ontology $\mathcal{O}'_i$ to narrow down its models to those containing at most (respectively at least) i many partitions induced by the equality relation of the

Algorithm 1: Evaluation of count distinct with single counting variable

```
CDCSingleColumn($T, \mathcal{O} = \langle S, A \rangle$)
    // $p, x, C \notin S$
    $\mathcal{O}'$ := $\mathcal{O} \cup \{p(x,t) \mid t \in T\}$
    $a$, $b$ := $-1$
    // find minimum
    for $i$:=$n$ to 1 do
        $\mathcal{O}'_i = \mathcal{O}' \cup \{(\leq i\, p)(x)\}$
        if $CC(\mathcal{O}'_i)$ = $false$
            $a$ := $i$ + 1
            break
    // find maximum
    $\mathcal{O}'$ := $\mathcal{O}' \cup \{C \equiv \{t_1\} \sqcup \{t_2\} \sqcup ... \sqcup \{t_n\}\}$
    for $i$:=1 to $n$ do
        $\mathcal{O}'_i = \mathcal{O}' \cup \{(\geq i\, p \cdot C)(x)\}$
        if $CC(\mathcal{O}'_i)$ = false
            $b$ := $i$ − 1
            break
    return $(a, b)$
```

counted elements encoded in those models. In this implementation, we use number restrictions to achieve this effect. A similar algorithm can be designed using different $\mathcal{SROIQ}(\mathcal{D})$ constructs.

Algorithm 2 shows the implementation of the CEC semantics of count distinct queries $\mathsf{CD}^{\mathcal{C}}(Q, \mathcal{O})$. The algorithm makes use of the conjunctive query answering service of the used reasoner. In the case where the query Q has more than one result variables, the algorithm transforms the problem to a single column problem. The single column problem is evaluated in both cases by calling the procedure $\mathsf{CDCSingleColumn}(T, \mathcal{O})$ from Algorithm 1.

The transformation of multi to single column counting is necessary because in $\mathcal{SROIQ}(\mathcal{D})$ there are no constructs to apply the same restrictions used in the single column case. The idea of the transformation is to create a one-to-one mapping between fresh individuals, one for each result row, with the elements of that row. This is achieved by modeling columns as bijective relations, using HasKey and Func axioms and associating through the column relations, each fresh individual corresponding to a result binding with the values of the bound variables.

5.1 Practical Correctness of CDC

In this section, we prove the correctness of Algorithm 2. First, we define the required practical restrictions of the queried ontology.

Definition 8 (Ontology Restrictions)**.** *Let S' be the set of elements composing the counted bindings. An ontology*

Algorithm 2: Evaluation of count distinct with multiple counting variables

```
CDC($Q, \mathcal{O}$)
    // $T$ is a multiset of $n$ $k$-sized tuples
    // of individuals from $S$
    $T$ := evalQuery($Q, \mathcal{O}$)
    if $\sharp\mathcal{R}_{var}(Q) > 1$ // $k > 1$, tuple counting
        $\mathcal{O}'$ := $\mathcal{O} \cup$
            $\{\mathsf{HasKey}(\mathsf{A}, \mathsf{c_1}, \mathsf{c_2}, \cdots, \mathsf{c_k})\} \cup$
            $\{\mathsf{Func}(\mathsf{c_1}), \mathsf{Func}(\mathsf{c_2}), \cdots, \mathsf{Func}(\mathsf{c_k})\} \cup$
            $\{\mathsf{A}(r_i), \mathsf{c_1}(r_i, t_{i,1}), \cdots, \mathsf{c_k}(r_i, t_{i,k}) \mid i \in \langle 1, n \rangle\}$
        $T'$ := $\{r_1, r_2, \cdots, r_n\}$
        return CDCSingleColumn($T', \mathcal{O}'$)
    else // $k = 1$, element counting
        return CDCSingleColumn($T, \mathcal{O}$)
```

can be queried using Algorithm 2 if any of the following axioms are not entailed by the queried ontology.

- $\top \sqsubseteq \forall \top \cdot \bot$
- $\forall \top \cdot \bot(s), \forall s \in S'$

The first restriction allows us to add fresh properties and associate fresh individuals with them. The second restriction allows us to associate the counted elements with other individuals using fresh properties. Note that these restrictions will be satisfied in most practical ontologies. For example, if an ontology entails the first axiom in Definition 8, it will have only models in which no individual is related to any other individual. In the case of our running example of incident reports, using such an ontology will not allow to associate incidents with flights. The second axiom in Definition 8 has a similar impact as the first one. Specifically, it restricts adding new relations to the counted elements.

Next, we prove the correctness of Algorithm 2. The proof is split into two parts. Lemma 3 shows the correctness of the procedure shown in Algorithm 1. Lemma 4 shows the correctness of the transformation of counting multiple columns to counting a single column. Note that both proofs consider that the input ontology $\mathcal{O}$ fulfills the restrictions in Definition 8.

Lemma 3. *Algorithm 1 is a correct procedure of* $\mathsf{CD}^{\mathcal{C}}(Q, \mathcal{O})$ *considering the input ontology $\mathcal{O}$ fulfills the restrictions in Definition 8.*

Proof. Let $\langle a, b \rangle = \mathsf{CD}^{\mathcal{C}}(Q, \mathcal{O})$ and let $\langle a', b' \rangle = CDSingleColumn(, \mathcal{O})$ where $T = Q_{\mathcal{O}}$. We show that $a = a'$ and $b = b'$. Here we will show the proof for $a = a'$. The proof for $b = b'$ is similar. First, we prove that $a \leq a'$. The set axioms in $\mathcal{O}$ is a subset of the axioms in $\mathcal{O}'_i$ ontology. By definition, there exists a model $\mathcal{I}$ of $\mathcal{O}'_{a'}$, $\mathcal{I} \models \mathcal{O}'_{a'}$. Because of the monotonicity property of $\mathcal{SROIQ}(\mathcal{D})$ ontologies, $\mathcal{I}$ is also a model of $\mathcal{O}$. The number $n = \sharp((Q_{\mathcal{O}})^{\mathcal{I}})$ is greater or equal to a and it is equal to the number a'. Therefore $a \leq a'$.

Now we prove that $a \geq a'$. This is equivalent to the statement that for any $\mathcal{I}$ model of $\mathcal{O}$, $i = \sharp((Q_{\mathcal{O}})^{\mathcal{I}})$, there exists a model $\mathcal{I}'$ of $\mathcal{O}'_i$, where $\sharp(T^{\mathcal{I}}) = \sharp(T^{\mathcal{I}'})$. We construct $\mathcal{I}'$ from $\mathcal{I}$ as follows: $z^{\mathcal{I}'} := z^{\mathcal{I}}, z \in S$, $x^{\mathcal{I}'} := a$, where a is an arbitrary element from $\Delta^{\mathcal{I}}$; $p^{\mathcal{I}} = \{(a, t_1), (a, t_2), \cdots, (a, t_n)\}$. It is easy to show that $\mathcal{I}'$ is a model of $\mathcal{O}$ as it has the same domain and mappings as $\mathcal{I}$. The following implies by the construction of $\mathcal{I}'$: $\sharp(T^{\mathcal{I}}) = \sharp(T^{\mathcal{I}'})$; $\mathcal{I}' \models \{p(x,t) \mid t \in T\}$. Also $\mathcal{I}' \models (\leq i\, p)(x)$ which is implied by the interpretation conditions of the class expression $\leq i\, p$, see Definition 1, and the equality $\sharp\{y \mid (x,y) \in p^{\mathcal{I}} \text{ and } y \in \Delta^{\mathcal{I}}\} = \sharp\{t_i^{\mathcal{I}} \mid (a, t_i^{\mathcal{I}}) \in p^{\mathcal{I}} \text{ and } t_i \in T\} = \sharp(T^{\mathcal{I}'}) = \sharp(T^{\mathcal{I}}) = i$. Therefore $\mathcal{I}' \models \mathcal{O}'_i$ and $\sharp T^{\mathcal{I}} = T^{\mathcal{I}'}$. □

Lemma 4. *The transformation from multi to single column counting in Algorithm 2 is a practically correct procedure for computing* $\mathsf{CD}^{\mathcal{C}}(Q, \mathcal{O})$.

Proof. Counting the set of interpreted row individuals is equivalent to counting the set of interpreted result bindings if there is a one-to-one correspondence between the interpreted rows and interpreted result bindings for each model of the ontology. Let $\mathcal{I}$ be a model of the ontology $\mathcal{O}'$.

To prove the correspondence, we first show that there is a unique interpreted result binding $t_i^{\mathcal{I}}, t_i \in T$ for each interpreted row $r_i \in T'$. Because of the functional restrictions on the column properties $c_1, c_2, \cdots, c_k$, the interpreted row $r_i^{\mathcal{I}}$ is restricted to have at most one value per column property. Therefore, there is exactly one interpreted unique tuple $t_i^{\mathcal{I}}$ associated with $r_i^{\mathcal{I}}$.

Second, we prove that each interpreted tuple $t_i^{\mathcal{I}}$ is associated with exaclty one interpreted row $r_i^{\mathcal{I}}$. In this case, the HasKey axiom restricts each unique $t_i^{\mathcal{I}}$ to be associated with exactly one interpreted row $r_i^{\mathcal{I}}$. This proves the one-to-one correspondence.

Because there is a one-to-one correspondence between the interpreted rows and interpreted result bindings for each model of the ontology, the proposed transformation from multi to single column counting is correct. □

Theorem 2. *The* $\mathsf{CDC}(\mathsf{Q}, \mathcal{O})$ *Algorithm 2 is a correct procedure of* $\mathsf{CD}^{\mathcal{C}}(Q, \mathcal{O})$.

Proof. The proof of this theorem is a consequence of Lemmas 3 and 4. □

6 Experiments

In this section, we report the results of our experiments with Algorithm 1 which is the core procedure of the implementation of the CEC semantics. In the experiment, we are counting a set of individuals T described in an OWL 2 ontology. The goal of the experiments is to form an initial assessment of the practicality of the algorithm.

In our first tests we experimented with small sizes of T, e.g., 1 to 10. We found that, as expected the bottleneck of the procedure is the consistency check operation used on lines 8 and 15 in Algorithm 1. Note that the complexity of the algorithm depends on the complexity of the consistency check procedure. In the case of $\mathcal{SROIQ}(\mathcal{D})$ reasoner the complexity is known to be NEXPTIME.

We tried to optimize this operation. In general, using an arbitrary reasoner for consistency checking has the limitation that in each iteration the consistency checking must start from scratch. Note that both loops on lines 6 and 13 in Algorithm 1 iterate over the restriction of the count from the general to the more specific. Theoretically, because of the monotonicity property of $\mathcal{SROIQ}(\mathcal{D})$, using incremental reasoner may improve performance.

In practice, the improvement gained from switching to an incremental reasoner was negligible compared to another factor. The last consistency check in the iteration process (the one that should fail) may take a very long time even for a small number of elements, e.g., 10 is that. This is because the reasoner must check all possible combinations of partitionings of the counted elements. The number of possible partitionings with k non-empty partitions of an n element set is given by the Stirling number $s(n,k)$. The following examples demonstrate the problem evaluating an inconsistent $\mathcal{O}'_i$ ontology, e.g. $s(10,8) = 750$, $s(20,16) = 22350954$.

This observation led us to another optimization idea, where we attempt to recognize the last consistency check based on how long it takes to terminate. We implemented a consistency check timeout strategy in which we terminate the cycles at lines 6 and 13 if the consistency check takes longer than the total time spent on that operation during the execution. Note that this strategy has an impact on correctness. Because the last consistency check may be falsely identified, the results might be incomplete. Soundness is preserved in the sense that the calculated interval will be contained in $\mathsf{CD}^{\mathcal{C}}(Q, \mathcal{O})$. For example, if the correct result of a count distinct query is $\mathsf{CD}^{\mathcal{C}}(Q, \mathcal{O}) = \langle 100, 150 \rangle$ and the timeout strategy fails it will always return an interval contained within the correct one, e.g., $\mathsf{CDCSingleColumn}(T, O) = \langle 125, 140 \rangle$.

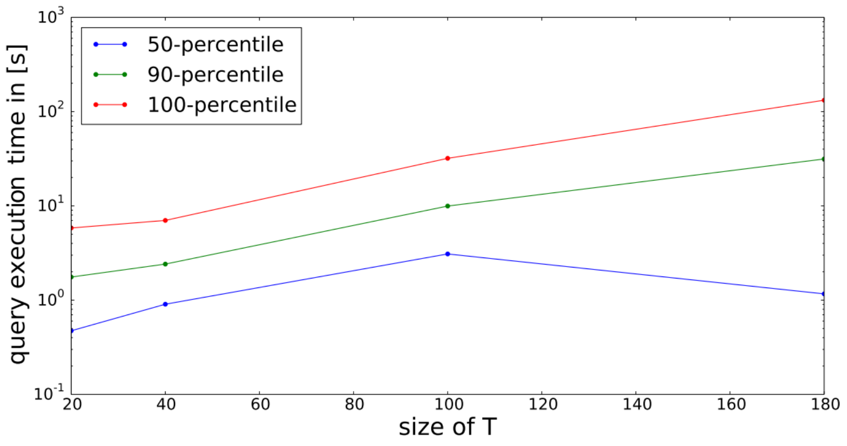


Figure 1: Execution time dependence on size of T

Figure 1 shows the results of Algorithm 1 using an incremental reasoner and the consistency timeout strategy described above. We evaluated 4 queries (i.e., 4 different sizes of $T \in \{20, 40, 100, 180\}$) over randomly generated artificial datasets. Each query was executed 100 times. Figure 1 shows the median, the 90- and 100-percentiles of the times that was needed to compute the query in each of the executions.

We observed the expected increase of execution time with the size of T. We also note that most of the executions were under a minute long. There were some cases for which the execution took nearly two and a half minute. The timeout strategy improved performance dramatically. Nevertheless, more experiments are needed to improve the performance of the algorithm and to check how the correctness of the results was compromised.

The queries were executed against randomly generated artificial datasets. The structure of each dataset D_i consists of individuals whose identity is complete in D_i. Each dataset partitions the set of individuals to n disjoint classes from the global schema. The merged dataset contains incomplete identity of all the individuals from all the datasets. The datasets were composed of up to 200 individuals and a random number (from 4 up to 150) of pairwise disjoint classes. The synthetic data over which we conduct our experiments try to capture the motivational example from Section 2. The common ontology is simulated by the disjointWith axiom between classes used to annotate the individuals. The level of matching between datasets of event reports D_1 and D_2 is captured by differentFrom axioms between the individuals.

The experiments were conducted with the Pellet reasoner [23] on a Dell laptop with Intel Core i7 processor and 8 GB memory.

7 State of Art

In recent years, expressive query languages, like SPARQL-DL [24], OWL-SAIQL [13], SQWRL [18] for OWL 2 [25] or SeRQL [2], SPARQL [20] for RDF, have been introduced and implemented in the field of semantic web. There have been deep studies [7, 22, 14, 5, 10] evaluating conjunctive queries in RDF and OWL, but few efforts have been spent on an algebra, as well as aggregation functions.

Recently the RDF query language SPARQL [20] has been extended towards the new SPARQL 1.1[2] [6] including many new constructs, e.g. aggregation functions or negation as failure. There are already publicly available implementations, e.g. ARQ [8], KGRAM [4], RDF::Query [26] or RDF4J [15] (previously known as Sesame [9]). Independently on SPARQL 1.1 the authors of [21] discuss the topic of aggregation over data structured as RDF graphs rather than on the relational data returned by the query (the result set table). The authors stress the need of different modes of aggregation. A few years ago the SQWRL query language for OWL [18] was proposed. All these efforts implement or interpret aggregation using the basic relational semantics, i.e. the result of aggregation functions is computed over the results of the non-aggregate variant of the query and neglect the impact of the interplay between aggregation functions and the inferred domain elements.

More closely related to our study is the work [3]. It shows that *exact semantics*, of aggregate queries over a $DL\text{-}Lite_A$ ontology returns results only if the Abox is not empty and if axioms in the Tbox resolve as constraints over cardinalities of the *groups* in the aggregate query. The authors propose *epistemic semantics* of aggregate

[2] SPARQL 1.1 is a W3C recommendation since March 2013.

queries in the settings of data integration use-cases returning the aggregate function's results known from (inferred by) the ontology. The authors propose an evaluation algorithm for a sub set of aggregate queries, i.e. restricted epistemic aggregate queries, defined using functional dependency of query variables w.r.t. the Tbox of the queried ontology. Compared to their work we investigate different epistemic semantics of count distinct queries in case the Unique Name Assumption does not hold, which is a typical case for the linked data integration scenario.

Although not exactly in the same framework of query answering like in this paper, the authors in [16] propose an approach to model/represent quantification over types without actually using explicit numerical information using an ontology design pattern. In comparison, we focus on answering count distinct queries over datasets containing incomplete knowledge.

In [12], a novel interpretation of *count distinct queries*, called *Semantic Tuple Count* (STC), was introduced and it was shown that the evaluation of the STC interpretation of count distinct queries is decidable in $\mathcal{SROIQ}(\mathcal{D})$. Also, STC is compared to three other known interpretations – *Basic Count* (conventional interpretation based on UNA), *Semantic Count* (a form of *Certain Answers* for count distinct queries defined on an interval of possible counts), and *Epistemic Count* (defined as the minimum of the *Semantic Count* interpretation)[3]. The term STC is equivalent to our term *Certain Epistemic Count* (CEC).

8 Conclusion and Future Work

This work presened in the paper extends our previous work [12] with a new semantics for count distinct queries – possible epistemic count (PEC) and a proof of its decidability. In this paper, we also propose an implementation of the certain epistemic count (CEC) semantics of count distinct queries over OWL 2 DL ontologies. We design and implement a reasoner-backed meta-algorithm and proved its correctness assuming some practical constraints on the queries ontology.

We also conducted some experiments over artificial, randomly generated data. During the experiments we verified that evaluating the first inconsistency using Algorithm 1 is not tractable. This was expected but poses challenges for future research. We report performance improvements when applying two optimization techniques, i.e., using an incremental reasoner and a time-out strategy, see Section 6.

We plan to extend experiments so that we can gain a better understanding of its performance characteristics. We plan to design and investigate possible optimizations.

The results of the work on count distinct queries can be used for the validation of cardinality constraints. We plan to spend time for investigating possibilities of their validation.

[3] Note, that comparing to [12] our definition of *Epistemic Count* is different.

Acknowledgements

This work was supported by grant No. GA 16-09713S Efficient Exploration of Linked Data Cloud of the Grant Agency of the Czech Republic and by grant No.SGS16/229/OHK3/3T/13 Supporting ontological data quality in information systems of the Czech Technical University in Prague.

References

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, *The description logic handbook: theory, implementation, and applications*, F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds. Cambridge University Press, 2003.

[2] J. Broekstra and A. Kampman, "An RDF query and transformation language," in *Semantic Web and Peer-to-Peer*, S. Staab and H. Stuckenschmidt, Eds., 2006, pp. 23–39.

[3] D. Calvanese, E. Kharlamov, W. Nutt, and C. Thorne, "Aggregate queries over ontologies," in *ONISW*, 2008, pp. 97–104.

[4] O. Corby, "Kgram: a knowledge graph abstract machine," Mar. 2012, http://wimmics.inria.fr/corese.

[5] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler, "Conjunctive query answering in the description logic $\mathcal{SHIQ}$," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007.

[6] S. Harris and A. Seaborne, "SPARQL 1.1 query language," W3C, W3C Recommendation, Mar. 2013, http://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[7] I. Horrocks, O. Kutz, and U. Sattler, "The even more irresistible sroiq," in *KR*, P. Doherty, J. Mylopoulos, and C. A. Welty, Eds. AAAI Press, 2006, pp. 57–67.

[8] A. Jena, "ARQ - A SPARQL Processor for Jena," Apr. 2011, http://jena.apache.org/documentation/query.

[9] A. Kampman, C. Fluit, and J. Broekstra, “Sesame,” Jan. 2013, http://sourceforge.net/projects/sesame.

[10] I. Kollia, B. Glimm, and I. Horrocks, “Query answering over SROIQ knowledge bases with SPARQL,” in *Proceedings of the International Workshop on Description Logic*, 2011.

[11] B. Kostov and P. Kremen, “Count aggregation in semantic queries - technical report,” Czech Technical University in Prague, Dept. of Cybernetics, Tech. Rep., 2013.

[12] B. Kostov and P. Křemen, “Count aggregation in semantic queries,” *International Workshop on Scalable Semantic Web Knowledge Base Systems co-located the International Semantic Web Conference (SSWS@ISWC)*, 2013.

[13] E. Kubias, S. Schenk, S. Staab, and J. Z. Pan, “OWL SAIQL - an OWL DL query language for ontology extraction,” in *W3C web ontology language (OWL) - experiences and directions workshop*, 2007.

[14] P. Křemen and Z. Kouba, “Conjunctive query optimization in OWL2-DL,” in *Proceedings of the 22th International Conference on Database and Expert System Applications*, ser. LNCS, vol. 6861, 2011.

[15] J. Leigh, “Rdf4j 2.2.2 released,” June 2017, http://rdf4j.org.

[16] D. C. Martínez, K. Janowicz, and P. Hitzler, “A logical geo-ontology design pattern for quantifying over types,” in *SIGSPATIAL/GIS*, 2012, pp. 239–248.

[17] B. Motik, B. C. Grau, and P. Patel-Schneider, “OWL 2 web ontology language direct semantics (second edition),” W3C, W3C Recommendation, Dec. 2012, http://www.w3.org/TR/2012/REC-owl2-direct-semantics-20121211/.

[18] M. J. O’Connor and A. K. Das, “SQWRL: A query language for OWL,” in *OWLED*, 2009.

[19] B. Parsia, P. Patel-Schneider, and B. Motik, “OWL 2 web ontology language structural specification and functional-style syntax (second edition),” W3C, W3C Recommendation, Dec. 2012, http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.

[20] E. Prud’hommeaux and A. Seaborne, “SPARQL query language for RDF,” W3C, W3C Recommendation, Jan. 2008, http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115, cit. 9.2017.

[21] D. Y. Seid and S. Mehrotra, “Grouping and aggregate queries over semantic web databases,” in *ICSC*, 2007, pp. 775–782.

[22] E. Sirin and B. Parsia, “Optimizations for Answering Conjunctive ABox Queries,” in *Description Logics*, ser. CEUR, vol. 189, 2006.

[23] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, “Pellet: a practical OWL-DL reasoner,” *J. Web Sem.*, vol. 5, no. 2, pp. 51–53, 2007.

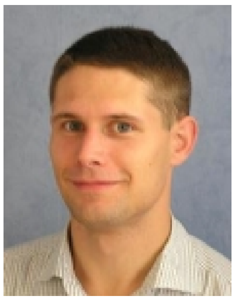
[24] E. Sirin and B. Parsia, “SPARQL-DL: SPARQL query for OWL-DL,” in *3rd OWL Experiences and Directions Workshop (OWLED-2007)*, 2007.

[25] W3C, “OWL 2 web ontology language document overview (second edition),” W3C, W3C Recommendation, Dec. 2012, http://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

[26] G. T. Williams, “RDF::Query - a complete SPARQL 1.1 query and update implementation for use with RDF::Trine,” Nov. 2012, http://search.cpan.org/dist/RDF-Query/.

AUTHOR BIOGRAPHIES

Bogdan Kostov is a Ph.D. student in the field of artifical intelligence and biocybernetics at the Czech Technical University in Prague, Czech Republic. His research topics are ontology development, semantic query answering that he applied in the domains of cultural heritage and aviation safety.



Dr. Petr Křemen received his Ph.D. degree in artifical intelligence and biocybernetics from the Czech Technical University in Prague, Czech Republic. He leads a research team at the Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University, Prague in the field of ontology-based information systems, ontology development, ontology comparison, error explanation and query answering. He is an author of more than 50 peer-reviewed articles, mainly on international fora.