Estimating the Cardinality of Conjunctive Queries over RDF Data Using Graph Summarisation

Giorgio Stefanoni, Boris Motik, and Egor V. Kostylev
Department of Computer Science, University of Oxford
Oxford, UK

ABSTRACT

Accurately estimating the cardinality (i.e., the number of answers) of conjunctive queries is a long-standing problem in database research. It is usually addressed by estimating selection queries using various kinds of database synopses (e.g., histograms), and combining these into estimates of joins using ad hoc assumptions about the data. Such assumptions, however, are the main source of estimation errors, which are known to compound exponentially with the number of joins. This is particularly problematic in graph-like data models such as RDF, where queries are navigational and typically involve many joins and self-joins. To address this, we present a new technique for estimating cardinality of conjunctive queries in RDF. The technique uses a new synopsis based on graph summarisation interpreted using a possible world semantics, and it formalises the estimation problem as computing the expectation of query cardinality over all databases compatible with the summary. We present ways to correctly compute such expectation even for queries with self-joins, as well as two algorithms for summarising RDF graphs. Finally, we show empirically that our technique produces estimates that are more accurate, often by orders of magnitude, as well as more consistent than those produced by well-known RDF and RDMBS systems.

1. INTRODUCTION

The Resource Description Framework (RDF) is a popular graph-like data format supported by products of many database vendors such as Oracle, MarkLogic, Systap, OntoText, Virtuoso, and Franz Inc. RDF is typically used in applications such as information integration and data exchange on the Web, where developing a rigid schema is either infeasible or undesirable. Objects in RDF are called *resources* and are identified using URIs, and an RDF dataset is a set of triples of the form $\langle s, p, o \rangle$, which say that a *subject* s is connected by a *predicate* p to an *object* o; moreover, the built-in *rdf*:*type* resource associates resources with their classes. For example, $\langle Peter, owns, car \rangle$ and $\langle car, rdf{:}type, Roadster \rangle$ say that 'Peter owns a car, which is a roadster'. By viewing triples as directed labelled edges, an RDF dataset can be seen as a directed graph. SPARQL [16] is the standard language for querying RDF graphs, and, just like in SQL, conjunctive queries are the basic kind of SPARQL queries.

Estimating the cardinality (i.e., the number of answers) of a conjunctive query is a long-standing problem in databases with numerous applications. Query optimisers use cardinality estimates to compute the cost of query plans, and the accuracy of such estimates critically affects an optimiser's ability to identify efficient execution plans [22]. Cardinality estimates can also be used to display an approximate number of answers without evaluating a potentially costly query. Thus, cardinality estimators are key components of virtually all (relational or RDF) database systems known to us.

Systems solve this problem by summarising the database using *synopses*, which are paired with techniques for estimating specific types of queries. One-dimensional synopses, such as one-dimensional histograms [18, 28] and wavelets [24], can summarise one attribute of one relation. Multidimensional synopses, such as multidimensional histograms [2, 7, 15, 29], multidimensional wavelets [9, 11], discrete cosine transforms [20], and kernel methods [14], can summarise several attributes of one relation. Finally, schema-level synopses, such as graphical models [12, 35], sampling [3, 36], TuG synopses [31], and statistical views [6, 10, 33] can summarise attributes of different relations. Queries that cannot be estimated directly using the available synopses are broken into parts that can be estimated, and partial estimates are combined using ad hoc assumptions as we discuss shortly.

Such assumptions, however, often do not hold in practice, which is a major source of estimation errors. Moreover, due to the graph-like nature of RDF, queries in RDF often navigate over paths and thus tend to contain many joins and self-joins (10 joins or more are not rare). Estimation error is known to compound exponentially with every additional join [19], which makes applying the existing cardinality estimation techniques to RDF challenging. Several proposals aim to address these specifics of RDF data [17, 26, 27, 32], with varying degrees of success. We next present an example demonstrating why estimating the cardinality of conjunctive (not necessarily just RDF) queries remains challenging. Then, in the rest of this paper, we go on to present a novel cardinality estimation approach that can effectively handle conjunctive RDF queries with many (self-)joins.

1.1 Limitations of the Existing Approaches

Consider the example RDF graph in Figure 1 where owners are connected to their cars. For clarity, triples of the form

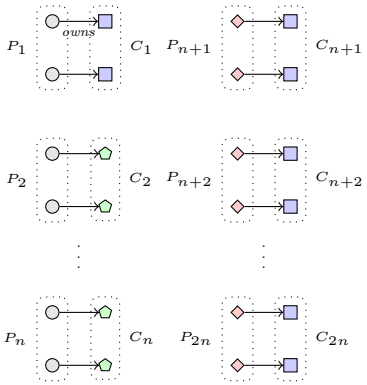

Figure 1: Example RDF graph

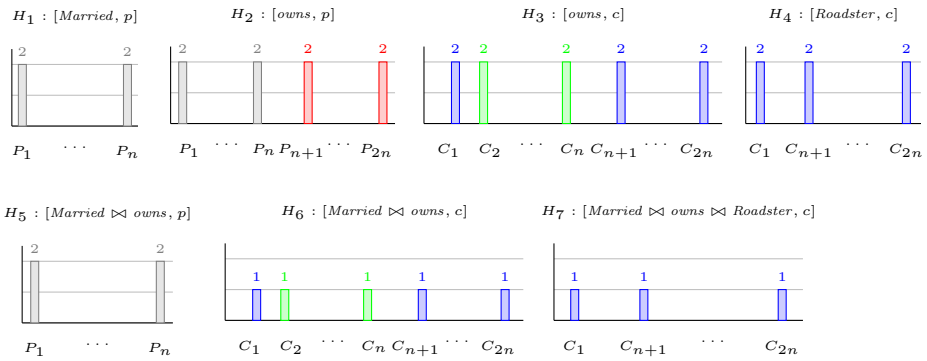

Figure 2: One-dimensional histograms for the example graph

$\langle d, rdf{:}type, o\rangle$ stating that resource d is of class o are encoded using colours and symbols: owners are married (grey circles) or single (red diamonds), and cars are roadsters (blue boxes) or vans (green pentagons). The meaning of P_i, C_i, and the dotted boxes is explained shortly. We wish to estimate the cardinality of the following SPARQL query, which retrieves all married people (?x) owning a roadster (?y).

```
SELECT ?x ?y WHERE { ?x rdf:type Married . ?x :owns ?y .
                     ?y rdf:type Roadster }
```

This query returns only two answers as all single people own roadsters, and most married people own vans. Hence, an accurate cardinality estimate must use correlations that become apparent only after joining the *owns* and *rdf:type* triples. In order to show how RDBMS techniques solve this problem, we convert our graph into a relational instance using vertical partitioning [1]: relation $owns(p, c)$ contains $\langle p, c\rangle$ for each triple $\langle p, owns, c\rangle$, relation $Married(p)$ contains $\langle p\rangle$ for each triple $\langle p, rdf{:}type, Married\rangle$, and relations $Single(p)$, $Roadster(c)$, and $Van(c)$ are constructed analogously. We transform analogously our query into the relational algebra expression $Married \bowtie owns \bowtie Roadster$.

Histograms are the most common synopses in RDBMSs [18, 28]. A one-dimensional histogram $H : [rel, att]$ for attribute *att* of relation *rel* is typically constructed by sorting the values of *att* in *rel* and then partitioning the values into buckets according to a *histogram rule*. A bucket b does not store the values themselves, but just records the minimum and the maximum value, the number f_b of total values (i.e., the bucket frequency), and the number d_b of distinct values. Figure 2 shows histograms H_1 to H_4 obtained by grouping the resources in the graph in Figure 1 using dotted boxes. For simplicity, we group the values of different relations in the same way, and so we refer to buckets in different histograms using unique names P_i and C_i; for example, we say that histograms H_1 and H_2 both contain a bucket P_1. Figure 2 shows just the frequency of each bucket. To estimate the cardinality of our query, one would first join H_1 and H_2 into histogram H_5: for each bucket b_1 of H_1 and each bucket b_2 of H_2 with the same value range, histogram H_5 contains a bucket b with frequency $f_b = (f_{b_1} \cdot f_{b_2})/\max(d_{b_1}, d_{b_2})$ [5] and $d_b = \min(d_{b_1}, d_{b_2})$ distinct values (in general, if the ranges of b_1 and b_2 overlap but do not match exactly, one first *splits* buckets into equal ranges). The formula for f_b is obtained by several ad hoc assumptions about the data: all values of one attribute are contained in another (*containment assumption*) and are uniformly distributed (*uniformity assumption*) [5]. The next step is problematic: H_5 summarises the attribute p of $Married \bowtie owns$, but we need a histogram for c. A common ad hoc solution is to assume that the distributions of c in $Married \bowtie owns$ and in *owns* are the same (*preservation assumption*) [5]. The latter is given by H_3, but the sum of all bucket frequencies in H_3 is $4n$; in contrast, the sum of the bucket frequencies in H_5 is $2n$ so, to preserve the cardinality of $Married \bowtie owns$, one scales H_3 by $2n/4n$ to obtain H_6. One then joins H_6 with H_4 into H_7, and one sums the frequencies of all buckets in H_7 to estimate the query cardinality as $n + 1$. Note that the estimation error depends on n and is thus unbounded.

The need to capture correlations is well understood, and multidimensional histograms aim to summarise several attributes of a single relation [2, 7, 15, 29]. Thus, if a single relation contained all information about ownership and the classes of owners and cars, such a histogram could capture the fact that most married people own vans. In our example, however, this correlation becomes apparent only via joins, so multidimensional histograms are not helpful. Thus, to correctly estimate the cardinality of our query, we should describe $Married \bowtie owns$ by a two-dimensional histogram reflecting the correlations of p and c values after the first join, so that we can take this into account in the final join with *Roadster*. However, to the best of our knowledge, existing work on multidimensional histograms focuses mainly on estimating selections over a single relation, rather than propagating correlations over joins. A notable exception is the work on multidimensional wavelets [9, 11], which can be understood as multidimensional synopses that group values based on similarity (e.g., integers whose difference is small). Proposed cardinality estimation formulae propagate correlations through joins, but without a clear semantic interpretation of the computed results. Moreover, these techniques seem to be applicable primarily to numeric data, unlike RDF data that consists mainly of opaque identifiers.

Relational histograms have been adapted to the RDF setting [32], and they have been refined to allow accurately estimate binary joins and frequent path queries [27], and star queries [26]. However, outside these query classes, they combine subquery results using ad hoc assumptions and thus suffer from the same problems we outlined earlier.

Schema-level synopses capture data correlations across predetermined join paths, which in some approaches cannot

include self-joins. Graphical models summarise the universal relation obtained from all possible joins [12, 17, 35]; TuG synopses represent correlations in the corresponding data graph [31]; a query can be rewritten over views with precise statistics [6, 10, 33]; and an unbiased estimator of query cardinality can be obtained by sampling the database [3, 36]. These techniques provide excellent results on queries covered by the given join paths, but otherwise they fall back on ad hoc assumptions for combining estimates to subqueries. RDF data is often very heterogeneous so specifying the relevant join paths in advance is difficult. Similarly, it is often difficult to anticipate in advance the maximum length of paths in queries, which is needed to specify the relevant self-joins (in approaches that support self-joins at all).

1.2 Our Contributions

To address these issues, in Section 3 we propose a new, principled technique for estimating the cardinality of conjunctive queries over RDF data. Our technique uses graph summaries as schema-level synopses, where a summary is obtained by merging similar resources of an RDF graph. Following LeFevre and Terzi [21], we interpret summaries using a *possible world semantics* as a family of RDF graphs represented by the summary. We formalise the cardinality estimation problem as computing the expectation of the query cardinality across this family of possible graphs.

In Section 4 we present a closed-form formula for computing the expected cardinality, and we prove the formula correct for all conjunctive queries, even those with self-joins. We also show how to compute the cardinality variance and use it obtain a confidence interval for the estimate.

As in all schema-level approaches known to us, evaluating the estimation formula can be exponential in query size. The performance of cardinality estimation is critical in applications such as query planning, so in Section 5 we adapt the well-known results on queries of bounded hypertree-width [13] and present an algorithm that computes the estimate in time exponential only in a certain structural parameter of the query. This can intuitively be understood as using a bushy 'join plan' that reduces complexity by projecting out attributes as soon as they are not needed and aggregating the results after each projection. RDF queries are often tree-shaped or are similar to tree-shaped queries, so the mentioned parameter is typically low, which we exploit to considerably reduce cardinality estimation times.

In Section 6 we present two algorithms for RDF graph summarisation. The first one identifies similar resources using their types, and outgoing and incoming edges, while the second one also takes resource neighbourhood into account.

In Section 7 we present the results of an extensive evaluation on three well-known RDF datasets with 55M to 109M triples, and 15 to 103 queries. We compared the accuracy of our technique against RDF-3X [27] (whose cardinality estimator specifically targets RDF), PostgreSQL (which uses the traditional histogram-based approach), and SystemX (a prominent commercial RDBMS that combines histograms with dynamic sampling); we considered both triple table and vertical partitioning for storing RDF in RDBMSs. We focused on comparing the accuracy of different cardinality estimation approaches; investigating the impact on query plans is an orthogonal issue that is out of scope of this work. Our results show that our technique is generally more accurate than the related approaches, often by orders of magnitude.

The proofs of all of our technical results are given in full in the appendices.

2. DEFINITIONS AND NOTATION

We now recapitulate the relevant definitions and notation. RDF vocabulary consist of *resources*, which are either URIs (i.e., abstract entities) or *literals* (i.e., concrete values of datatypes such as *xsd:string* or *xsd:integer*). A *term* is a resource or a variable. An *(RDF) atom* is of the form $\langle s, p, o \rangle$, where s, p, and o are terms called the *subject*, *predicate*, and *object*, respectively. An *(RDF) triple* is a variable-free atom. An *(RDF) graph* is a finite set of triples.

A *substitution* π is a mapping of variables to terms; the domain and range of π are $\mathsf{dom}(\pi)$ and $\mathsf{rng}(\pi)$, respectively; $\mathsf{varrng}(\pi)$ is the set of all variables in $\mathsf{rng}(\pi)$; and π is *empty* if $\mathsf{dom}(\pi) = \emptyset$. Let Z be a term, an atom, or a set of atoms. Then, $\pi(Z)$ is obtained from Z by replacing each occurrence of $x \in \mathsf{dom}(\pi)$ in Z with $\pi(x)$; if Z is a set of atoms, then $\pi(Z)$ is also a set and does not contain duplicates. Moreover, $\mathsf{res}(Z)$, $\mathsf{var}(Z)$, and $\mathsf{term}(Z)$ are the sets of resources, variables, and terms occurring in Z, respectively.

SPARQL [16] is a standard query language for RDF. Its syntax is quite verbose, so we next introduce a simpler, more compact notation. SPARQL variables are written as $?x$, but we drop the question mark and reserve (possibly indexed) letters x, y, z for variables. A *conjunctive query* (*CQ*) q is a finite set of atoms. A substitution π is an *answer* to q on an RDF graph G if $\mathsf{dom}(\pi) = \mathsf{var}(q)$ and $\pi(q) \subseteq G$; moreover, $[\![q]\!]_G$ is set of all answers to q on G, and the *cardinality* of q on G is the size of $[\![q]\!]_G$. For technical reasons, q can be empty, so then $[\![q]\!]_G$ contains just one empty answer. Our definition of conjunctive queries thus covers SPARQL queries of the form `SELECT * WHERE { BGP }` where `BGP` is a *basic graph pattern*, which form the foundation of SPARQL. Please note that, without `DISTINCT`, projecting variables in the `SELECT` clause does not affect the query cardinality, and we leave developing support for `DISTINCT` for our future work.

For integers $0 < m \leq n$, the *binomial coefficient* is $\binom{n}{m}$; the *falling factorial* is $(n)_m = n(n-1)\dots(n-m+1)$; and $[m, n]$ is the set of all integers between m and n (inclusive).

3. SUMMARISATION FRAMEWORK

Summarisation can reduce the size of a graph by orders of magnitude while still supporting tasks such as graph exploration [25, 34], fast approximate graph analytics [21, 30], and query processing [8]. We show that summaries can also be used to estimate CQ cardinality in a principled way.

Consider the graph G in Figure 3 where company employees are connected to their cars and managers; resource classes are encoded as in the legend. We create a summary S from G by merging the resources of G into *buckets* as shown by dotted boxes; for example, we replace employees e_1 and e_2 with bucket b_1, and e_3 and e_4 with another bucket b_3. Moreover, we label each edge in the summary with the number of edges of G that collapse due to merging; for example, the *manages* edge from e_1 to e_2 collapses into a self-loop on b_1 of weight 1, and the edges from e_1 to e_3, and from e_2 to e_4 collapse into a single edge from b_1 to b_3 of weight 2.

Summary S thus contains less information than the graph G it was obtained from. Our objective is to estimate the cardinality of a CQ q using S, rather than G; hence, we must 'reconstruct' the graph(s) that S represents. We use

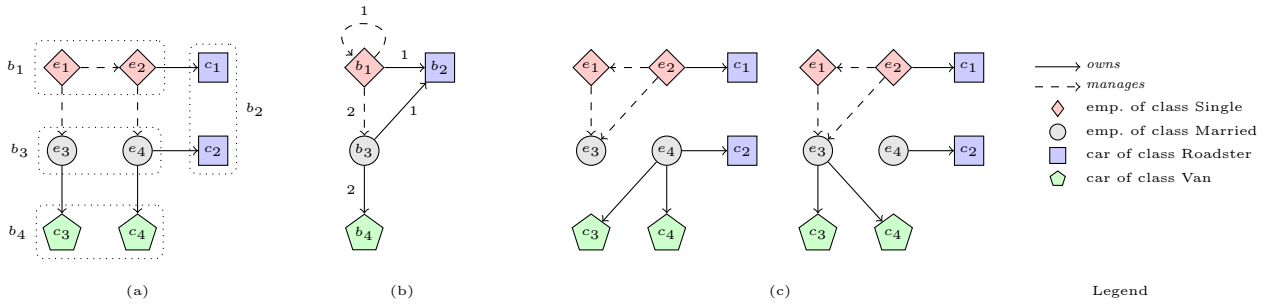

Figure 3: An example RDF graph G (a), its summary S (b), and example graphs G_1 and G_2 that S represents (c)

the 'possible worlds' semantics [21], where S represents with equal probability each graph G' that summarises as S—that is, if we replace the resources in G' in the same way as when we constructed S from G, we obtain S. In Figure 3, in addition to G, summary S also represents graphs G_1 and G_2. We then estimate the cardinality of q on G as the expected cardinality of q over all graphs represented by S.

We next define our notion of a summary. This notion is independent from any graph the summary is obtained from since its main purpose is to represent a family of graphs.

Definition 3.1. *A* summary *S is a tuple $\langle H, w, \mu\rangle$ with the following components:*

- *H is an RDF graph called the* summarisation graph*, and the resources in $\mathsf{res}(H)$ are called* buckets*;*
- *$w : H \to \mathbb{N}$ is a* weight function *that assigns a positive natural number to each triple in H; and*
- *μ is a surjective* summarisation function *from a finite set of resources $\mathsf{dom}(\mu)$ to $\mathsf{res}(H)$.*

For Z a term, an atom, or a set of atoms, $\mu(Z)$ is obtained from Z by replacing each resource $d \in \mathsf{res}(Z) \cap \mathsf{dom}(\mu)$ with $\mu(d)$; when Z is a set, $\mu(Z)$ is a set as well.

*The S-*size *of a bucket $b \in \mathsf{res}(H)$ and the S-*size *of a triple $\langle b_1, b_2, b_3\rangle \in \mathsf{res}(H) \times \mathsf{res}(H) \times \mathsf{res}(H)$ are defined as*

$$s[b] = |\{d \in \mathsf{dom}(\mu) \mid \mu(d) = b\}| \quad \text{and}$$
$$s[\langle b_1, b_2, b_3\rangle] = s[b_1] \times s[b_2] \times s[b_3].$$

To obtain accurate estimates, we should merge resources with similar connections, and a resource ordering is unlikely to provide us with a good similarity criterion. Thus, whereas buckets in histograms just record value ranges, buckets in our approach are associated with resources explicitly via μ. In our example, $\mu(e_1) = \mu(e_2) = b_1$, $\mu(c_1) = \mu(c_2) = b_2$, and so on, and we define μ as identity on all remaining resources, all of which occur in G as predicate or object of *rdf:type* (e.g., *owns*, *rdf:type*, *Car*, and so on). The domain and the range of μ thus need not be disjoint. The size of a bucket b is the number of resources mapped to b; in our example, $s[b_i] = 2$ for $i \in [1, 4]$, but $s[\mathit{rdf{:}type}] = s[\mathit{Car}] = 1$. While values can be grouped differently in histograms of distinct relations, we summarise resources across all of G. For $f \in G$ and $h \in H$ where $\mu(f) = h$, we say that f is *summarised* as h, and that h can be *expanded into* f. Clearly, h can be expanded into at most $s[h]$ triples. In our example, we have

$$s[\langle b_1, \mathit{manages}, b_1\rangle] = s[b_1] \times s[\mathit{manages}] \times s[b_1] \;\; = 4,$$
$$s[\langle b_1, \mathit{rdf{:}type}, \mathit{Car}\rangle] = s[b_1] \times s[\mathit{rdf{:}type}] \times s[\mathit{Car}] = 2.$$

We next formalise the 'possible worlds' interpretation.

Definition 3.2. *A summary $S = \langle H, w, \mu\rangle$* represents *an RDF graph G if*

- *$\mathsf{res}(G) \subseteq \mathsf{dom}(\mu)$,*
- *$H = \mu(G)$, and*
- *$w[h] = |\{f \in G \mid \mu(f) = h\}|$ holds for each $h \in H$.*

Set R_S contains each graph that S represents. Finally, summary S is consistent *if $R_S \neq \emptyset$.*

For each $h \in H$, at most $s[h]$ triples can be summarised into h, so $w[h] \leq s[h]$ must hold for S to be consistent. Conversely, we can obtain a graph in R_S by expanding each $h \in H$ arbitrarily into $w[h]$ triples if $w[h] \leq s[h]$ holds. Thus, we can check consistency of S as in Proposition 3.3.

Proposition 3.3. *A summary $S = \langle H, w, \mu\rangle$ is consistent if and only if $w[h] \leq s[h]$ holds for each $h \in H$.*

Based on our interpretation of R_S, it is now natural to estimate the cardinality of a CQ q as the expected cardinality of q over all graphs represented by S, and to use the variance of the cardinality of q as a measure of confidence.

Definition 3.4. *The* expectation *$E_{q,S}$ and the* variance *$\sigma^2_{q,S}$ of the cardinality of a CQ q on a summary S are*

$$E_{q,S} = \sum_{G \in R_S} \frac{|[\![q]\!]_G|}{|R_S|} \;\text{ and }\; \sigma^2_{q,S} = \sum_{G \in R_S} \frac{(|[\![q]\!]_G| - E_{q,S})^2}{|R_S|}.$$

We see several important benefits of our framework. First, although summarisation loses information, we can control what information is lost and what remains preserved. For example, summarising the graph in Figure 1 as shown in the figure retains the information that married people typically own vans; moreover, this information remains preserved if we further merge buckets $P_2, \ldots, P_n$ and $C_2, \ldots, C_n$. In addition, unlike the ad hoc assumptions mentioned in Section 1.1, our 'possible worlds' interpretation has a clear interpretation. As a consequence, for each graph $G \in R_S$ and each positive number k, Chebyshev's inequality states that $P(|[\![q]\!]_G - E_{q,S}| \geq k \cdot \sigma_{q,S}) \leq \frac{1}{k^2}$, which can provide us with statistical bounds on the estimation error: we can fix a desired confidence $\wp$ and then compute k such that the exact answer is within $E_{q,S} \pm k \cdot \sigma_{q,S}$ with probability $\wp$. We are unaware of other approaches that provide similar bounds.

4. COMPUTING THE EXPECTATION

Given a CQ q and a summary $S = \langle H, w, \mu\rangle$, we can compute $E_{q,S}$ by iterating over R_S, but this is impractical since $|R_S|$ is exponential in $|H|$. Thus, we next present formulas that compute $E_{q,S}$ in time polynomial in $|H|$.

4.1 Intuition

We discuss our intuitions using summary S from Figure 3. Let $q_1 = \{f_1, f_2\}$ contain triples $f_1 = \langle e_1, \mathit{manages}, e_3\rangle$ and $f_2 = \langle e_3, \mathit{owns}, c_3\rangle$. Query q_1 does not contain variables so, for each $G' \in R_S$, either $[\![q_1]\!]_{G'} = 1$ if $q_1 \subseteq G'$, or $[\![q_1]\!]_{G'} = 0$ otherwise; thus, the numerator of $E_{q_1,S}$ is the number of graphs in R_S containing q_1. Triples f_1 and f_2 are summarised as $h_1 = \langle b_1, \mathit{manages}, b_3\rangle$ and $h_2 = \langle b_3, \mathit{owns}, b_4\rangle$; for readability, let $w_i = w[h_i]$ and $s_i = s[h_i]$ for $i \in \{1, 2\}$. Now note that every expansion of h_1 can be seen as a choice of w_1 triples from s_1 possibilities, so there are a total of $\binom{s_1}{w_1}$ expansions of h_1. Moreover, a total of $\binom{s_1-1}{w_1-1}$ expansions of h_1 contain f_1: in addition to f_1, we choose the remaining $w_1 - 1$ triples from $s_1 - 1$ possibilities. Analogously, h_2 has $\binom{s_2}{w_2}$ expansions, of which $\binom{s_2-1}{w_2-1}$ contain f_2. Now, crucially, f_1 and f_2 are summarised as distinct triples, so the expansions of h_1 and h_2 are independent; hence, the total number of expansions of h_1 and h_2, and the number of expansions containing both f_1 and f_2, can be obtained by multiplying the above factors. Finally, the expansions of each $h \in H \setminus \{h_1, h_2\}$ are independent from and thus irrelevant to q_1, so they do not affect $E_{q_1,S}$. Hence, we get

$$E_{q_1,S} = \frac{\binom{s_1-1}{w_1-1}}{\binom{s_1}{w_1}} \cdot \frac{\binom{s_2-1}{w_2-1}}{\binom{s_2}{w_2}} = \frac{w_1}{s_1} \cdot \frac{w_2}{s_2} = \frac{2}{2 \cdot 2} \cdot \frac{2}{2 \cdot 2} = 0.25.$$

Let $q_2 = \{\langle x, \mathit{manages}, y\rangle, \langle y, \mathit{owns}, z\rangle\}$. For each answer $\pi \in [\![q_2]\!]_G$, query $\pi(q_2)$ does not contain variables; thus, we can compute $E_{\pi(q_2),S}$ as in the previous paragraph, and we can compute $E_{q_2,S}$ by summing the contribution of each π. Enumerating all π would be inefficient, but we note that, for each $\pi \in [\![q_2]\!]_G$, there exists $\tau \in [\![\mu(q_2)]\!]_H$ such that $\mu(\pi(x)) = \tau(x)$ for all $x \in \mathsf{dom}(\pi)$; we say that τ *expands into* π. Thus, for each τ, we can compute $E_{\pi(q_2),S}$ for just one prototypical substitution π that τ expands into, and then multiply $E_{\pi(q_2),S}$ by the number of expansions of τ. Evaluating $\mu(q_2)$ on H produces the following answers.

$$\frac{\tau_1\colon\ x\ \ y\ \ z}{b_1\ b_3\ b_4} \qquad \frac{\tau_2\colon\ x\ \ y\ \ z}{b_1\ b_1\ b_2} \qquad \frac{\tau_3\colon\ x\ \ y\ \ z}{b_1\ b_3\ b_2}$$

For τ_1, we can select, say, $\pi = \{x \mapsto e_1, y \mapsto e_3, z \mapsto c_3\}$ as a prototypical expansion of τ_1; then $\pi(q_2) = q_1$, and so we have $E_{\pi(q_2),S} = E_{q_1,S} = 0.25$. Moreover, since τ_1 maps both atoms of q_2 to distinct triples in H, each expansion π of τ_1 does the same; consequently, we can obtain all such π by expanding each variable in $\mathsf{dom}(\tau_1)$ independently, and so there are $s[\tau_1(x)] \cdot s[\tau_1(y)] \cdot s[\tau_1(z)] = 2 \cdot 2 \cdot 2 = 8$ expansions of τ_1. Thus, τ_1 contributes to $E_{q_2,S}$ by $8 \cdot 0.25 = 2$. We compute the contributions of τ_2 and τ_3 analogously as $2^3 \cdot \frac{1 \cdot 1}{4 \cdot 4} = 0.5$ and $2^3 \cdot \frac{2 \cdot 1}{4 \cdot 4} = 1$, respectively. Finally, by summing all contributions, we get $E_{q_2,S} = 2 + 0.5 + 1 = 3.5$.

We can generalise these principles to formula (1) that, given a CQ q, sums the contribution of each $\tau \in [\![\mu(q)]\!]_H$; the first factor counts the expansions of τ to a prototypical π, and the second one computes the contribution of π.

$$E_{q,S} = \sum_{\tau \in [\![\mu(q)]\!]_H} \prod_{x \in \mathsf{var}(q)} s[\tau(x)] \cdot \prod_{a \in q} \frac{w[\tau(\mu(a))]}{s[\tau(\mu(a))]} \qquad (1)$$

However, formula (1) is correct only if each answer to $\mu(q)$ in H maps all atoms of $\mu(q)$ to distinct triples in H. Consider $q_3 = \{\langle e_3, \mathit{owns}, x\rangle, \langle e_3, \mathit{owns}, y\rangle\}$ and $[\![\mu(q_3)]\!]_H$ as follows.

$$\frac{\tau_1\colon\ x\ \ y}{b_2\ b_4} \qquad \frac{\tau_2\colon\ x\ \ y}{b_4\ b_2} \qquad \frac{\tau_3\colon\ x\ \ y}{b_4\ b_4} \qquad \frac{\tau_4\colon\ x\ \ y}{b_2\ b_2}$$

Answers τ_1 and τ_2 map the atoms of $\mu(q_3)$ to distinct triples in H, so their contributions can be computed as in (1) as $2 \cdot 2 \cdot \frac{2}{2 \cdot 2} \cdot \frac{1}{2 \cdot 2} = 0.5$. However, τ_3 maps both atoms of $\mu(q_3)$ to the same triple $h_3 = \langle b_3, \mathit{owns}, b_4\rangle$, so τ_3 expands into two kinds of π. First, π can map both atoms of q_3 to the same triple (e.g., $\langle e_3, \mathit{owns}, c_4\rangle \in G_2$). Query $\pi(q_3)$ then contains just one atom, so $E_{\pi(q_3),S} = w[h_3]/s[h_3] = 2/4 = 0.5$; moreover, x and y are mapped to the same value in each such π, so the number of such π is $N_1 = s[\tau_3(x)] = 2$. Second, π can map the atoms of q_3 to distinct triples. Then, each expansion of h_3 containing $\pi(q_3)$ corresponds to a choice of $w[h_3] - 2$ triples out of $s[h_3] - 2$ possibilities, so we get

$$E_{\pi(q_3),S} = \frac{\binom{s[h_3]-2}{w[h_3]-2}}{\binom{s[h_3]}{w[h_3]}} = \frac{(w[h_3])_2}{(s[h_3])_2} = \frac{(2)_2}{(4)_2} = \frac{2 \cdot 1}{4 \cdot 3} = 0.167.$$

Moreover, there are $N_2 = s[\tau_3(x)] \cdot s[\tau_3(y)] = 2 \cdot 2 = 4$ expansions of τ_3, and $N_1 = 2$ of these map x and y to the same resource; thus, $N_2 - N_1 = 2$ expansions of τ_3 map x and y to distinct resources; and so τ_3 contributes to $E_{q_3,S}$ by $N_1 \cdot 0.5 + (N_2 - N_1) \cdot 0.167 = 1.33$. Finally, τ_4 maps both atoms of $\mu(q_3)$ to the same triple $h_4 = \langle b_3, \mathit{owns}, b_2\rangle$, but no expansion of τ_4 contains two triples since $w[h_4] = 1$; thus, we consider only $s[\tau_4(x)] = 2$ expansions π of τ_4, each mapping both atoms of q_3 to the same triple and thus contributing by $E_{\pi(q_3),S} = 1/2 = 0.5$; hence, τ_4 contributes to $E_{q_3,S}$ by $2 \cdot 0.5 = 1$. Thus, $E_{q_3,S} = 0.5 + 0.5 + 1.33 + 1 = 3.33$.

4.2 Formalisation

For the rest of this section, we fix a consistent summary $S = \langle H, w, \mu\rangle$ and a CQ q with $\mathsf{res}(q) \subseteq \mathsf{dom}(\mu)$. We also fix an arbitrary total order $\leq$ on terms such that $d \leq x$ holds for each resource d and each variable x.

The following notion of a partition of q allows us to count the different types of answers of q on graphs in R_S.

Definition 4.1. *A* partition *of q is a set P of mutually disjoint nonempty subsets u_i of q such that $q = \bigcup_{u_i \in P} u_i$.*

The term reachability graph *$\mathcal{G}_P$ for P contains all terms from $\mathsf{term}(q)$ as vertices, and undirected edges between terms s_1 and s_2, p_1 and p_2, and o_1 and o_2 for each $u \in P$ and all atoms $\langle s_1, p_1, o_1\rangle$ and $\langle s_2, p_2, o_2\rangle$ in u. Partition P is* unifiable *if no two distinct resources from $\mathsf{res}(q)$ are reachable in $\mathcal{G}_P$, in which case the substitution γ_P maps each variable $x \in \mathsf{var}(q)$ to the $\leq$-least term reachable from x in $\mathcal{G}_P$.*[1]

The partition base *B for q is the set of all unifiable partitions of q, and $\preceq$ is the partial order on B where, for all partitions $P, P' \in B$, the order satisfies $P \preceq P'$ if and only if, for each $u \in P$, there exists $u' \in P'$ with $u \subseteq u'$.*

For $P, P' \in B$ with $P \preceq P'$, a chain from P to P' of length *$\ell \geq 0$ is a sequence $P = P_0, \ldots, P_\ell = P'$ of partitions from B where $P_{i-1} \prec P_i$ holds for $i \in [1, \ell]$; moreover, $C^e(P, P')$ and $C^o(P, P')$ are the numbers of chains from P to P' of even and odd length; finally, $K(P, P') = C^e(P, P') - C^o(P, P')$.*

Note that, $C^e(P, P) = 1$ and $C^o(P, P) = 0$ for each $P \in B$, so $K(P, P) = 1$. We next discuss Definition 4.1 using query q_3 from Section 4.1; for convenience, let $a_1 = \langle e_3, \mathit{owns}, x\rangle$ and $a_2 = \langle e_3, \mathit{owns}, y\rangle$. The query has two unifiable partitions, $P_1 = \{\{a_1\}, \{a_2\}\}$ and $P_2 = \{\{a_1, a_2\}\}$. Partition P_1 represents the answers of q_3 on a graph in R_S where a_1 and a_2 are mapped to distinct triples; and P_2 represents

[1] Note that $\emptyset$ is the only unifying partition of $q = \emptyset$.

answers where a_1 and a_2 are mapped to the same triple, as reflected by $\gamma_{P_2} = \{y \mapsto x\}$ (assuming $x \leq y$). Moreover, $P_1 \prec P_2$ and $K(P_1, P_2) = -1$ show that each answer of q_3 satisfying P_2 is contained in an answer of q_3 satisfying P_1.

Definition 4.2 shows how to compute the contributions of each answer τ to $\mu(q)$ on H.

Definition 4.2. *Let τ be an answer to $\mu(q)$ on H. A partition $P \in B$ is* satisfied *by τ if, for each $x \in \mathsf{var}(q)$,*

- *$\gamma_P(x) \in \mathsf{var}(q)$ implies $\tau(x) = \tau(\gamma_P(x))$, and*
- *$\gamma_P(x) \in \mathsf{res}(q)$ implies $\tau(x) = \mu(\gamma_P(x))$.*

In such a case, B_τ is the set of all partitions in B that are satisfied by τ. For each $P \in B_\tau$, let

$$N_\tau(P) = \prod_{x \in \mathsf{varrng}(\gamma_P)} s[\tau(x)] \qquad \text{and}$$

$$F_\tau(P) = \begin{cases} \prod_{h \in \tau(\mu(q))} \frac{(w[h])_{\#_h}}{(s[h])_{\#_h}} & \text{if } \#_h \leq w[h] \ \forall h \in \tau(\mu(q)) \\ 0 & \text{otherwise} \end{cases}$$

$$\text{where } \#_h = |\{u \in P \mid \tau(\mu(u)) = \{h\}\}|.$$

In our example, $\gamma_{P_2}(x) = \gamma_{P_2}(y)$ and $\tau_1(x) \neq \tau_1(y)$ imply that P_2 is not satisfied by τ_1, which intuitively means that no expansion of τ_1 can match the atoms of q_3 as required by P_2. Moreover, $N_{\tau_3}(P_1) = s[\tau_3(x)] \cdot s[\tau_3(y)] = 4$ and $N_{\tau_3}(P_2) = s[\tau_3(x)] = 2$ count the expansions of τ_3 that comply with γ_{P_1} and γ_{P_2}, respectively; thus, the number of expansions of τ_3 complying with just P_1 is

$$K(P_1, P_1) \cdot N_{\tau_3}(P_1) + K(P_1, P_2) \cdot N_{\tau_3}(P_2) = 1 \cdot 4 - 1 \cdot 2 = 2.$$

The contribution of each expansion of τ_3 matching the atoms of q_3 to distinct triples is $F_{\tau_3}(P_1)$: set $\tau_3(\mu(u))$ contains just h_3 for each $u \in P_1$, so $\#_{h_3} = 2$, and $F_{\tau_3}(P_1) = 0.167$.

Thus, to compute $E_{q,S}$, we evaluate $\mu(q)$ in H and, for each answer τ, we combine $F_\tau(P)$ and $N_\tau(P')$ as follows.

Theorem 4.3. *The following identity holds:*

$$E_{q,S} = \sum_{\tau \in [\![\mu(q)]\!]_H} \sum_{P \in B_\tau} F_\tau(P) \cdot \sum_{\substack{P' \in B_\tau \\ P \preceq P'}} K(P, P') \cdot N_\tau(P').$$

Theorem 4.4 shows how to compute the variance of the cardinality, and it is proved using the well-known fact that $\sigma^2_X = E[X^2] - (E[X])^2$ holds for each random variable X.

Theorem 4.4. *Let q' be the query obtained from q by replacing each variable in $\mathsf{var}(q)$ with a fresh variable; then,*

$$\sigma^2_{q,S} = E_{q \cup q',S} - E^2_{q,S}.$$

The following definition identifies queries for which we can use the simpler formula (1). Queries q_1 and q_2 from Section 4.1 are μ-unification-free, whereas q_3 is not. Moreover, note that query $q_4 = \{\langle e_3, owns, x\rangle, \langle e_4, owns, y\rangle\}$ is also *not* μ-unification-free: its atoms unify *after* applying μ to q_4.

Definition 4.5. *Atoms a_1 and a_2 are* unifiable *if there exists a substitution κ such that $\kappa(a_1) = \kappa(a_2)$. Query q is* μ-unification-free *if no two distinct atoms of $\mu(q)$ are unifiable; otherwise, query q is* μ-unifiable.

If q is μ-unification-free, then P consisting of $\{a_i\}$ for each $a_i \in q$ is the only unifiable partition of q. Thus, P' in the last sum in Theorem 4.3 must be P, so the sum reduces to $N_\tau(P)$, which is equal to the first product in (1). Also, the atoms of $\mu(q)$ cannot be equated, so $\#_h = 1$ for each $h \in \tau(\mu(q))$ and $F_\tau(P)$ reduces to the second product in (1).

Proposition 4.6. *If query q is μ-unification-free, then formula (1) correctly computes $E_{q,S}$.*

Finally, we show that deciding whether $E_{q,S} = m/n$ for given integers m and n (i.e., the decision problem naturally corresponding to the problem of computing $E_{q,S}$) is in $\mathsf{P}^{\#\mathsf{P}[1]}$—that is, it can be solved in polynomial time with one call to a $\#\mathsf{P}$ oracle, where the latter is the class of counting problems associated with NP. It is known that $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}[1]} \subseteq \mathsf{PSPACE}$. Thus, solving our problem requires numbers of only polynomial length, which is interesting since the numbers in the numerator and the denominator of $E_{q,S}$ from Definition 3.4 can have exponential length. Please note that the size of q is the main source of complexity since q can be evaluated in S in time polynomial in the size of H.

Theorem 4.7. *The problem of deciding, given S, q, and integers m and n, whether $E_{q,S} = \frac{m}{n}$ holds is in $\mathsf{P}^{\#\mathsf{P}[1]}$, and is at least as hard as the problem of checking whether the number of answers to q on a graph is equal to a given integer.*

5. HYPERTREE DECOMPOSITIONS

To compute $E_{q,S}$, we must evaluate $\mu(q)$ in a summarisation graph H, which is worst-case polynomial in the size of H but exponential in the size of q. Even a small H is usually highly connected so, as we show in Section 7, this step can be slow. This problem is common to all multidimensional approaches mentioned in Section 1, but, to the best of our knowledge, it has not been considered thus far.

A query q can be evaluated efficiently using a bushy join plan $\mathcal{D}$ called a *hypertree (HT) decomposition* of q [13]. As we discus next, one can evaluate q in an RDF graph in time polynomial in the size of q and exponential in a parameter $\mathsf{hw}(\mathcal{D})$ called the hypertree-width, which measures the complexity of $\mathcal{D}$. Typical queries usually admit $\mathcal{D}$ with small $\mathsf{hw}(\mathcal{D})$, so even long queries can be evaluated efficiently.

We adapt these ideas and present an algorithm that, given a decomposition $\mathcal{D}$ of a μ-unification-free CQ q, computes $E_{q,S}$ in time exponential in $\mathsf{hw}(\mathcal{D})$ and polynomial in the size of q. We first review the definition of HT decompositions.

Definition 5.1. *A* hypertree decomposition *$\mathcal{D}$ of a CQ q is a tuple $\langle \mathcal{T}, X, p\rangle$ where $\mathcal{T}$ is a rooted tree with vertices $\mathsf{vert}(\mathcal{T})$ and, for each vertex $v \in \mathsf{vert}(\mathcal{T})$, function X labels v with a set of variables $X^v \subseteq \mathsf{var}(q)$, and function p labels v with a CQ $p^v \subseteq q$, such that*

1. *for each atom $a \in q$, there exists a vertex $v \in \mathsf{vert}(\mathcal{T})$ satisfying $a \in p^v$ and $\mathsf{var}(a) \subseteq X^v$;*
2. *for each $x \in \mathsf{var}(q)$, the set $\{v \in \mathsf{vert}(\mathcal{T}) \mid x \in X^v\}$ of vertices induces a (connected) subtree of $\mathcal{T}$; and*
3. *$X^v \subseteq \mathsf{var}(p^v)$ for each $v \in \mathsf{vert}(\mathcal{T})$.*

The hypertree-width *of $\mathcal{D}$ is $\mathsf{hw}(\mathcal{D}) = \max_{v \in \mathsf{vert}(\mathcal{T})} |p^v|$.*

A decomposition $\mathcal{D}$ with minimal $\mathsf{hw}(\mathcal{D})$ can be computed efficiently [13], and the standard algorithm evaluates q in an RDF graph by processing $\mathcal{D}$ bottom-up: for each vertex v, it evaluates p^v and reduces the result by semi-joins with the children, and then projects out the variables not occurring

Algorithm 1 Computing $E_{q,S}$ using a HT decomposition

Inputs:
$S = \langle H, w, \mu\rangle$: consistent summary
q : μ-unification-free conjunctive query
$\mathcal{D} = \langle \mathcal{T}, X, p\rangle$: hypertree decomposition of q

1: **for all** $v \in \mathsf{vert}(\mathcal{T})$ **do** $p^v_* := \emptyset$
2: **for all** $a \in q$ **do**
3: Add a to p^v_* for one arbitrary $v \in \mathsf{vert}(\mathcal{T})$ with $a \in p^v$
4: **for all** $v \in \mathsf{vert}(\mathcal{T})$ from the leaves upwards **do**
5: **if** parent v' of v exists **then** $Z := X^v \cap X^{v'}$
6: **else** $Z := \emptyset$
7: $M^v := \{\langle \tau|_{X^v},\ \prod_{a \in p^v_*} \frac{w[\tau(\mu(a))]}{s[\tau(\mu(a))]}\rangle \mid \tau \in [\![\mu(p^v)]\!]_H\}$
8: $E^v := \mathtt{Grp}(\mathtt{Pr}_{Z,S}(M^v \bowtie E^{v_1} \bowtie \cdots \bowtie E^{v_\ell}))$
where $v_1, \ldots, v_\ell$ are all the children of v in $\mathcal{T}$
9: **return** e such that $\langle \tau_\emptyset, e\rangle \in E^{v_0}$ for v_0 the root of $\mathcal{T}$

in the parent of v. Condition 2 of Definition 5.1 ensures the correctness of the last step and the desired complexity.

We adapt these ideas as follows. First, our approach works on pairs $\langle \tau, e\rangle$ of an answer τ to p^v (and thus of a partial answer to q) and a rational number e that accumulates the products from formula (1). Second, projection can produce tuples $\langle \tau, e_i\rangle$ that coincide on τ and possibly even on e_i, so the algorithm keeps multisets of $\langle \tau, e\rangle$ (as in the bag semantics of SQL and SPARQL) and it aggregates all e_i after each projection. Third, when projecting a variable x, it must incorporate the S-size of $\tau(x)$ into the relevant factors e.

Definition 5.2. *Let L, L_1, and L_2 be finite mutisets of pairs $\langle \tau, e\rangle$ of a substitution τ and a rational number e.*

The grouping $\mathtt{Grp}(L)$ *of L is the set containing $\langle \tau, e\rangle$ for each τ such that $\langle \tau, e_i\rangle \in L$, and e is the sum of all these e_i.*

The projection $\mathtt{Pr}_{Z,S}(L)$ *of L for Z a set of variables and S a summary is the multiset containing one*

$$\langle \tau|_Z, e \cdot \prod_{x \in \mathsf{dom}(\tau) \setminus Z} s[\tau(x)]\rangle$$

for each occurrence of $\langle \tau, e\rangle \in L$, where $\tau|_Z$ is the substitution obtained from τ by restricting its domain to Z, and $s[\tau(x)]$ is the S-size of the bucket $\tau(x)$.

The join $L_1 \bowtie L_2$ *of L_1 and L_2 is the multiset containing one $\langle \tau_1 \cup \tau_2, e_1 \cdot e_2\rangle$ for each occurrence of $\langle \tau_1, e_1\rangle \in L_1$ and each occurrence of $\langle \tau_2, e_2\rangle \in L_2$ such that $\tau_1(x) = \tau_2(x)$ holds for each $x \in \mathsf{dom}(\tau_1) \cap \mathsf{dom}(\tau_2)$.*

Algorithm 1 computes $E_{q,S}$ of a μ-unification-free CQ q as follows. Since $\mathcal{D}$ can assign a query atom a to multiple p^v, the algorithm first assigns each $a \in q$ to precisely one vertex in $\mathcal{D}$ (lines 1–3) so that the factor for a is added only once. The algorithm then processes each vertex v of $\mathcal{D}$ bottom-up (lines 4–8): it computes the variables Z relevant to the parent of v if one exists (lines 5–6), evaluates p^v in H and adds the appropriate factors (line 7), and then reduces the result by semi-joins with the children of v and projects the result to Z (line 8). The set E^{v_0} for v_0 the root of $\mathcal{D}$ contains precisely one $\langle \tau_\emptyset, e\rangle$ where $\tau_\emptyset$ is the empty substitution, and e is the expectation of q. Theorem 5.3 proves the algorithm correct by pushing sums into products in formula (1).

Theorem 5.3. *Algorithm 1 computes $E_{q,S}$ in time exponential in $\mathsf{hw}(\mathcal{D})$ but polynomial in the size of q and S.*

If q is μ-unifiable, unifying atoms of $\mu(q)$ can be assigned to distinct p^v, so we cannot compute $F_\tau(P)$, $K(P, P')$, and $N_\tau(P')$ locally. However, we strongly believe we can overcome this problem and handle arbitrary CQs.

6. COMPUTING THE SUMMARY

We now consider computing S from G. Algorithms for unlabelled graphs [21, 25, 30] are not directly applicable to RDF. We adapted the SNAP [34] technique to RDF, but it produced very large summaries. Finally, at the time our paper was written, the technical details and the implementation of a promising RDF-specific summarisation technique [8] were unavailable. Thus, we developed two simple algorithms that merge resources based on their similarity.

Our first approach assigns to each resource $d \in \mathsf{res}(G)$ a *type* $\mathtt{t}_d = \langle C, O, I\rangle$ with the following components:

$$C = \begin{cases} \{o \in \mathsf{res}(G) \mid \langle d, \mathit{rdf{:}type}, o\rangle \in G\} & \text{if } d \text{ is a URI} \\ \{\text{the datatype of } d\} & \text{if } d \text{ is a literal,} \end{cases}$$

$$O = \{p \in \mathsf{res}(G) \mid \exists o \in \mathsf{res}(G) \text{ such that } \langle d, p, o\rangle \in G\},$$

$$I = \{p \in \mathsf{res}(G) \mid \exists s \in \mathsf{res}(G) \text{ such that } \langle s, p, d\rangle \in G\}.$$

If $\mathtt{t}_{d_1} = \mathtt{t}_{d_2}$, then resources d_1 and d_2 participate in the same classes/datatypes, outgoing predicates, and incoming predicates, and are thus likely similar. The *typed summary* $S = \langle H, w, \mu\rangle$ of G then merges all resources with the same type into a single bucket, with the exception of resources used in G as classes or predicates. More precisely, for each resource $d \in \mathsf{res}(G)$, we define μ as follows.

- If d occurs in G in a triple of the form $\langle s, d, o\rangle$ or $\langle s, \mathit{rdf{:}type}, d\rangle$, we define $\mu(d) = d$.
- Otherwise, we define $\mu(d) = b_{\mathtt{t}_d}$, where $b_{\mathtt{t}_d}$ is a distinct bucket uniquely associated with $\mathtt{t}_d$.

We finally define $H = \mu(G)$ and set w accordingly. This computation clearly requires time linear in the size of G.

As our experiments in Section 7 show, the typed summary often produces very good estimations; however, if G contains many types, the typed summary can be large. Therefore, we developed an algorithm that, given an arbitrary summary $S = \langle H, w, \mu\rangle$, further reduces the size of S until the number of triples meets a predetermined space budget.

To identify similar buckets in S, we define the *vicinity* $V_S(b) = V_S^{out}(b) \cup V_S^{in}(b)$ of a bucket $b \in \mathsf{res}(H)$ as

$$V_S^{out}(b) = \{\langle b_p, b_o\rangle \mid \langle b, b_p, b_o\rangle \in H \wedge b_p \neq \mu(\mathit{rdf{:}type})\},$$

$$V_S^{in}(b) = \{\langle b_i, b_p\rangle \mid \langle b_i, b_p, b\rangle \in H \wedge b_p \neq \mu(\mathit{rdf{:}type})\}.$$

Since $V_S(b)$ describes the connections of b, given buckets $b_1, b_2 \in \mathsf{res}(H)$, the degree of commonality between $V_S(b_1)$ and $V_S(b_2)$ is indicative of the similarity of b_1 and b_2. This is captured by the *Jaccard index* [23] of b_1 and b_2 as

$$\mathcal{J}_S(b_1, b_2) = \frac{|V_S(b_1) \cap V_S(b_2)|}{|V_S(b_1) \cup V_S(b_2)|}$$

if $V_S(b_1) \cup V_S(b_2) \neq \emptyset$, and otherwise $\mathcal{J}_S(b_1, b_2) = 1$. Thus, we can naïvely reduce the size of S by repeatedly merging the pair of buckets b_1 and b_2 with maximal $\mathcal{J}_S(b_1, b_2)$, but this is impractical for two reasons. First, computing $\mathcal{J}_S(b_1, b_2)$ requires iterating over $V_S(b_1)$ and $V_S(b_2)$, which can be large. Second, the number of pairs of b_1 and b_2 can be prohibitively large in even moderately sized summaries.

Algorithm 2 MinHash summary refinement algorithm

Inputs:
$S = \langle H, w, \mu \rangle$: summary
$\mathcal{B}_1, \ldots, \mathcal{B}_Q$: pairwise disjoint subsets of $\mathsf{res}(H)$
$m \times n$: size of the MinHash scheme
ℓ : the number of bins
Target : target size of the summary
Threshold : similarity lower threshold

Generate a MinHash scheme $\mathcal{F}$ of size $m \times n$
Generate a locality sensitive hash LSH : $\mathbb{N}^n \to [1, \ell]$
loop
  Queue $:= \emptyset$
  for all $k := 1$ **to** Q **do**
    for all $b \in \mathcal{B}_k$ **do** Compute $\mathcal{M}^b$ using $\mathcal{F}$ and S
    for $i := 1$ **to** m **do**
      for $p := 1$ **to** ℓ **do** $Bin_p := \emptyset$
      for all $b \in \mathcal{B}_k$ **do** Add b to $Bin_{\text{LSH}(\mathcal{M}^b[i])}$
      for $p := 1$ **to** ℓ **do**
        for $b_1, b_2 \in Bin_p$ with $b_1 \neq b_2$ **do**
          if $\hat{\mathcal{J}}_S(b_1, b_2) \geq$ *Threshold* **then**
            Add $\langle b_2, b_1 \rangle$ to *Queue*
            Remove b_2 from $\mathcal{B}_k$
            Remove b_1 and b_2 from Bin_p
  if *Queue* $= \emptyset$ **then fail**
  for all $\langle b_2, b_1 \rangle \in$ *Queue* **do** Merge b_2 into b_1 in S
  if $|H| \leq$ *Target* **then return** S

We address the first problem using MinHashing [23], which approximates $\mathcal{J}_S(b_1, b_2)$. For integer parameters m and n, we generate a *MinHash scheme $\mathcal{F}$ of size $m \times n$*, which is a $m \times n$ matrix of hash functions chosen uniformly at random with replacement where each $\mathcal{F}[i, j]$ maps pairs of resources to natural numbers. The *signature* of a bucket $b \in \mathsf{res}(H)$ on $\mathcal{F}$ and S is the $m \times n$ matrix $\mathcal{M}^b$ where, for all i and j,

$$\mathcal{M}^b[i, j] = \min_{\alpha \in V_S(b)} \mathcal{F}[i, j](\alpha)$$

if $V_S(b) \neq \emptyset$, and otherwise $\mathcal{M}^b[i, j] = \infty$. It is known that

$$\hat{\mathcal{J}}_S(b_1, b_2) = \frac{|\{\langle i, j \rangle \mid \mathcal{M}^{b_1}[i, j] = \mathcal{M}^{b_2}[i, j]\}|}{m \times n}$$

approximates well the Jaccard index of buckets b_1 and b_2 [23]. For $i \in [1, m]$, we write $\mathcal{M}^b[i]$ for the i-th row of $\mathcal{M}^b$.

We address the second problem using locality sensitive hashing [23]: we generate a hash function LSH for $\mathcal{M}^b[i]$ and use it to assign each bucket b to a bin Bin_j, $1 \leq j \leq \ell$. For $b_1, b_2 \in Bin_j$, it is known that $\hat{\mathcal{J}}_S(b_1, b_2)$ is likely to be high so, while we still need to check $\hat{\mathcal{J}}_S(b_1, b_2)$, we can consider pairs of b_1 and b_2 from each bin rather than the entire graph.

Algorithm 2 uses these ideas to refine a summary S. It takes pairwise disjoint sets of buckets $\mathcal{B}_1, \ldots, \mathcal{B}_Q$ of S, and it merges buckets only inside each $\mathcal{B}_k$; this can be used to prevent merging certain kinds of buckets. For each $\mathcal{B}_k$, the algorithm first computes the MinHash signature of each $b \in \mathcal{B}_k$ and, by applying locality sensitive hashing to each $\mathcal{M}^b[i]$, it adds b to a bin. Next, it identifies all pairs of sufficiently similar buckets b_1 and b_2, adds them to a queue of pairs to be merged, and removes them from further consideration. The relevant buckets are then merged after each iteration, which improves the performance compared to merging buckets immediately. This is repeated until either H fits into the space budget, or there are no candidates for merging. Since the space budget is checked after each iteration, the resulting summary can contain fewer triples than *Target*.

7. EXPERIMENTAL EVALUATION

We implemented our techniques in a prototype system called SumRDF. The system is written in Java, and it evaluates queries in graphs and summaries in RAM using left-deep index nested loop joins. We evaluated our system with four distinct objectives. First, we compared the accuracy of the estimates produced by SumRDF and several state of the art RDF and relational systems. Second, we investigated the impact of the correct handling of μ-unifiable queries; these correspond to self-joins and are thus difficult to estimate. Third, we investigated whether the standard deviation provides a measure of estimation accuracy. Fourth, to show practical feasibility, we measured the estimation times and the improvements of using HT decompositions. We ran all our experiments on a Dell computer with 100 GB of RAM, 64-bit Fedora 22, and two Xeon X5667 processors with 16 physical cores. Our system, all test data, all queries, and the results of all measurements are available online.[2]

7.1 Test Data

We evaluated our approach using three well-known RDF benchmarks shown in Table 1. Queries are grouped by their shape as linear, star, snowflake (joined stars), and complex (e.g., cyclic), and for each group the table shows the number of all and μ-unifiable queries, and the min/max number of atoms. Each query is identifiable by a unique ID (QID).

LUBM is a popular synthetic benchmark comprising a data generator, an OWL 2 ontology, and 14 test queries. The ontology must be taken into account for the queries to produce nonempty results, so we generated an RDF graph with 500 universities and extended it with triples implied by the ontology, thus obtaining 91M triples. The 14 test queries are relatively simple and are all μ-unification-free, so we handcrafted 24 additional queries, five of which are μ-unifiable.

WatDiv [4] is another synthetic benchmark designed to stress-test RDF systems. We used its data generator to obtain an RDF graph with 108M triples. The benchmark also includes three queries and 17 query templates with one parameter, and a generator that replaces the parameter with resources from the RDF graph. We instantiated each query template into five concrete queries and, if the parameter was not in the class position, we obtained one more query by replacing the parameter with a fresh variable. We thus obtained 103 queries, all of which are μ-unification-free.

DBLP is a prominent bibliography database that has been converted into an RDF graph of 55M triples.[3] We used six queries that have already been considered in the literature [37], and we handcrafted nine additional queries. We thus obtained 15 queries, eight of which are μ-unifiable.

7.2 Summary Construction

For each test RDF graph, we constructed a summary with properties shown in Table 1 using a space budget of 15,000

[2] http://www.cs.ox.ac.uk/isg/tools/SumRDF/
[3] http://datahub.io/dataset/l3s-dblp

	Original graph G		Summary S				Queries by type															
	Nr resources	Nr triples	Nr buckets	Nr triples	Reduction factor	Construction time	Linear				Star				Snowflake				Complex			
							total	μ-un.	min	max	total	μ-un.	min	max	total	μ-un.	min	max	total	μ-un.	min	max
LUBM	16.4M	91.1M	76	446	204K	121s	16	3	1	6	2	0	4	10	6	0	4	10	14	2	4	9
WatDiv	10.2M	108.9M	339	7,039	15.5K	722s	30	0	2	3	40	0	2	9	30	0	5	9	3	0	6	10
DBLP	25.4M	55.5M	498	10,432	5.3K	106s	4	0	1	4	1	1	3	3	2	1	6	7	8	6	5	11

Table 1: Dataset, summary, and query statistics

triples. The typed summaries of LUBM and DBLP contain 446 and 10,432 triples, respectively, which fits comfortably in the space budget. In contrast, the typed summary of WatDiv contains more than 41.3M triples, so we further compressed the typed summary to $7,039$ triples using Algorithm 2 and $m = 40$, $n = 2$, $\ell = 32,768$, $\mathit{Threshold} = 0.15$, and each $\mathcal{B}_k$, $1 \leq k \leq Q$, containing all buckets of the typed summary of the same class; the latter is important since merging resources connected via *rdf:type* to different classes considerably reduces the accuracy estimate. As Table 1 shows, we compressed each RDF graph by between three and five orders of magnitude in at most ten minutes.

7.3 Comparison Systems

As we discussed in Section 1, the existing schema-level synopses require users to specify the relevant join paths that typically cannot include self-joins, but the schema-less nature of RDF makes this difficult. Moreover, these techniques have not been implemented in systems that we could readily use in our evaluation. Thus, we based our evaluation on the following three prominent RDF and relational systems.

RDF-3X [27] v0.3.7 is a state of the art RDF system whose cardinality estimator was specifically tailored to RDF and was shown to outperform the related approaches.

PostgreSQL v9.4.5 features a conventional cardinality estimator based on one-dimensional histograms. The system collected all statistics automatically.

SystemX is a commercial RDBMS with a highly tuned cardinality estimator that supports two modes: in the *static mode* the system uses statistics collected automatically prior to estimation, and in the *dynamic mode* the system can sample the data during estimation if it doubts the accuracy of its estimates. We used the maximum sampling level.

To store RDF data into RDBMSs, we replaced all resources with unique integers, and then we considered two storage modes. First, we stored all triples in one ternary *triple table*, so all queries amount to self-joins over this table. Second, we *vertically partitioned* the data by storing $\langle s, \mathit{rdf{:}type}, o\rangle$ as tuple $\langle s\rangle$ in a unary relation o, and $\langle s, p, o\rangle$ with $p \neq \mathit{rdf{:}type}$ as tuple $\langle s, o\rangle$ in a binary relation p. Using multiple tables allows for more accurate statistics. We thus obtained a total of seven comparison systems: RDF-3X, two (P-T and P-V) PostgreSQL variants, and two static (XS-T and XS-V) and two dynamic (XD-T and XD-V) SystemX variants. All systems provide an `EXPLAIN` facility that outputs the cardinality estimate of a query.

7.4 Accuracy Experiments

We measured for each system and each query the *q-error* $\max(N/E, E/N)$, where N and E are the exact and estimated cardinalities of the query, respectively. The q-error shows the difference in the orders of magnitude regardless of whether E over- or undershoots N, and it is widely used in the literature [26]. Unlike other systems, the estimates of SumRDF can be between zero and one, which skews the q-error; hence, we rounded all such estimates to one.

Showing the q-errors for each of the 156 test queries would not be very informative, so in Figure 5 we show the q-error distribution per dataset and system. Each bar chart shows the percentages of the queries with q-error in the intervals listed in the legend; and each box plot groups the q-errors of a system by quartiles: the box shows the lower and upper quartiles, the line dividing the box shows the median, the line whiskers show the minimum and maximum, and the diamond mark shows the average. Please note that the maximum q-errors of P-T on LUBM and WatDiv and of XS-T on LUBM fall considerably after the end of the y axis.

As one can see, SumRDF is considerably more accurate on LUBM and WatDiv than the other systems, with the exception of SystemX with dynamic sampling on WatDiv; moreover, SumRDF achieves the lowest maximum and average error on DBLP. By further analysing the results, we noticed that SumRDF consistently undershoots on all queries with q-error above 10. We could not detect a similar pattern for other systems: they undershoot on some, but overshoot on other queries; moreover, the maximum q-errors were always due to overshooting. Interestingly, different variants of the same system would differ greatly on the same query; for example, P-T might overshoot while P-V would undershoot, and similarly for SystemX. Finally, for each dataset and each system, there is a query where the system is most accurate, and another query where the system is least accurate. In other words, no system consistently outperformed or underperformed all others, and so the aggregate information shown in Figure 5 seems most informative.

We further analysed *difficult queries*, which we defined as queries with the q-error of SumRDF above 10. For each of these queries, most other systems would exhibit a q-error above 10 as well. In LUBM we found five difficult queries, with q-errors between 5.5K and 10K; three are triangular queries, and two are cyclic queries. In WatDiv, six difficult queries were obtained from query template L2: five queries obtained by instantiating the parameter have q-errors between 120 and 630, and the query obtained by replacing the parameter with a variable has a q-error of 16.6K. In addition, there are five difficult queries on WatDiv with q-errors between 10 and 20. Finally, on DBLP, three difficult queries have q-errors between 10 and 20, and additional seven queries have q-errors above 20; and all these are either cyclic or have many (up to three) selections.

Our analysis showed that selections are sometimes hard for SumRDF. For example, the five queries obtained from template L2 of WatDiv all contain atom $\langle x, \mathit{likes}, \mathit{Product0}\rangle$, and the q-error for just that atom alone is 5.6K. This is because the bucket of *Product0* is large, and predicate *likes* is not equally distributed among the resources in the bucket; moreover, the bucket size occurs in the denominator of our formulas, so SumRDF undershoots by a large factor. A possible way to overcome this might be to refine buckets

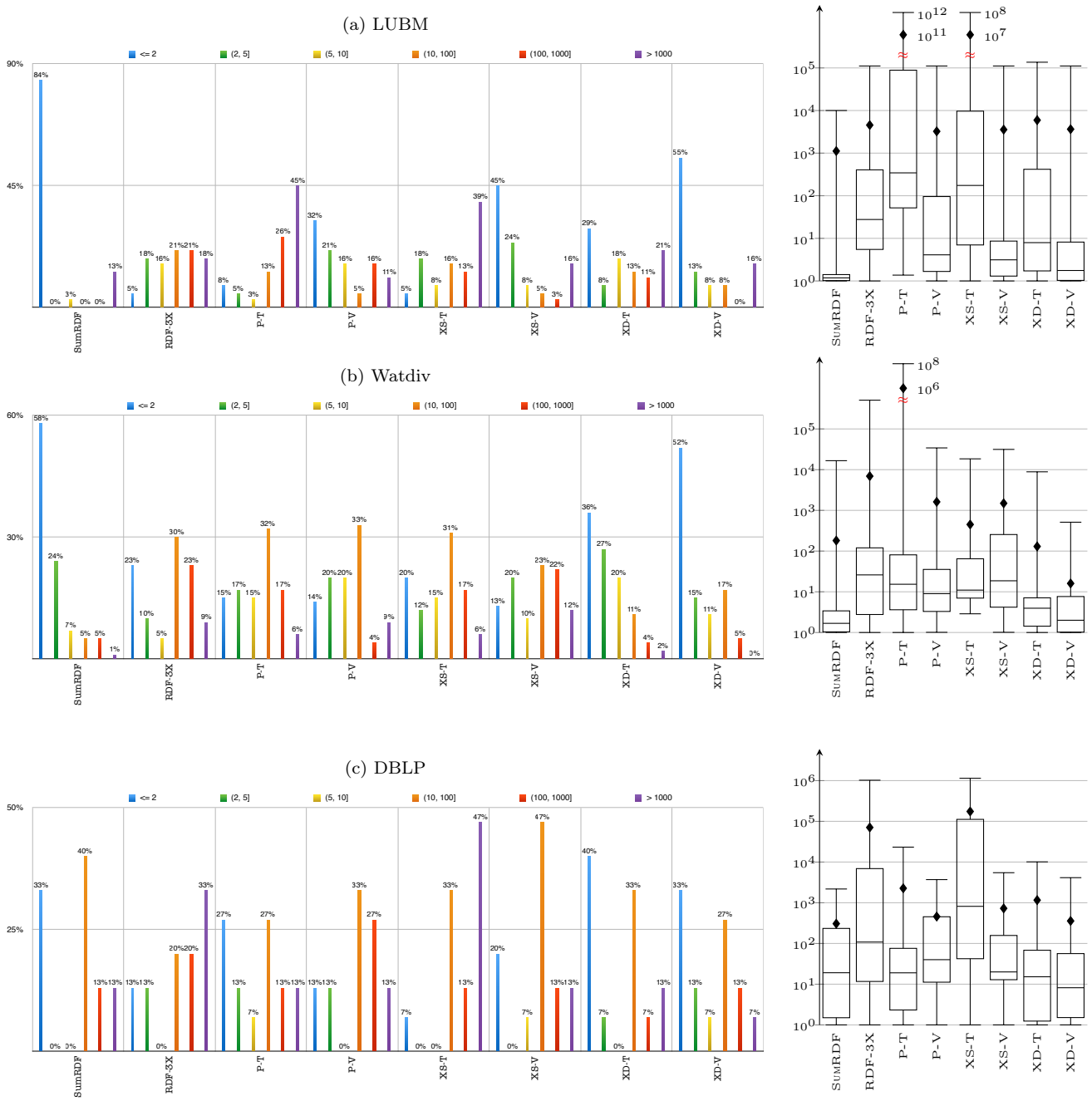

Figure 5: Histograms and box plots of the distribution of the q-error on test datasets

		LUBM				WatDiv				DBLP			
		min	median	avg	max	min	median	avg	max	min	median	avg	max
without HT	all	2.3	24.1	565.0	16.5K	3.0	120.9	25.8K	15.3M	38.5	1.9K	144.6K	1.9M
	μ-unification-free	2.3	19.4	597.7	16.5K					38.5	1.0K	1.0K	2.1K
with HT	μ-unification-free	0.1	18.3	24.7	94.9	9.5	40.2	108.0	806.0	23.3	443.6	504.6	1.1K

Table 2: Query estimation times in microseconds with and without HT

	QID	Eq. (1)	SumRDF	RDF-3X	P-T	P-V	XS-T	XS-V	XD-T	XD-V
LUBM	U1	13.9K	7.6K	71.2	3.5T	122.5	26.3G	2.6K	17.5K	3.4K
	U2	111.0K	9.2K	111.0K	7.9T	111.0K	12.3M	111.0K	982.0	111.0K
	U3	1.0	1.4	254.5	25G	2.7	87.3M	1.2	4.8K	1.0
	U4	1.2	1.8	455.4	650G	9.8	49.2M	1.1	2.7K	1.2
	U5	1.1	1.0	5.9	11.2K	1.6	10.2K	1.5	12.2K	1.5
DBLP	D1	1.3	1.3	9.9K	106.3	3.4	220.4K	1.4	2.8	2.8
	D2	1.7	1.7	1.0M	23.2K	40.0	1.0M	8.8	15.2	3.2
	D3	238.0	19.1	11.9K	45.4	238.0	238.0	238.0	79.3	238.0
	D4	467.7	411.7	3.9K	432.1	1.0K	213.5K	5.5K	6.3K	55.8
	D5	1.0K	1.0K	313.3	3.9	344.4	4.1K	4.1K	2.0	4.1K
	D6	57.0	57.0	57.0	2.7	57.0	57.0	57.0	57.0	57.0
	D7	2.2K	2.2K	10.1K	10.1K	561.3	10.1K	76.5	10.1K	8.2
	D8	817.0	817.0	817.0	38.9	817.0	817.0	817.0	817.0	817.0
min		1.1	1.0	5.9	2.7	1.6	57.0	1.1	2.0	1.1
med		238.0	57.0	817.0	10.1K	122.5	213.5K	76.6	982.1	55.8
avg		9.9K	1.6K	90.1K	940.8G	8.8K	31.8M	9.5K	4.3K	9.2K
max		111K	9.2K	1.0M	7.9T	111.0K	263.4M	111.0K	17.6K	111.0K

Table 3: The q-error for μ-unifiable queries

Probability $\wp$	99%			95%			90%		
Q-Error ε	10	100	1000	10	100	1000	10	100	1000
$\frac{\varepsilon-1}{\varepsilon\cdot\sqrt{1-\wp}}$ (%)	9.00	9.90	9.99	20.13	22.15	22.35	28.48	31.33	31.61
LUBM (34)	16 (0)	16 (0)	16 (0)	16 (0)	16 (0)	16 (0)	23 (1)	23 (1)	24 (1)
WatDiv (81)	29 (0)	29 (0)	29 (0)	37 (2)	38 (0)	38 (0)	44 (2)	45 (0)	45 (0)
DBLP (9)	4 (1)	4 (0)	4 (0)	5 (1)	5 (0)	5 (0)	5 (1)	5 (0)	5 (0)

Table 4: The confidence interval for the q-error

containing resources with such unequal distributions. SystemX with dynamic sampling was more accurate on selections since, when the statistics were insufficiently precise, the system just additionally sampled the data.

7.5 Impact of μ-Unification

Formula (1) for μ-unifiable queries is much simpler, and using Algorithm 1 it can be evaluated more efficiently than the formula for general CQs. To see whether the general formula is worth the added complexity, for each μ-unifiable query, Table 3 compares the q-error of the estimate computing using formula (1) with the q-errors of all other systems (where SumRDF uses the correct formula). The table also aggregates the q-errors for each system. WatDiv does not occur in the table since its queries are μ-unification-free.

As one can see, μ-unifiable queries tend to be difficult: for each query, at least one system has q-error above 1,000. However, handling μ-unifiable queries correctly reduces the maximum error by an order of magnitude, and it also considerably improves the median and the average. Finally, SumRDF considerably outperforms all other systems on all aggregate metrics, apart from the median of XD-V. In contrast, the aggregate metrics of formula (1) seem similar to P-V. These results show that our principled approach to μ-unifiable queries contributes substantially to the accuracy.

7.6 Standard Deviation

As we discussed in Section 3, Chebyshev's inequality determines a confidence interval for $E_{q,S}$, and we next reformulate it to obtain a confidence interval on the q-error.

Proposition 7.1. *The q-error of any CQ q is less than $\varepsilon \geq 1$ with probability at least $\wp$ if $\frac{\sigma_{q,S}}{E_{q,S}} < (1-\frac{1}{\varepsilon})\sqrt{1-\wp}$.*

Equivalently, if $\sigma_{q,S}/E_{q,S} < (1-1/\varepsilon)\sqrt{1-\wp}$ holds, the q-error of a CQ q is less than ε on at least $\wp \cdot |R_S|$ graphs from R_S. As one might intuitively expect, $\sigma_{q,S}$ should cover a smaller portion of $E_{q,S}$ for higher $\wp$ and lower ε.

In Table 4 we show the bounds on $\sigma_{q,S}/E_{q,S}$ for nine combinations of $\wp$ and ε; moreover, for each dataset, we show the number of queries satisfying this ratio and (in parentheses) for how many of these the q-error exceeds ε. For example, for $\wp = 95\%$ and $\varepsilon = 10$, there are 37 WatDiv queries where $\sigma_{q,S}$ is within 20.13% of $E_{q,S}$, and the q-error is above 10 on two of these 37 queries. The computation of $\sigma_{q,S}$ timed out on some queries after five minutes: query $q \cup q'$ from Theorem 4.4 is always μ-unifiable and tends to be very complex. Thus, next to the name of each dataset, we show in parentheses the number of queries on which the computation of $\sigma_{q,S}$ succeeded. As we discuss in Section 7.7, we are confident that, by extending Algorithm 1 to μ-unifiable queries, we can make the computation of $\sigma_{q,S}$ practically feasible.

Thus, with probability 99%, we bounded correctly the q-error by 10 for 47% of LUBM and 35% of WatDiv queries; and we did the same, but for q-error of 100, for 44% of DBLP queries. We can bound the q-error of more queries by reducing $\wp$ and increasing ε, eventually covering 70% of LUBM, 55% of WatDiv, and 55% of DBLP queries. As expected, decreasing $\wp$ increases the likelihood that the q-error on our given graph exceeds ε. Finally, as Table 4 shows, factor $(\varepsilon-1)/(\varepsilon\cdot\sqrt{1-\wp})$ depends much less on ε than on $\wp$, so using large values of ε does not seem to be necessary.

Please note that a higher $\sigma_{q,S}/E_{q,S}$ ratio does not necessarily imply large q-error: while the cardinality of q on the graphs in R_S might be more variable, the query might just have precisely $E_{q,S}$ answers on our specific graph. Nevertheless, the q-error was usually significant if $\sigma_{q,S}/E_{q,S} \geq 1$.

7.7 Estimation Times

To investigate practicability in terms of estimation times, we measured the average estimation time for each query over 20 repeated runs, and in Table 2 we report the minimum, median, average, and maximum times for all queries and all μ-unification-free queries. The median on LUBM and WatDiv is below 120 microseconds, but the maximum can be high, and on DBLP it can reach seconds. This is because summarisation tends to make H highly connected, and so $[\![\mu(q)]\!]_H$ can be large, particularly for big queries.

However, Table 2 also shows that Algorithm 1 can dramatically improve estimation times: the maximum is reduced by a factor 18.9K on WatDiv, and by 175 on LUBM; gains on DBLP are more modest, but still noticeable. This is very encouraging, and so extending Algorithm 1 to μ-unifiable queries is an important goal for our future work.

Please note that we have not tuned our implementation for performance as this was not our main goal in this paper. Moreover, query planners call a cardinality estimator repeatedly for many subqueries of the query being planned, and one can significantly reduce the cumulative estimation time by caching computation results between estimator calls.

8. CONCLUSION & OUTLOOK

We presented a new approach for estimating the cardinality of CQs on RDF graphs. Our approach is based on graph summarisation and features a precise semantics allowing us to formalise the estimation problem as computing the expected query cardinality in a family of graphs. We can also provide statistical confidence in our estimate by computing the standard deviation. Finally, we showed experimentally that our approach considerably outperforms state of the art RDF and relational cardinality estimators.

We see many exciting opportunities for future work. On the theoretical side, supporting range queries should be easy, but adding `DISTINCT` and aggregation is likely be more involved. We shall also investigate ways to incorporate further statistical information about the data and thus reduce the number of graphs represented by a summary. Another important challenge is to extend the hypertree decomposition algorithm to μ-unifiable queries. Finally, we shall extend our approach to the relational data model as well. On the practical side, we shall further analyse the sources of errors and develop ways of computing more precise summaries, possibly by incorporating variants of dynamic sampling.

9. REFERENCES

[1] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. Scalable Semantic Web Data Management Using Vertical Partitioning. In *VLDB*, 2007.

[2] A. Aboulnaga and S. Chaudhuri. Self-tuning Histograms: Building Histograms Without Looking at Data. In *SIGMOD*, 1999.

[3] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. Join Synopses for Approximate Query Answering. *SIGMOD Rec.*, 28(2), 1999.

[4] G. Aluç, O. Hartig, M. T. Özsu, and K. Daudjee. Diversified Stress Testing of RDF Data Management Systems. In *ISWC*, 2014.

[5] N. Bruno. *Statistics on Query Expressions in Relational Database Management Systems*. PhD thesis, Columbia University, New York, NY, USA, 2003.

[6] N. Bruno and S. Chaudhuri. Conditional Selectivity for Statistics on Query Expressions. In *SIGMOD*, 2004.

[7] N. Bruno, S. Chaudhuri, and L. Gravano. STHoles: A Multidimensional Workload-aware Histogram. *SIGMOD Rec.*, 30(2), 2001.

[8] Š. Čebirić, F. Goasdoué, and I. Manolescu. Query-oriented Summarization of RDF Graphs. *P-VLDB.*, 8(12), 2015.

[9] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim. Approximate query processing using wavelets. *VLDB J.*, 10(2), 2001.

[10] C. A. Galindo-Legaria, M. Joshi, F. Waas, and M. Wu. Statistics on Views. In *VLDB*, 2003.

[11] M. Garofalakis and P. B. Gibbons. Wavelet Synopses with Error Guarantees. In *SIGMOD*, 2002.

[12] L. Getoor, B. Taskar, and D. Koller. Selectivity Estimation Using Probabilistic Models. *SIGMOD Rec.*, 30(2), 2001.

[13] G. Gottlob, Z. Miklós, and T. Schwentick. Generalized hypertree decompositions: NP-hardness and tractable variants. *J. ACM*, 56(6), 2009.

[14] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi. Approximating Multi-dimensional Aggregate Range Queries over Real Attributes. In *SIGMOD*, 2000.

[15] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi. Selectivity estimators for multidimensional range queries over real attributes. *VLDB J.*, 14(2), 2005.

[16] S. Harris and A. Seaborne. SPARQL 1.1 Query Language, W3C Recommendation, March 21 2013.

[17] H. Huang and C. Liu. Estimating Selectivity for Joined RDF Triple Patterns. In *CIKM*, 2011.

[18] Y. Ioannidis. The History of Histograms (Abridged). In *VLDB*, 2003.

[19] Y. E. Ioannidis and S. Christodoulakis. On the Propagation of Errors in the Size of Join Results. In *SIGMOD*, 1991.

[20] J.-H. Lee, D.-H. Kim, and C.-W. Chung. Multi-dimensional Selectivity Estimation Using Compressed Histogram Information. In *SIGMOD*, 1999.

[21] K. LeFevre and E. Terzi. GraSS: Graph Structure Summarization. In *SDM*, 2010.

[22] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How Good Are Query Optimizers, Really? *P-VLDB*, 9(3), 2015.

[23] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of Massive Datasets, 2nd Ed.* Cambridge University Press, 2014.

[24] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-based Histograms for Selectivity Estimation. In *SIGMOD*, 1998.

[25] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph Summarization with Bounded Error. In *SIGMOD*, 2008.

[26] T. Neumann and G. Moerkotte. Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins. In *ICDE*, 2011.

[27] T. Neumann and G. Weikum. The RDF-3X engine for scalable management of RDF data. *VLDB J.*, 19(1), 2010.

[28] V. Poosala, P. J. Haas, Y. E. Ioannidis, and E. J. Shekita. Improved Histograms for Selectivity Estimation of Range Predicates. In *SIGMOD*, 1996.

[29] V. Poosala and Y. E. Ioannidis. Selectivity Estimation Without the Attribute Value Independence Assumption. In *VLDB*, 1997.

[30] M. Riondato, D. García-Soriano, and F. Bonchi. Graph Summarization with Quality Guarantees. In *ICDM*, 2014.

[31] J. Spiegel and N. Polyzotis. TuG Synopses for Approximate Query Answering. *ACM TODS*, 34(1), 2009.

[32] M. Stocker, A. Seaborne, A. Bernstein, C. Kiefer, and D. Reynolds. SPARQL Basic Graph Pattern Optimization Using Selectivity Estimation. In *WWW*, 2008.

[33] P. Terlecki, H. Bati, C. A. Galindo-Legaria, and P. Zabback. Filtered statistics. In *SIGMOD*, 2009.

[34] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient Aggregation for Graph Summarization. In *SIGMOD*, 2008.

[35] K. Tzoumas, A. Deshpande, and C. S. Jensen. Efficiently Adapting Graphical Models for Selectivity Estimation. *VLDB J.*, 22(1), 2013.

[36] F. Yu, W.-C. Hou, C. Luo, D. Che, and M. Zhu. CS2: A New Database Synopsis for Query Estimation. In *SIGMOD*, 2013.

[37] L. Zou, M. T. Özsu, L. Chen, X. Shen, R. Huang, and D. Zhao. gStore: A Graph-based SPARQL Query Engine. *VLDB J.*, 23(4), 2014.

APPENDIX

A. PROOF FOR SECTION 3

To prove Proposition 3.3, we start by defining the inverse of the summarisation function μ in a summary $S = \langle H, w, \mu \rangle$.

DEFINITION A.1. *For $S = \langle H, w, \mu \rangle$ a summary, the* inverse function μ^- *of μ maps each bucket $b \in \mathsf{res}(H)$ and each triple $h \in H$ as follows:*

$$\mu^-(b) = \{d \in \mathsf{dom}(\mu) \mid \mu(d) = b\},$$
$$\mu^-(h) = \{f \in \mathsf{dom}(\mu) \times \mathsf{dom}(\mu) \times \mathsf{dom}(\mu) \mid \mu(f) = h\}.$$

Please note that, by Definition 3.1 of size $s[\cdot]$, we have that $s[b] = |\mu^-(b)|$ and $s[h] = |\mu^-(h)|$, for each $b \in \mathsf{res}(H)$ and each $h \in H$. The next lemma shows that the consistency of a summary can be determined as described in Proposition 3.3; in addition, it shows how to count the number of graphs represented by a consistent summary.

LEMMA A.2. *For $S = \langle H, w, \mu \rangle$ a summary, S is consistent if and only if $w[h] \leq s[h]$ for each $h \in H$. Furthermore, when S is consistent, the following holds:*

$$|R_S| = \prod_{h \in H} \binom{s[h]}{w[h]}. \tag{2}$$

PROOF. Let $S = \langle H, w, \mu \rangle$ be a summary. Then, by Definition 3.2 of a graph represented by S and by Definition A.1 of μ^-, for each graph G, we have that $G \in R_S$ if and only if

(a) $G \subseteq \bigcup_{h \in H} \mu^-(h)$, and

(b) $w[h] = |\mu^-(h) \cap G|$ for each $h \in H$.

We next show that S is consistent if and only if $w[h] \leq s[h]$ for each $h \in H$.

($\Rightarrow$) Assume that S is consistent, and so $R_S \neq \emptyset$. Consider an arbitrary RDF graph $G \in R_S$ and an arbitrary triple $h \in H$. Then, $w[h]$ is the number of triples in $\mu^-(h)$ that occur in G. Since $s[h] = |\mu^-(h)|$, we have that $w[h] \leq s[h]$.

($\Leftarrow$) Assume that $w[h] \leq s[h]$ for each $h \in H$. Then, for each $h \in H$, let G_h be an arbitrary, but fixed subset of $\mu^-(h)$ of size $w[h]$; since $w[h] \leq s[h]$, such G_h exists. Finally, let $G = \bigcup_{h \in H} G_h$. By properties (a) and (b), we have that $G \in R_S$.

Therefore, S is consistent if and only if $w[h] \leq s[h]$ for each $h \in H$. To conclude the proof, we assume that S is consistent and show that Equation (2) holds. By properties (a) and (b), each graph $G \in R_S$ can be generated by independently choosing, for each triple $h \in H$, a set G_h of atoms of size $w[h]$ such that $G_h \subseteq \mu^-(h)$. For each $h \in H$, there are precisely

$$\binom{s[h]}{w[h]}$$

distinct such sets G_h. So, we can count the total number of RDF graphs in R_S as shown in Equation (2). □

B. PROOFS FOR SECTION 4

In this appendix, we prove Theorems 4.3 and 4.4, and Proposition 4.6. For the rest of this section, we fix an arbitrary consistent summary $S = \langle H, w, \mu \rangle$ and a CQ q such that $\mathsf{res}(q) \subseteq \mathsf{dom}(\mu)$.

B.1 Properties of the Partition Base

We start by formalising the properties of the unifiable partitions of q that were intuitively outlined in Section 4.1. To this end, we first define the notion of an expansion w.r.t. a partition P of an answer τ to $\mu(q)$ over H.

DEFINITION B.1. *Let τ be an answer to $\mu(q)$ on H and let $P \in B_\tau$ be a partition. A substitution π with $\mathsf{dom}(\pi) = \mathsf{var}(q)$ and $\mathsf{rng}(\pi) \subseteq \mathsf{dom}(\mu)$ is a* τ-expansion *to P if, for each $x \in \mathsf{var}(q)$,*

(1) $\pi(x) \in \mu^-(\tau(x))$,

(2) $\gamma_P(x) \in \mathsf{var}(q)$ implies that $\pi(x) = \pi(\gamma_P(x))$,

(3) $\gamma_P(x) \in \mathsf{res}(q)$ implies that $\pi(x) = \gamma_P(x)$.

Then, $\mathsf{exp}_\tau(P)$ contains each τ-expansion to P, and $\mathsf{exp}^{\prec}_\tau(P)$ contains each $\pi \in \mathsf{exp}_\tau(P)$ for which no $P' \in B_\tau$ exists such that $P \prec P'$ and $\pi \in \mathsf{exp}_\tau(P')$.

PROPOSITION B.2. *For each answer τ to $\mu(q)$ on H and each partition $P \in B_\tau$, the following properties hold:*

(P1) for each $\pi \in \mathsf{exp}_\tau(P)$, exactly one partition $P' \in B_\tau$ exists such that $P \preceq P'$ and $\pi \in \mathsf{exp}^{\prec}_\tau(P')$,

(P2) for each $\pi \in \mathsf{exp}^{\prec}_\tau(P)$ and all atoms $a, a' \in q$, there exists $u \in P$ with $a, a' \in u$ if and only if $\pi(a) = \pi(a')$.

PROOF. For the rest of this proof, we let τ be an arbitrary answer to $\mu(q)$ on H.

Property (P1) Since B_τ is finite and $\preceq$ is a partial order on B_τ, for each $P \in B_\tau$ and each $\pi \in \mathsf{exp}_\tau(P)$, there exists a partition $P' \in B_\tau$ such that $P \preceq P'$ and $\pi \in \mathsf{exp}_\tau^{\prec}(P')$. Hence, we next show that $\mathsf{exp}_\tau^{\prec}(P_1) \cap \mathsf{exp}_\tau^{\prec}(P_2) = \emptyset$ for all distinct partitions $P_1, P_2 \in B_\tau$.

Consider two arbitrary distinct partitions $P_1, P_2 \in B_\tau$. By the definition of $\mathsf{exp}_\tau^{\prec}(P_1)$ and $\mathsf{exp}_\tau^{\prec}(P_2)$, if $P_1 \prec P_2$ or $P_2 \prec P_1$, then $\mathsf{exp}_\tau^{\prec}(P_1) \cap \mathsf{exp}_\tau^{\prec}(P_2) = \emptyset$. So, in the rest of this proof, we consider the case in which $P_1 \not\prec P_2$ and $P_2 \not\prec P_1$.

Let $\approx$ be the smallest equivalence relation on (the atoms of) q such that $a \approx a'$ for all atoms $a, a' \in q$ such that $a, a' \in u$ for $u \in P_i$ with $i \in [1, 2]$. Then, let P be the set of all the equivalence classes of $\approx$. Since P_1 and P_2 are partitions of q and $\approx$ is an equivalence relation on the atoms of q, we have that P is a partition of q as well. By the definition of $\approx$, for each $i \in [1, 2]$ and each $u \in P_i$, there exists an equivalence class of $\approx$ that contains each atom in u; thus, we have that $P_i \preceq P$. By the initial assumptions, we have that $P_1 \neq P_2$, $P_1 \not\prec P_2$, and $P_2 \not\prec P_1$; therefore, we also have that $P_1 \prec P$ and $P_2 \prec P$. We next show that $P \in B_\tau$ and, for each $\pi \in \mathsf{exp}_\tau(P_1) \cap \mathsf{exp}_\tau(P_2)$, we have that $\pi \in \mathsf{exp}_\tau(P)$. Please note that this suffices to show that $\mathsf{exp}_\tau^{\prec}(P_1) \cap \mathsf{exp}_\tau^{\prec}(P_2) = \emptyset$ since $P_1 \prec P$ and $P_2 \prec P$.

Consider an arbitrary substitution π with $\mathsf{dom}(\pi) = \mathsf{var}(q)$ and $\mathsf{rng}(\pi) \subseteq \mathsf{dom}(\mu)$ such that $\pi \in \mathsf{exp}_\tau(P_1) \cap \mathsf{exp}_\tau(P_2)$. Since π is a τ-expansion to P_1 and P_2, for each $x \in \mathsf{var}(q)$, we have that $\tau(x) = \mu(\pi(x))$. Next, let $\mathcal{G}_P$ be the term reachability graph for P defined in Definition 4.1. To prove that π is a τ-expansion to P, we first show that, for all terms $s, t \in \mathsf{term}(q)$ such that s reaches t in $\mathcal{G}_P$, we have that $\pi(s) = \pi(t)$. To this end, we show that the following property holds for all atoms $a, a' \in q$:

(A1) $a \approx a'$ implies that $\pi(a) = \pi(a')$.

To prove auxiliary property (A1), we proceed by induction on the number of steps required to derive $a \approx a'$. For the base case, the empty relation $\approx_0$ clearly satisfies the property. For the inductive step, consider an arbitrary relation $\approx_n$ obtained in n steps that satisfies the property; we show that the same holds for all the equivalences derivable from $\approx_n$. Since the equality relation $=$ is reflexive, symmetric, and transitive, the derivation of $a \approx_{n+1} a'$ due to reflexivity, symmetry, or transitivity clearly preserves the required property. So, we next consider an arbitrary $i \in [1, 2]$, a set $u \in P_i$, and two atoms $a, a' \in u$, so that we derive $a \approx_{n+1} a'$. Let $a = \langle s, p, o \rangle$ and let $a' = \langle s', p', o' \rangle$. Then, by the definition of $\mathcal{G}_{P_i}$, terms s, p, and o reach s', p', and o' in $\mathcal{G}_{P_i}$, respectively. Thus, by the definition of γ_{P_i}, we also have that $\gamma_{P_i}(s) = \gamma_{P_i}(s')$, $\gamma_{P_i}(p) = \gamma_{P_i}(p')$, and $\gamma_{P_i}(o) = \gamma_{P_i}(o')$. By properties (2) and (3) in Definition B.1, we have that $\pi(a) = \pi(a')$, as required.

Please note that $\mathcal{G}_P$ is obtained by adding an undirected edge between s_1 and s_2, p_1 and p_2, and o_1 and o_2 for each $u \in P$ and all atoms $\langle s_1, p_1, o_1 \rangle, \langle s_2, p_2, o_2 \rangle \in u$. By the definition of P, we have that $\langle s_1, p_1, o_1 \rangle \approx \langle s_2, p_2, o_2 \rangle$ for each such pair of atoms. So, because the equality relation $=$ is reflexive, symmetric, and transitive, property (A1) implies that $\pi(s) = \pi(t)$ for all terms that are reachable in $\mathcal{G}_P$.

Since $\pi(s) = \pi(t)$ for all $s, t \in \mathsf{term}(q)$ such that s reaches t in $\mathcal{G}_P$, partition P is unifiable, and so $P \in B$. We next show that P is satisfied by τ and that π satisfies conditions (2) and (3) in Definition B.1. For $x \in \mathsf{var}(q)$ a variable, we next consider two cases.

- Assume that $\gamma_P(x)$ is a variable $y \in \mathsf{var}(q)$. By the definition of γ_P, we have that x reaches y in $\mathcal{G}_P$, and so $\pi(x) = \pi(y)$. Since $\tau(x) = \mu(\pi(x))$ and $\tau(y) = \mu(\pi(y))$, we conclude that $\tau(x) = \tau(y)$.
- Assume that $\gamma_P(x)$ is a resource $d \in \mathsf{res}(q)$. By the definition of γ_P, we have that x reaches d in $\mathcal{G}_P$, and so $\pi(x) = d$. Since $\tau(x) = \mu(\pi(x))$, we conclude that $\tau(x) = \mu(d)$.

Therefore, P is satisfied by τ and π satisfies conditions (2) and (3) in Definition B.1; thus, π is a τ-expansion to P, as required.

Property (P2) Let $P \in B_\tau$ be a partition and let $\pi \in \mathsf{exp}_\tau^{\prec}(P)$ be a τ-expansion; we show that, for all $a, a' \in q$, there exists $u \in P$ with $a, a' \in u$ if and only if $\pi(a) = \pi(a')$.

($\Rightarrow$) By properties (2) and (3) in Definition B.1 and by Definition 4.1 of γ_P, we have that $\pi(a) = \pi(a')$ for each $u \in P$ and all $a, a' \in u$, as required.

($\Leftarrow$) For the sake of a contradiction, assume that two atoms $a, a' \in q$ exist such that $\pi(a) = \pi(a')$, but no $u \in P$ exists such that $a, a' \in u$. Then, let P' be the partition of q such that, for all $a_1, a_2 \in q$, there exists $u' \in P'$ such that $a_1, a_2 \in u'$ if and only if $\pi(a_1) = \pi(a_2)$. Please note that, by the definition of P', a set $u' \in P'$ exists such that $a, a' \in u'$.

We first show that $P \prec P'$. To this end, consider an arbitrary $u \in P$ and two arbitrary atoms $a_1, a_2 \in u$. Because π is a τ-expansion to P, we have that $\pi(a) = \pi(a')$, and so a set $u' \in P'$ exists such that $a_1, a_2 \in u'$. Thus, we have that $P \preceq P'$. Furthermore, by the initial assumption, there exists no $u \in P$ such that $a, a' \in u$; hence, we also have that $P \prec P'$.

By the construction of P', we have that $\pi(s) = \pi(t)$ for all terms $s, t \in \mathsf{term}(q)$ such that s reaches t in $\mathcal{G}_{P'}$. So, P' is a unifiable partition of q and satisfies properties (2) and (3) of Definition B.1. Since π is a τ-expansion to P, we have that $\tau(x) = \mu(\pi(x))$ for each $x \in \mathsf{var}(q)$; thus, P' is satisfied by τ and π is a τ-expansion to P'. This is a contradiction, since $P \prec P'$ and $\pi \in \mathsf{exp}_\tau^{\prec}(P)$. $\square$

B.2 Counting Expansions

We next show how one can effectively count, given an answer τ to $\mu(q)$ over H and a partition $P \in B_\tau$, the number of τ-expansions to P. To this end, we first an auxiliary lemma.

LEMMA B.3. *For all partitions $P, P' \in B$ with $P \prec P'$,*

$$-K(P, P') \;=\; \sum_{P \prec P'' \preceq P'} K(P'', P'). \tag{3}$$

PROOF. Let P and P' be as specified in the lemma. By Definition 4.1, we have that

$$-K(P,P') \;=\; -(C^e(P,P') - C^o(P,P')) \;=\; C^o(P,P') - C^e(P,P').$$

Because P and P' are distinct partitions, by Definition 4.1 of a chain from P to P', the number of even chains from P to P' in B can be computed by summing, for each $P'' \in B$ with $P \prec P''$ and $P'' \preceq P'$, the number of odd chains from P'' to P' in B. Similarly, the number of odd chains from P to P' in B can be computed by summing, for each $P'' \in B$ with $P \prec P''$ and $P'' \preceq P'$, the number of even chains from P'' to P' in B. Hence, we have that

$$\begin{aligned} C^e(P,P') &= \sum_{P \prec P'' \preceq P'} C^o(P'',P'), \text{ and} \\ C^o(P,P') &= \sum_{P \prec P'' \preceq P'} C^e(P'',P'). \end{aligned}$$

Therefore, we have that

$$-K(P,P') = C^o(P,P') - C^e(P,P') = \left(\sum_{P \prec P'' \preceq P'} C^e(P'',P')\right) - \left(\sum_{P \prec P'' \preceq P'} C^o(P'',P')\right)$$

$$= \sum_{P \prec P'' \preceq P'} \big(C^e(P'',P') - C^o(P'',P')\big) = \sum_{P \prec P'' \preceq P'} K(P'',P').$$

□

PROPOSITION B.4. *The following equations hold for each answer τ to $\mu(q)$ on $\mathsf{res}(H)$ and each $P \in B_\tau$:*

$$|\mathsf{exp}_\tau(P)| \;=\; N_\tau(P), \tag{4}$$

$$|\mathsf{exp}_\tau^{\prec}(P)| \;=\; \sum_{P' \in B_\tau, P \preceq P'} K(P,P') \cdot N_\tau(P'). \tag{5}$$

PROOF. Let τ be as specified in the lemma, and let $\preceq_\tau$ be the restriction of $\preceq$ to B_τ.

We first prove Equation (4). To this end, consider a partition $P \in B_\tau$. By Definition B.1 of a τ-expansion to P, each $\pi \in \mathsf{exp}_\tau(P)$ can vary only on the variables occurring in the range of γ_P. Furthermore, for each $x \in \mathsf{varrng}(\gamma_P)$, we have that $\pi(x) \in \mu^-(\tau(x))$ and $|\mu^-(\tau(x))| = s[\tau(x)]$; thus, there are $N_\tau(P) = \prod_{x \in \mathsf{varrng}(\gamma_P)} s[\tau(x)]$ distinct τ-expansions to P.

We next prove Equation (5) by induction on $\preceq_\tau$.

Base case. Consider an arbitrary partition $P \in B_\tau$ that is $\preceq_\tau$-maximal. We then have that $\mathsf{exp}_\tau^{\prec}(P) = \mathsf{exp}_\tau(P)$; furthermore, by Equation (4), we have that $|\mathsf{exp}_\tau(P)| = N_\tau(P)$. In addition, we have that $\sum_{P \preceq_\tau P'} K(P,P') \cdot N_\tau(P') = K(P,P) \cdot N_\tau(P)$. Finally, by Definition 4.1 of K, we have that $K(P,P) = 1$; so, Equation (5) holds.

Inductive Step. Consider an arbitrary $P \in B_\tau$ and assume that, for each $P'' \in B_\tau$ with $P \prec_\tau P''$, we have that

$$|\mathsf{exp}_\tau^{\prec}(P'')| \;=\; \sum_{P'' \preceq_\tau P'} K(P'',P') \cdot N_\tau(P');$$

we show that the same holds for P. By the definition of $\mathsf{exp}_\tau^{\prec}(P)$, we have that $\mathsf{exp}_\tau^{\prec}(P) = \mathsf{exp}_\tau(P) \setminus \bigcup_{P \prec_\tau P''} \mathsf{exp}_\tau(P'')$. By Proposition B.2, for each $\pi \in \bigcup_{P \prec_\tau P''} \mathsf{exp}_\tau(P'')$, exactly one partition $P'' \in B_\tau$ exists such that $P \prec_\tau P''$ and $\pi \in \mathsf{exp}_\tau^{\prec}(P'')$. Therefore, we have that

$$\begin{aligned} \mathsf{exp}_\tau^{\prec}(P) &= \mathsf{exp}_\tau(P) \setminus \bigcup_{P \prec_\tau P''} \mathsf{exp}_\tau^{\prec}(P''), \text{ and} \\ |\mathsf{exp}_\tau^{\prec}(P)| &= |\mathsf{exp}_\tau(P)| - \sum_{P \prec_\tau P''} |\mathsf{exp}_\tau^{\prec}(P'')|. \end{aligned}$$

By Equation (4), we have that $|\mathsf{exp}_\tau(P)| = N_\tau(P)$. So, by the inductive hypothesis, we can compute $|\mathsf{exp}_\tau^{\prec}(P)|$ as follows.

$$|\mathsf{exp}_\tau^{\prec}(P)| = N_\tau(P) - \sum_{P \prec_\tau P''} \left(\sum_{P'' \preceq_\tau P'} K(P'',P') \cdot N_\tau(P')\right) = N_\tau(P) - \sum_{P \prec_\tau P'} N_\tau(P') \cdot \left(\sum_{P \prec_\tau P'' \preceq_\tau P'} K(P'',P')\right).$$

By Lemma B.3, for each $P' \in B_\tau$ with $P \prec_\tau P'$, we have that $\sum_{P \prec_\tau P'' \preceq_\tau P'} K(P'', P') = -K(P, P')$; thus, we have that

$$|\mathsf{exp}_\tau^\prec(P)| = N_\tau(P) - \sum_{P \prec_\tau P'} N_\tau(P') \cdot \left(\sum_{P \prec_\tau P'' \preceq_\tau P'} K(P'', P') \right)$$

$$= N_\tau(P) - \left(\sum_{P \prec_\tau P'} -N_\tau(P') \cdot K(P, P') \right) = N_\tau(P) + \left(\sum_{P \prec_\tau P'} N_\tau(P') \cdot K(P, P') \right).$$

Finally, by Definition 4.1 of K, we have that $K(P, P)$ is 1, so the following holds:

$$|\mathsf{exp}_\tau^\prec(P)| = N_\tau(P) + \left(\sum_{P \prec_\tau P'} N_\tau(P') \cdot K(P, P') \right)$$

$$= K(P, P) \cdot N_\tau(P) + \left(\sum_{P \prec_\tau P'} K(P, P') \cdot N_\tau(P') \right) = \sum_{P \preceq_\tau P'} K(P, P') \cdot N_\tau(P').$$

□

B.3 Proofs of Theorem 4.3 and Proposition 4.6

We are now ready to prove the correctness of the formulae for computing $E_{q,S}$ given in Theorem 4.3 and Proposition 4.6.

LEMMA B.5. *For all $n, k, m \in \mathbb{N}$ with $n \geq m \geq k > 0$,*

$$\frac{\binom{n-k}{m-k}}{\binom{n}{m}} = \frac{(m)_k}{(n)_k}.$$

PROOF. Let n, k, m be natural numbers as specified in the lemma. By the definition of binomial coefficient, we have that

$$\binom{n}{m} = \frac{n!}{m! \cdot (n-m)!}, \text{ and}$$

$$\binom{n-k}{m-k} = \frac{(n-k)!}{(m-k)! \cdot (n-k-(m-k))!} = \frac{(n-k)!}{(m-k)! \cdot (n-m)!}.$$

By using the standard laws of arithmetics, we then have that

$$\frac{\binom{n-k}{m-k}}{\binom{n}{m}} = \frac{\frac{(n-k)!}{(m-k)! \cdot (n-m)!}}{\frac{n!}{m! \cdot (n-m)!}} = \frac{(n-k)! \cdot m!}{n! \cdot (m-k)!} = \frac{m \cdots (m-k+1)}{n \cdots (n-k+1)} = \frac{(m)_k}{(n)_k}.$$

□

LEMMA B.6. *Let τ be an answer to $\mu(q)$ over H and let $P \in B_\tau$ be a partition. The following holds for each $\pi \in \mathsf{exp}_\tau^\prec(P)$:*

$$F_\tau(P) = E_{\pi(q),S}. \tag{6}$$

PROOF. Let τ and P be as specified in the lemma. Consider an arbitrary substitution $\pi \in \mathsf{exp}_\tau^\prec(P)$ and let $q_\pi = \pi(q)$; we show that $F_\tau(P) = E_{q_\pi,S}$. Note that, for each $a \in q_\pi$, we have that $\mathsf{var}(a) = \emptyset$ and $\mathsf{res}(a) \subseteq \mathsf{dom}(\mu)$.

By the definition of τ-expansion, we have that $\tau(x) = \mu(\pi(x))$ for each $x \in \mathsf{var}(q)$, and so $\mu(q_\pi) = \tau(\mu(q))$. Because P is satisfied by τ, for each $u \in P$ and all atoms $a, a' \in u$, we have that $\tau(\mu(a)) = \tau(\mu(a'))$; so, $\tau(\mu(u))$ is a set consisting of a single triple from H. In the following, for each triple $h \in \tau(\mu(q))$, let $\#_h \in \mathbb{N}$ be as specified in Definition 4.2.

By property (P2) of Proposition B.2, for all atoms $a, a' \in q$, we have that $\pi(a) = \pi(a')$ if and only if there exists $u \in P$ with $a, a' \in u$. That is, there is a one-to-one correspondence between the triples in q_π and the sets in P. So, for each triple $h \in \tau(\mu(q))$, we have that

$$\#_h \;=\; |\{u \in P \mid \tau(\mu(u)) = \{h\}\}| \;=\; |\{f \in q_\pi \mid \mu(f) = h\}|. \tag{7}$$

By Definition 3.4 of expectation, we have that

$$E_{q_\pi,S} \;= \sum_{G \in R_S} \frac{|[\![q_\pi]\!]_G|}{|R_S|}. \tag{8}$$

By the initial assumption, S is consistent; so, $|R_S| > 0$ and $E_{q_\pi,S}$ is well-defined. Next, let $\mathcal{N} = \sum_{G \in R_S} |[\![q_\pi]\!]_G|$. Since q_π is a variable-free, $|[\![q_\pi]\!]_G|$ is either 0 or 1; thus, $\mathcal{N}$ is the number of graphs $G \in R_S$ with $q_\pi \subseteq G$. By Definition 3.2 of a graph represented by S, for each graph G, we have that $G \in R_S$ and $q_\pi \subseteq G$ if and only if

(A) $G \subseteq \bigcup_{h \in H} \mu^-(h)$,

(B) for each $h \in H$, the set $\mu^-(h) \cap G$ contains each $f \in q_\pi$ with $\mu(f) = h$, and

(C) $w[h] = |\mu^-(h) \cap G|$ for each $h \in H$.

To prove the lemma, we next consider two alternative cases.

(Case 1) Assume that a triple $h \in \tau(\mu(q))$ exists such that $|\#_h| > w[h]$; so $F_\tau(P) = 0$. By Equation (7), $\#_h$ is the number of triples in q_π that μ maps onto h. So, there is no graph G that satisfies both properties (B) and (C) above, and thus $\mathcal{N} = 0$ and $E_{q_\pi,S} = 0$. Therefore, $F_\tau(P) = E_{q_\pi,S}$ and the lemma holds.

(Case 2) Assume that $\#_h \leq w[h]$ for each $h \in \tau(\mu(q))$; so

$$F_\tau(P) = \prod_{h \in \tau(\mu(q))} \frac{(w[h])_{\#_h}}{(s[h])_{\#_h}}.$$

Each graph $G \in R_S$ with $q_\pi \subseteq G$ can be generated by independently choosing, for each $h \in H$, a subset G_h of $\mu^-(h)$ such that $|G_h| = w[h]$ and G_h contains each $f \in q_\pi$ with $\mu(f) = h$. Please recall that $s[h]$ is the size of $\mu^-(h)$. Next, we show how to compute, for each $h \in H$, the number $\mathcal{N}_h$ of distinct such sets by considering two alternative cases.

- Assume that $h \in \tau(\mu(q))$. By Equation (7), there are precisely $\#_h$ distinct triples $f \in q_\pi$ such that $\mu(f) = h$, and so

$$\mathcal{N}_h = \binom{s[h] - \#_h}{w[h] - \#_h}.$$

- Assume that $h \notin \tau(\mu(q))$. Hence, no triple $f \in q_\pi$ exists such that $\mu(f) = h$. So, we have that

$$\mathcal{N}_h = \binom{s[h]}{w[h]}.$$

Therefore, we can compute $\mathcal{N}$ as follows:

$$\mathcal{N} \;=\; \sum_{G \in R_S} |[\![q_\pi]\!]_G| \;=\; \prod_{h \in H} \mathcal{N}_h \;=\; \prod_{h \in \tau(\mu(q))} \binom{s[h] - \#_h}{w[h] - \#_h} \;\cdot\; \prod_{h \in (H \setminus \tau(\mu(q)))} \binom{s[h]}{w[h]}.$$

By the definition of expected answer, we then have that

$$E_{q_\pi,S} \;=\; \sum_{G \in R_S} \frac{|[\![q_\pi]\!]_G|}{|R_S|} \;=\; \frac{\mathcal{N}}{|R_S|} \;=\; \frac{\prod_{h \in \tau(\mu(q))} \binom{s[h]-\#_h}{w[h]-\#_h} \cdot \prod_{h \in (H \setminus \tau(\mu(q)))} \binom{s[h]}{w[h]}}{|R_S|}.$$

By using the formula for computing $|R_S|$ from Lemma A.2, we obtain the following:

$$E_{q_\pi,S} \;=\; \frac{\prod_{h \in \tau(\mu(q))} \binom{s[h]-\#_h}{w[h]-\#_h} \cdot \prod_{h \in (H \setminus \tau(\mu(q)))} \binom{s[h]}{w[h]}}{|R_S|} \;=\; \frac{\prod_{h \in \tau(\mu(q))} \binom{s[h]-\#_h}{w[h]-\#_h} \cdot \prod_{h \in (H \setminus \tau(\mu(q)))} \binom{s[h]}{w[h]}}{\prod_{h \in H} \binom{s[h]}{w[h]}}$$

$$= \frac{\prod_{h \in \tau(\mu(q))} \binom{s[h]-\#_h}{w[h]-\#_h} \cdot \prod_{h \in (H \setminus \tau(\mu(q)))} \binom{s[h]}{w[h]}}{\prod_{h \in \tau(\mu(q))} \binom{s[h]}{w[h]} \cdot \prod_{h \in (H \setminus \tau(\mu(q)))} \binom{s[h]}{w[h]}} \;=\; \prod_{h \in \tau(\mu(q))} \frac{\binom{s[h]-\#_h}{w[h]-\#_h}}{\binom{s[h]}{w[h]}}.$$

Finally, by Lemma B.5, we have that

$$E_{q_\pi,S} \;=\; \prod_{h \in \tau(\mu(q))} \frac{\binom{s[h]-\#_h}{w[h]-\#_h}}{\binom{s[h]}{w[h]}} \;=\; \prod_{h \in \tau(\mu(q))} \frac{(w[h])_{\#_h}}{(s[h])_{\#_h}} \;=\; F_\tau(P).$$

□

We are now ready to prove Theorem 4.3.

THEOREM 4.3. *The following identity holds:*

$$E_{q,S} = \sum_{\tau \in [\![\mu(q)]\!]_H} \sum_{P \in B_\tau} F_\tau(P) \cdot \sum_{\substack{P' \in B_\tau \\ P \preceq P'}} K(P, P') \cdot N_\tau(P').$$

PROOF. In the rest of this proof, for each substitution π with $\mathsf{dom}(\pi) = \mathsf{var}(q)$ and $\mathsf{rng}(\pi) \subseteq \mathsf{dom}(\mu)$, we let $q_\pi = \pi(q)$; furthermore, let $P_0 = \{\{a\} \mid a \in q\}$ be a partition of q. By Definition 4.1 of partition base, we have that $P_0 \in B$; in addition, for each answer τ $\mu(q)$ on H, we have that $P_0 \in B_\tau$; finally, for each $P \in B$, we have that $P_0 \preceq P$.

Consider an arbitrary $G \in R_S$; because S is consistent, such G exists. Then, $|[\![q]\!]_G|$ is the number of substitutions π with $\mathsf{dom}(\pi) = \mathsf{var}(q)$ and $\mathsf{rng}(\pi) \subseteq \mathsf{dom}(\mu)$ such that $q_\pi \subseteq G$. Since $G \in R_S$, for each such π, the substitution τ such that

$\tau(x) = \mu(\pi(x))$, for each $x \in \mathsf{dom}(\pi)$, is an answer to $\mu(q)$ on H. In addition, we have that $\pi \in \mathsf{exp}_\tau(P_0)$. So, by Definition 3.4 of expected number of answers, the following holds:

$$E_{q,S} = \sum_{G \in R_S} \frac{|[\![q]\!]_G|}{|R_S|} = \sum_{G \in R_S} \sum_{\tau \in [\![\mu(q)]\!]_H} \sum_{\pi \in \mathsf{exp}_\tau(P_0)} \frac{|[\![q_\pi]\!]_G|}{|R_S|}.$$

By Proposition B.2, we have that $\mathsf{exp}_\tau(P_0) = \bigcup_{P \in B_\tau, P_0 \preceq P} \mathsf{exp}_\tau^{\prec}(P)$. Then, because $P_0 \preceq P$ for each $P \in B_\tau$, we have that

$$E_{q,S} = \sum_{G \in R_S} \sum_{\tau \in [\![\mu(q)]\!]_H} \sum_{\pi \in \mathsf{exp}_\tau(P_0)} \frac{|[\![q_\pi]\!]_G|}{|R_S|} = \sum_{G \in R_S} \sum_{\tau \in [\![\mu(q)]\!]_H} \sum_{P \in B_\tau} \sum_{\pi \in \mathsf{exp}_\tau^{\prec}(P)} \frac{|[\![q_\pi]\!]_G|}{|R_S|}.$$

So, by the definition of $E_{q_\pi,S}$, we can compute $E_{q,S}$ as follows:

$$E_{q,S} = \sum_{\tau \in [\![\mu(q)]\!]_H} \sum_{P \in B_\tau} \sum_{\pi \in \mathsf{exp}_\tau^{\prec}(P)} \sum_{G \in R_S} \frac{|[\![q_\pi]\!]_G|}{|R_S|} = \sum_{\tau \in [\![\mu(q)]\!]_H} \sum_{P \in B_\tau} \sum_{\pi \in \mathsf{exp}_\tau^{\prec}(P)} E_{q_\pi,S}.$$

By Lemma B.6, we then have that

$$E_{q,S} = \sum_{\tau \in [\![\mu(q)]\!]_H} \sum_{P \in B_\tau} \sum_{\pi \in \mathsf{exp}_\tau^{\prec}(P)} F_\tau(P).$$

Finally, Proposition B.4 shows how to count the number of $\preceq$-maximal τ-expansions, so we can rewrite $E_{q,S}$ as follows:

$$E_{q,S} = \sum_{\tau \in [\![\mu(q)]\!]_H} \sum_{P \in B_\tau} F_\tau(P) \cdot \sum_{P' \in B_\tau, P \preceq P'} K(P, P') \cdot N_\tau(P').$$

□

As pointed out in Section 4.2, Proposition 4.6 follows immediately from Theorem 4.3.

PROPOSITION 4.6. *If query q is μ-unification-free, then formula* (1) *correctly computes $E_{q,S}$.*

B.4 Proof of Theorem 4.4

We next show that our method for computing $\sigma_{q,S}$ is correct.

THEOREM 4.4. *Let q' be the query obtained from q by replacing each variable in $\mathsf{var}(q)$ with a fresh variable; then,*

$$\sigma^2_{q,S} = E_{q \cup q',S} - E^2_{q,S}.$$

PROOF. Let q' be as specified in the theorem and let ρ be the unique substitution with $\mathsf{dom}(\rho) = \mathsf{var}(q)$ and $\mathsf{rng}(\rho) = \mathsf{var}(q')$ such that $q' = \rho(q)$.

We start by showing that Equation (9) holds for each RDF graph $G \in R_S$.

$$|[\![q]\!]_G|^2 = |[\![q \cup q']\!]_G| \tag{9}$$

To this end, consider an arbitrary graph $G \in R_S$. Then, $|[\![q]\!]_G|^2$ is the cardinality of the set *Pairs* that contains each pair $\langle \pi, \pi' \rangle$ of substitutions mapping the variables of q to $\mathsf{dom}(\mu)$ such that $\{\pi(q), \pi'(q)\} \subseteq G$. But then, Equation (9) holds because the set of answers to $q \cup q'$ over G contains precisely, for each pair $\langle \pi, \pi' \rangle \in Pairs$, one distinct substitution $\kappa_{\pi,\pi'}$ from $\mathsf{var}(q) \cup \mathsf{var}(q')$ to $\mathsf{dom}(\mu)$. Such $\kappa_{\pi,\pi'}$ can be obtained from π and π' by setting, for each $x \in \mathsf{var}(q)$,

$$\kappa_{\pi,\pi'}(x) = \pi(x) \text{ and } \kappa_{\pi,\pi'}(\rho(x)) = \pi'(x).$$

We are now ready to prove the theorem. By Definition 3.4 of variance, we have that

$$\sigma^2_{q,S} = \sum_{G \in R_S} \frac{|[\![q]\!]_G|^2 - 2 \cdot |[\![q]\!]_G| \cdot E_{q,S} + E_{q,S})^2}{|R_S|}.$$

By using Equation (9) and by manipulating summations in the usual way, we can express $\sigma^2_{q,S}$ as follows.

$$\sum_{G \in R_S} \left[\frac{|[\![q \cup q']\!]_G|}{|R_S|} \right] - 2 \cdot E_{q,S} \cdot \sum_{G \in R_S} \left[\frac{|[\![q]\!]_G|}{|R_S|} \right] + E^2_{q,S}$$

By Definition 3.4 of expectation, we then have that

$$\sigma^2_{q,S} = E_{q \cup q',S} - 2 \cdot E^2_{q,S} + E^2_{q,S} = E_{q \cup q',S} - (E_{q,S})^2,$$

and so the theorem holds. □

B.5 Proof of Theorem 4.7

LEMMA B.7. *For any chain $P_0, \dots, P_\ell$ of partitions of a CQ q it holds that $n < |q|$.*

PROOF. This statement follows from the triples that $P \prec P'$ implies $|P| > |P'|$ and that the maximal size of a partition of q is $|q|$. □

We first recall definitions of the counting complexity classes, which we use in the proof.

DEFINITION B.8.

- *The class* #P *consists of all the functions f for each of which there exists an* NP *machine such that, for all input words x, $f(x)$ is the number of accepting computations (written in binary).*
- *The class* GapP *consists of all the functions f for each of which there exists an* NP *machine such that, for all input words x, $f(x)$ is the difference of the number of accepting computations and the number of rejecting computations.*

THEOREM 4.7. *The problem of deciding, given S, q, and integers m and n, whether $E_{q,S} = \frac{m}{n}$ holds is in $\mathsf{P}^{\#\mathsf{P}[1]}$, and is at least as hard as the problem of checking whether the number of answers to q on a graph is equal to a given integer.*

PROOF. The lower bound follows from the triple that the problem of checking whether $|[\![q]\!]_G| = k$ is, essentially, a subproblem of ESTIMATION. Indeed, the reduction is trivial: for a graph G and number k we just need to take $m = k$, $n = 1$, and a summary with a summarisation graph H isomorphic to G, the corresponding bijective function μ and weight function w such that $w(h) = 1$ for each $h \in H$.

The rest of the proof is devoted to showing the upper bound. It is well-known that $\mathsf{P}^{\#\mathsf{P}[1]} = \mathsf{P}^{\mathsf{GapP}[1]}$, we will next show that ESTIMATION is in $\mathsf{P}^{\mathsf{GapP}[1]}$.

Let $S = \langle H, w, \mu \rangle$ be a summary. Without loss of generality, we assume that it is consistent (this can be easily checked in polynomial time). Consider the number

$$n' = \prod_{h \in H} s[h]!.$$

On the one hand, n' is a common divisor of all non-zero $F_\tau(P)$ appearing in the formula for computing $E_{q,S}$. On the other, n' can be computed by a polynomially bounded deterministic Turing machine, because there are polynomial number of atoms h in H and each $s[h]$ is polynomially bounded by the size of $\mathsf{dom}(\mu)$. Therefore, our $\mathsf{P}^{\mathsf{GapP}[1]}$ algorithm will work as follows:

1. in the main deterministic polynomial Turing machine compute n';
2. call a GapP oracle to compute

$$m' = \sum_{\tau \in [\![\mu(q)]\!]_H} \sum_{P \in B'_\tau} \sum_{P' \in B_\tau, P \preceq P'} \left(\prod_{h \in \tau(\mu(q))} (w[h])_{\#_h} \right) \cdot \frac{n'}{\prod_{h \in \tau(\mu(q))} (s[h])_{\#_h}} \cdot K(P, P') \cdot N_\tau(P'),$$

   for B'_τ a subset of B_τ that consists of all P with $\#_h \leq w[h]$ for all $h \in \tau(\mu(q))$;
3. return `true` if and only if $m' \cdot n = m \cdot n'$.

Note that the output of a GapP oracle is a number, written in binary, exponentially bounded by the size of the input, so the multiplications and comparison can indeed be done by a deterministic polynomial Turing machine. Hence, we are left to show that m' can be computed by a GapP oracle.

To this end, consider a nondeterministic Turing machine that, on input S and q, proceeds as follows:

1. guess a substitution τ from $\mathsf{var}(q)$ to $\mathsf{res}(H)$ and partitions $P, P_1, \dots, P_{\ell-1}, P'$ of q, with $\ell < |q|$;
2. check that $\tau \in [\![\mu(q)]\!]_H$, $P \in B'_\tau$, $P' \in B_\tau$, and $P \prec P_1 \prec \dots \prec P_{\ell-1} \prec P'$; if at least one of these conditions does not hold, fork into two computations, one of which immediately accepts, and one rejects (i.e., the overall contribution to the difference is 0);
3. for every $h \in \tau(\mu(q))$ and every $i \in [0, \#_h]$ fork into $w[h] - i$ computations; the forks are done consecutively, so overall there are $\prod_{h \in \tau(\mu(q))} (w[h])_{\#_h}$ computations for every τ and $P, P_1, \dots, P_{\ell-1}, P'$; note that the number $w[h]$ is polynomially bounded by the size of the input, so overall there are polynomial number of forks to polynomial number of computations on each fork;
4. similarly, by means of a polynomial number of forks, split each computation into $n' \Big/ \prod_{h \in \tau(\mu(q))} (s[h])_{\#_h}$ ones;
5. similarly, split each computation into $N_\tau(P) = \prod_{x \in \mathsf{varrng}(\gamma_P)} s[\tau(x)]$ ones;
6. accept if ℓ is even and reject if it is odd.

On the one hand, this Turing machine indeed works in polynomial time. On the other, it the difference of its accepting and rejecting computations is precisely m': by Lemma B.7, after step 2 it has the same number of active computations as the number of summands in the formula for m' (and other computations "neutralise" each other), then the further forks produce the required number of computations for each τ and $P, P_1, \dots, P_{\ell-1}, P'$, and, finally, computations with even lengths of chains accept and others reject, producing the required difference for the overall GapP function. □

C. PROOFS FOR SECTION 5

In this appendix, we fix a consistent summary $S = \langle H, w, \mu \rangle$, a μ-unification-free CQ q with $\mathsf{res}(q) \subseteq \mathsf{dom}(\mu)$, and a HT decomposition $\mathcal{D} = \langle \mathcal{T}, X, p \rangle$ for q. Next, for each $v \in \mathsf{vert}(\mathcal{T})$, let $\mathsf{vis}(v) = \emptyset$ if v is the root of $\mathcal{T}$; otherwise, $\mathsf{vis}(v) = X^v \cap X^{v'}$ for v' the parent of v in $\mathcal{T}$. Please note that, for each $v \in \mathsf{vert}(\mathcal{T})$, the set Z computed in lines 5 and 6 of Algorithm 1 is $\mathsf{vis}(v)$. Then, a *partial estimation* over a set of variables X_i is a multiset of pairs $\langle \tau, e \rangle$ that consists of a substitution τ with $\mathsf{dom}(\tau) = X_i$ and $\mathsf{rng}(\tau) \subseteq \mathsf{res}(H)$, and of a rational number e. Please note that Algorithm 1 associates each vertex v of $\mathcal{T}$ with partial estimations M^v and E^v over variables X^v and $\mathsf{vis}(v)$, respectively.

C.1 Properties of Operators $\bowtie$ and $\mathtt{Pr}$

Towards showing the correctness of Algorithm 1, we next prove important properties of the operations from Definition 5.2.

PROPOSITION C.1. *The operator $\bowtie$ on partial estimations is commutative and associative. Furthermore, the following identities hold for all partial estimations L_1 and L_2 over variables X_1 and X_2, respectively, and each set of variables $X' \subseteq X_1$:*

(a) $\mathtt{Grp}(\mathtt{Pr}_{X',S}(L_1)) = \mathtt{Grp}(\mathtt{Pr}_{X',S}(\mathtt{Grp}(L_1)))$;

(b) $\mathtt{Grp}(L_1 \bowtie L_2) = \mathtt{Grp}(\mathtt{Grp}(L_1) \bowtie L_2)$;

(c) $\mathtt{Pr}_{X_1,S}(L_1 \bowtie L_2) = L_1 \bowtie \mathtt{Pr}_{X_1 \cap X_2,S}(L_2)$.

PROOF. Since multiplication of rational numbers and union of substitutions are commutative and associative operations, we immediately have that the operator $\bowtie$ on partial estimations is commutative and associative too. Hence, in the rest of this proof, we show that properties (a)–(c) hold.

We start by proving property (a). Consider an arbitrary substitution τ occurring in a pair in L_1. By the definition of $\mathtt{Grp}(\mathtt{Pr}_{X',S}(L_1))$, there exists a unique occurrence of a tuple $\langle \tau, e \rangle$ in $\mathtt{Grp}(\mathtt{Pr}_{X',S}(L_1))$, where e is functionally determined by τ. Similarly, by the definition of $\mathtt{Grp}(\mathtt{Pr}_{X',S}(\mathtt{Grp}(L_1)))$, there exists a unique occurrence of a tuple $\langle \tau, \bar{e} \rangle$ in $\mathtt{Grp}(\mathtt{Pr}_{X',S}(\mathtt{Grp}(L_1)))$, where $\bar{e}$ is also functionally determined by τ. We next show that $e = \bar{e}$.

Let $\langle \tau_1, e_1 \rangle, \ldots, \langle \tau_n, e_n \rangle$ be an enumeration of the multiset that contains each distinct occurrence of $\langle \tau_i, e_i \rangle$ in L_1 such that $\tau(x) = \tau_i(x)$ for each variable $x \in X'$. By the definition of $\mathtt{Grp}(\mathtt{Pr}_{X',S}(L_1))$, we have that

$$e \;=\; \sum_{1 \leq i \leq n} \left(\prod_{x \in \mathsf{dom}(\tau_i) \setminus X'} s[\tau_i(x)] \right) \cdot e_i.$$

Let I be the set that, for each $j \in [1, n]$, contains the least index $i \in [1, j]$ such that $\tau_i = \tau_j$. Furthermore, for each $i \in I$, let K_i contain each index $k \in [i, n]$ such that $\tau_k = \tau_i$. Then, we have that

$$e \;=\; \sum_{i \in I} \sum_{k \in K_i} \left(\prod_{x \in \mathsf{dom}(\tau_k) \setminus X'} s[\tau_k(x)] \right) \cdot e_k.$$

Please note that, for each $i \in I$ and each $k \in K_i$, we have that

$$\prod_{x \in \mathsf{dom}(\tau_i) \setminus X'} s[\tau_i(x)] \;=\; \prod_{x \in \mathsf{varrng}(\tau_k) \setminus X'} s[\tau_k(x)].$$

Therefore, we have that

$$e \;=\; \sum_{i \in I} \left(\left(\prod_{x \in \mathsf{dom}(\tau_i) \setminus X'} s[\tau_i(x)] \right) \cdot \sum_{k \in K_i} e_k \right).$$

By the definition of $\mathtt{Grp}(\mathtt{Pr}_{X',S}(\mathtt{Grp}(L_1)))$, we then have that $e = \bar{e}$, as required.

We next prove property (b). Let τ be an arbitrary substitution such that $\tau = \tau'_1 \cup \tau'_2$ for pairs $\langle \tau'_1, e'_1 \rangle \in L_1$ and $\langle \tau'_2, e'_2 \rangle \in L_2$ such that $\tau'_1(x) = \tau'_2(x)$ for each $x \in \mathsf{dom}(\tau'_1) \cap \mathsf{dom}(\tau'_2)$. Then, by definition, there exists a unique occurrence of a tuple $\langle \tau, e \rangle$ in $\mathtt{Grp}(L_1 \bowtie L_2)$, where e is functionally determined by τ. Similarly, there exists a unique occurrence of a tuple $\langle \tau, \bar{e} \rangle$ in $\mathtt{Grp}(\mathtt{Grp}(L_1) \bowtie L_2)$, where $\bar{e}$ is also functionally determined by τ. We next show that $e = \bar{e}$.

Let $\langle \tau_1, e_1 \rangle, \ldots, \langle \tau_n, e_n \rangle$ be an enumeration of the multiset L_1; also, for each $i \in [1, n]$, let $\langle \tau_{i,1}, e_{i,1} \rangle, \ldots, \langle \tau_{i,m_i}, e_{i,m_i} \rangle$ be an enumeration of the multiset that contains each distinct occurrence of $\langle \tau_{i,j}, e_{i,j} \rangle$ in L_2 such that $\tau = \tau_i \cup \tau_{i,j}$ (note that m_i may be 0). By the definition of $\mathtt{Grp}(L_1 \bowtie L_2)$, we have that

$$e \;=\; \sum_{1 \leq i \leq n} \sum_{1 \leq j \leq m_i} e_i \cdot e_{i,j} \;=\; \sum_{1 \leq i \leq n} e_i \cdot \sum_{1 \leq j \leq m_i} e_{i,j}.$$

Next, let I be the set that, for each $j \in [1, n]$, contains the least index $i \in [1, j]$ such that $\tau_i = \tau_j$. Furthermore, for each $i \in I$, let K_i contain each index $k \in [i, n]$ such that $\tau_k = \tau_i$. Then, we have that

$$e \;=\; \sum_{i \in I} \sum_{k \in K_i} \left(e_k \cdot \sum_{1 \leq j \leq m_i} e_{i,j} \right).$$

Please note that, for each $i \in I$ and each $k \in K_i$, we have that

$$\sum_{1 \leq j \leq m_i} e_{i,j} \;=\; \sum_{1 \leq j \leq m_k} e_{k,j}.$$

Thus, we can factor the equation for e as follows:

$$e \;=\; \sum_{i \in I} \left(\sum_{k \in K_i} e_k \right) \cdot \sum_{1 \leq j \leq m_i} e_{i,j}.$$

By the definition of $\texttt{Grp}(\texttt{Grp}(L_1) \bowtie L_2)$, we have that $e = \bar{e}$, as required.

We now prove property (c). Consider two distinct occurrences $\langle \tau_1, e_1 \rangle$ and $\langle \tau_2, e_2 \rangle$ in L_1 and L_2, respectively, such that $\tau_1(x) = \tau_2(x)$ for each $x \in \mathsf{dom}(\tau_1) \cap \mathsf{dom}(\tau_2)$. By the definition of $\texttt{Pr}_{X_1,S}(L_1 \bowtie L_2)$, these two pairs contribute to a unique distinct occurrence of a pair $\langle \tau, e \rangle$ in $\texttt{Pr}_{X_1,S}(L_1 \bowtie L_2)$ where $\tau = (\tau_1 \cup \tau_2)|_{X_1}$ and

$$e = \left(\prod_{x \in \mathsf{dom}(\tau_1 \cup \tau_2) \setminus X_1} s[(\tau_1 \cup \tau_2)(x)] \right) \cdot e_1 \cdot e_2.$$

Similarly, $\langle \tau_1, e_1 \rangle$ and $\langle \tau_2, e_2 \rangle$ contribute a unique occurrence of $\langle \bar{\tau}, \bar{e} \rangle$ in $L_1 \bowtie \texttt{Pr}_{X_1 \cap X_2,S}(L_2)$ where $\bar{\tau} = \tau_1 \cup (\tau_2|_{X_1 \cap X_2})$, and

$$\bar{e} = e_1 \cdot \left(\prod_{x \in \mathsf{dom}(\tau_2) \setminus (X_1 \cap X_2)} s[\tau_2(x)] \right) \cdot e_2.$$

Since $\mathsf{dom}(\tau_1) = X_1$ and $\mathsf{dom}(\tau_2) = X_2$, we have that $\tau = \bar{\tau}$ and $e = \bar{e}$, and so $\langle \tau, e \rangle = \langle \bar{\tau}, \bar{e} \rangle$, as required. □

C.2 Properties of Partial Estimations

We now use Proposition C.1 to prove two properties of the partial estimations computed in Algorithm 1. To this end, a *path* from a node $v \in \mathsf{vert}(\mathcal{T})$ to a node $v' \in \mathsf{vert}(\mathcal{T})$ in $\mathcal{T}$ is a sequence $v_1, \ldots, v_k$ with $k > 1$ of nodes in $\mathsf{vert}(\mathcal{T})$ such that $v_1 = v$, $v_k = v'$, and v_i is a successor of v_{i-1} in $\mathcal{T}$ for each $i \in [2, k]$. A node $v' \in \mathsf{vert}(\mathcal{T})$ is a *descendant* of a node $v \in \mathsf{vert}(\mathcal{T})$ if there exists a path from v to v' in $\mathcal{T}$; furthermore, let $\mathsf{desc}(v)$ be the set of all descendants of v in $\mathcal{T}$.

The next lemma shows that, for each $v \in \mathsf{vert}(\mathcal{T})$, the partial estimation E^v computed by Algorithm 1 can be equivalently obtained by first computing the join $M^v \bowtie (\bowtie_{v' \in \mathsf{desc}(v)} M^{v'})$, then projecting the resulting partial estimation onto $\mathsf{vis}(v)$, and finally grouping the result of the projection.

Lemma C.2. *For each $v \in \mathsf{vert}(\mathcal{T})$, the partial estimation E^v computed by Algorithm 1 satisfies the following identity.*

$$E^v = \texttt{Grp}(\texttt{Pr}_{\mathsf{vis}(v),S}(M^v \bowtie (\bowtie_{v' \in \mathsf{desc}(v)} M^{v'})))$$

Proof. We proceed by structural induction on the decomposition $\mathcal{T}$ and show that the property holds for each $v \in \mathsf{vert}(\mathcal{T})$.

(*Base Case*) Let $v \in \mathsf{vert}(\mathcal{T})$ be a leaf node. Then, $\mathsf{desc}(v) = \emptyset$. Thus, $E^v = \texttt{Grp}(\texttt{Pr}_{\mathsf{vis}(v),S}(M^v))$ by line 8, and the property immediately holds.

(*Inductive Step*) Consider an arbitrary non-leaf node $v \in \mathsf{vert}(\mathcal{T})$ and assume that, for each child $v_i \in \mathsf{vert}(\mathcal{T})$ of v in $\mathcal{T}$, the property holds. By line 8, for $v_1, \ldots, v_\ell$ the children of v in $\mathcal{T}$, we have that

$$E^v = \texttt{Grp}(\texttt{Pr}_{\mathsf{vis}(v),S}(M^v \bowtie E^{v_1} \bowtie \cdots \bowtie E^{v_\ell})).$$

Then, for each child v_i of v in $\mathcal{T}$, let $L_i = M^{v_i} \bowtie (\bowtie_{v' \in \mathsf{desc}(v_i)} M^{v'})$ and let $X_i = X^{v_i} \cup \bigcup_{v' \in \mathsf{desc}(v_i)} X^{v'}$. By definition, L_i is over X_i and $\mathsf{vis}(v_i) = X^v \cap X^{v_i}$; furthermore, by the inductive hypothesis, we have that $E^{v_i} = \texttt{Grp}(\texttt{Pr}_{\mathsf{vis}(v_i),S}(L_i))$. By property 2 in the definition of HT decompositions, for each $v' \in \mathsf{desc}(v_i)$ and each variable $x \in X^{v'}$, if $x \in X^v$ then $x \in X^{v_i}$. Thus, we have that $\mathsf{vis}(v_i) = X^v \cap X_i$. By identity (c) in Proposition C.1, for each partial estimation L over X^v, we have that $L \bowtie \texttt{Pr}_{\mathsf{vis}(v_i),S}(L_i) = \texttt{Pr}_{X^v,S}(L \bowtie L_i)$. In addition, by the definition of the projection operator, we have that $\texttt{Pr}_{X^v,S}(L) = L$. Since $M^v \bowtie E^{v_1} \bowtie \cdots \bowtie E^{v_\ell}$ is over X^v, we can rewrite E^v as follows:

$$\begin{aligned}
E^v \;&=\; \texttt{Grp}(\texttt{Pr}_{\mathsf{vis}(v),S}(M^v \bowtie E^{v_1} \bowtie \cdots \bowtie E^{v_\ell})) && \text{(by line 8 of Algorithm 1)} \\
&=\; \texttt{Grp}(\texttt{Pr}_{\mathsf{vis}(v),S}(M^v \bowtie \texttt{Grp}(\texttt{Pr}_{\mathsf{vis}(v_1),S}(L_1)) \bowtie \cdots \bowtie \texttt{Grp}(\texttt{Pr}_{\mathsf{vis}(v_\ell),S}(L_\ell)))) && \text{(by the IH)} \\
&=\; \texttt{Grp}(\texttt{Pr}_{\mathsf{vis}(v),S}(M^v \bowtie \texttt{Pr}_{\mathsf{vis}(v_1),S}(L_1) \bowtie \cdots \bowtie \texttt{Pr}_{\mathsf{vis}(v_\ell),S}(L_\ell))) && \text{(by identities (a) and (b) in Prop. C.1)} \\
&=\; \texttt{Grp}(\texttt{Pr}_{\mathsf{vis}(v),S}(\texttt{Pr}_{X^v,S}(M^v \bowtie L_1 \bowtie \cdots \bowtie L_\ell))) && \text{(by identity (c) in Prop. C.1)} \\
&=\; \texttt{Grp}(\texttt{Pr}_{\mathsf{vis}(v),S}(M^v \bowtie L_1 \bowtie \cdots \bowtie L_\ell)). && \text{(by } \texttt{Pr}_{X^v,S}(L) = L \text{ for } L \text{ over } X^v)
\end{aligned}$$

By definition, we have that $L_1 \bowtie \cdots \bowtie L_\ell = \bowtie_{v' \in \mathsf{desc}(v)} M^{v'}$, and so the property holds. □

Next, let $\mathsf{vert}(\mathcal{T}) = \{v_0, \ldots, v_n\}$ and let v_0 be the root of $\mathcal{T}$. By definition, we then have that $\mathsf{vis}(v_0) = \emptyset$. Furthermore, by Lemma C.2, we have that $E^{v_0} = \texttt{Grp}(\texttt{Pr}_{\emptyset}(M^{v_0} \bowtie \cdots \bowtie M^{v_n}))$. The next lemma shows that the result of $M^{v_0} \bowtie \cdots \bowtie M^{v_n}$ contains all the information necessary to computing $E_{q,S}$.

LEMMA C.3. *The following two properties hold for* $\mathsf{vert}(\mathcal{T}) = \{v_0, \ldots, v_n\}$ *and* $L = M^{v_0} \bowtie \cdots \bowtie M^{v_n}$.

(1) For each $\tau \in [\![\mu(q)]\!]_H$, *a distinct occurrence of* $\langle \tau, \prod_{a \in q} \frac{w[\tau(\mu(a))]}{s[\tau(\mu(a))]} \rangle$ *exists in* L.

(2) For each distinct occurrence of $\langle \tau, e \rangle$ *in* L, *we have that* $\tau \in [\![\mu(q)]\!]_H$.

PROOF. We start by showing property (1). To this end, consider an arbitrary $\tau \in [\![\mu(q)]\!]_H$ and, for each $i \in [0, n]$, let $\tau_i = \tau|_{\mathsf{var}(p^{v_i})}$. Since $p^{v_i} \subseteq q$ and $\tau \in [\![\mu(q)]\!]_H$, we also have that $\tau_i \in [\![\mu(p^{v_i})]\!]_H$. By the definition of the decomposition, we have that $\mathsf{var}(q) = X^{v_0} \cup \cdots \cup X^{v_n}$; furthermore, for each $i \in [0, n]$, we have that $X^{v_i} \subseteq \mathsf{var}(p^{v_i})$. Hence, $\tau = (\tau_1|_{X^{v_1}}) \cup \cdots \cup (\tau_n|_{X^{v_n}})$. By the definition of M^{v_i}, for each $i \in [0, n]$, there exist a distinct occurrence of a tuple $\langle \tau_i|_{X^{v_i}}, e_i \rangle$, where

$$e_i \;=\; \prod_{a \in p_*^{v_i}} \frac{w[\tau_i(\mu(a))]}{s[\tau_i(\mu(a))]}.$$

By the definition of the join operator $\bowtie$ on partial estimations, the partial estimation L contains a distinct occurrence of

$$\langle (\tau_0|_{X^{v_0}}) \cup \cdots \cup (\tau_n|_{X^{v_n}}),\; e_0 \cdot \ldots \cdot e_n \rangle \;=\; \langle \tau, e_0 \cdot \ldots \cdot e_n \rangle,$$

where

$$e_0 \cdot \ldots \cdot e_n \;=\; \prod_{i=0}^{n} \prod_{a \in p_*^{v_i}} \frac{w[\tau_i(\mu(a))]}{s[\tau_i(\mu(a))]}.$$

By lines 1–3 of Algorithm 1, we have that $\{p_*^{v_i} \mid p_*^{v_i} \neq \emptyset\}$ is a partition of q. Thus, we have that

$$e_0 \cdot \ldots \cdot e_n \;=\; \prod_{i=0}^{n} \prod_{a \in p_*^{v_i}} \frac{w[\tau_i(\mu(a))]}{s[\tau_i(\mu(a))]} \;=\; \prod_{a \in q} \frac{w[\tau(\mu(a))]}{s[\tau(\mu(a))]},$$

as required.

We next prove property (2). Consider an arbitrary distinct occurrence of $\langle \tau, e \rangle$ in L. By the definition of the join operator $\bowtie$ on partial estimations, for each $i \in [0, n]$, there exists a unique distinct occurrence of a tuple $\langle \tau_i, e_i \rangle$ in M^{v_i} such that $\mathsf{dom}(\tau_i) = X^{v_i}$ and $\tau = \tau_0 \cup \cdots \cup \tau_n$. By line 7 of Algorithm 1, for each such τ_i, there exists $\bar{\tau}_i \supseteq \tau_i$ such that $\bar{\tau}_i \in [\![\mu(p^{v_i})]\!]_H$. Furthermore, because $p_*^{v_i} \subseteq p^{v_i}$ and $\mathsf{var}(p_*^{v_i}) \subseteq X^{v_i} \subseteq \mathsf{var}(p^{v_i})$, we also have that $\tau_i(\mu(p_*^{v_i})) \subseteq H$. Since $q = p_*^{v_1} \cup \cdots \cup p_*^{v_n}$, we have that $\tau \in [\![\mu(q)]\!]_H$, as required. □

C.3 Proof of Theorem 5.3

We are now ready to prove the correctness of our decomposition method.

THEOREM 5.3. *Algorithm 1 computes* $E_{q,S}$ *in time exponential in* $\mathsf{hw}(\mathcal{D})$ *but polynomial in the size of* q *and* S.

PROOF. We start by proving that Algorithm 1 computes $E_{q,S}$. To this end, let $\mathsf{vert}(\mathcal{T}) = \{v_0, \ldots, v_n\}$ and let v_0 be the root of $\mathcal{T}$. Then, by Lemma C.2, we have that

$$E^{v_0} = \mathtt{Grp}(\mathtt{Pr}_{\mathsf{vis}(v_0),S}(M^{v_0} \bowtie \cdots \bowtie M^{v_n})).$$

By Lemma C.3, the partial estimation $M^{v_0} \bowtie \cdots \bowtie M^{v_n}$ over $\mathsf{var}(q)$ contains precisely one distinct occurrence of

$$\langle \tau,\; \prod_{a \in q} \frac{w[\tau(\mu(a))]}{s[\tau(\mu(a))]} \rangle$$

for each $\tau \in [\![\mu(q)]\!]_H$. Since v_0 is the root of $\mathcal{T}$, we have that $\mathsf{vis}(v_0) = \emptyset$ by definition. So $\mathtt{Pr}_{\mathsf{vis}(v_0),S}(M^{v_0} \bowtie \cdots \bowtie M^{v_n})$ contains, for each $\tau \in [\![\mu(q)]\!]_\tau$, a unique distinct occurrence of

$$\langle \tau_\emptyset,\; \prod_{x \in \mathsf{var}(q)} s[\tau(x)] \prod_{a \in q} \frac{w[\tau(\mu(a))]}{s[\tau(\mu(a))]} \rangle.$$

Thus, by the definition of the grouping operator, $\mathtt{Grp}(E^{v_0})$ contains precisely one occurrence of $\langle \tau_\emptyset, e \rangle$ where

$$e \;=\; \sum_{\tau \in [\![\mu(q)]\!]_H} \left(\prod_{x \in \mathsf{var}(q)} s[\tau(x)] \prod_{a \in q} \frac{w[\tau(\mu(a))]}{s[\tau(\mu(a))]} \right).$$

Please note that q is μ-unification-free; so, by Proposition 4.6, we have that $e = E_{q,S}$, as required.

We next show that Algorithm 1 runs in time exponential in $\mathsf{hw}(\mathcal{D})$ but polynomial in the size of q and H. In the following, we use a model of computation based on Turing machines in which numbers are encoded in binary. We represent each non-negative rational number explicitly as the fraction $\frac{e_1}{e_2}$ of two integers $e_1 \geq 0$ and $e_2 > 0$. The number of bits required to store $\frac{e_1}{e_2}$ is thus the sum of the numbers of bits required to store e_1 and e_2, respectively. By Theorem 4.7, it follows that all the numbers computed by Algorithm 1 can be stored using a number of bits polynomial in the size of the input.

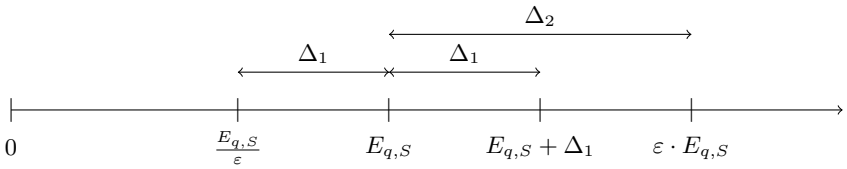

Figure 6: The relative positions of $\frac{E_{q,S}}{\varepsilon}$, $E_{q,S}$, and $E_{q,S} \cdot \varepsilon$ on the line

Lines 1–3 can clearly be computed in time polynomial in the input size; so, we next focus on lines 4–8. Consider an arbitrary vertex $v \in \mathsf{vert}(\mathcal{T})$. Since the size of p^v is bounded by $\mathsf{hw}(\mathcal{D})$, the partial estimation M^v in line 8 can be computed using a standard join algorithm in time exponential in $\mathsf{hw}(\mathcal{D})$ but polynomial in the size of q and S. On line 7, for each child v_i of v, partial estimation E^{v_i} is over the variables $\mathsf{vis}(v_i) \subseteq X^{v_i}$. Hence, the partial estimation $M^v \bowtie E^{v_1} \bowtie \cdots \bowtie E^{v_\ell}$ can be computed using a semi-join algorithm in polynomial time in the input size, and its size is bounded by the size of M^v. Finally, E^v is obtained from $M^v \bowtie E^{v_1} \bowtie \cdots \bowtie E^{v_\ell}$ by grouping and projecting. Both operations can be done in time polynomial in the input size. We then conclude that each E^v can be computed in time exponential in $\mathsf{hw}(\mathcal{D})$ but polynomial in the size of q and S, as required. □

D. PROOF OF SECTION 7

To prove Proposition 7.1, we first prove the following auxiliary lemma.

LEMMA D.1. *The probability that the q-error of a CQ q on a graph $G \in R_S$ chosen uniformly at random is less than $\varepsilon \geq 1$ is at least $1 - (\frac{\varepsilon \cdot \sigma_{q,S}}{(\varepsilon-1)\cdot E_{q,S}})^2$.*

PROOF. Let $S = \langle H, w, \mu \rangle$ be a consistent summary, let q be a CQ such that $\mathsf{res}(q) \subseteq \mathsf{dom}(\mu)$, and let $\varepsilon \geq 1$ be a rational number. Please recall that, for an arbitrary graph $G \in R_S$, the q-error of q on G is defined as

$$\max\left(\frac{|[\![q]\!]_G|}{E_{q,S}}, \frac{E_{q,S}}{|[\![q]\!]_G|}\right).$$

Now, consider an arbitrary graph $G \in R_S$ such that the q-error of q on G is at most ε. By the definition of the q-error, we must have that

$$\frac{E_{q,S}}{\varepsilon} \leq |[\![q]\!]_G| \leq E_{q,S} \cdot \varepsilon.$$

Since $\varepsilon \geq 1$, we also have that $\frac{E_{q,S}}{\varepsilon} \leq E_{q,S} \leq E_{q,S} \cdot \varepsilon$. Then, let

$$\Delta_1 = E_{q,S} - \frac{E_{q,S}}{\varepsilon} = \frac{E_{q,S} \cdot (\varepsilon - 1)}{\varepsilon},$$
$$\Delta_2 = E_{q,S} \cdot \varepsilon - E_{q,S} = E_{q,S} \cdot (\varepsilon - 1).$$

Due to $\varepsilon \geq 1$, we also have that $\Delta_1 \leq \Delta_2$. Figure 6 recapitulates the relative positions of $\frac{E_{q,S}}{\varepsilon}$, $E_{q,S}$, and $E_{q,S} \cdot \varepsilon$.

Next, let Y_{ans} and Y_{err} be two random variables mapping each RDF graph $G \in R_S$ as follows:

$$Y_{ans}(G) = |[\![q]\!]_G| \qquad \text{and} \qquad Y_{err}(G) = \max\left(\frac{|[\![q]\!]_G|}{E_{q,S}}, \frac{E_{q,S}}{|[\![q]\!]_G|}\right).$$

We next show that $\mathsf{Pr}(Y_{err} \leq \varepsilon) > 1 - \left(\frac{\varepsilon \cdot \sigma_{q,S}}{(\varepsilon-1)\cdot E_{q,S}}\right)^2$.

For each graph $G \in R_S$ such that the q-error of q on G is at most ε, we have that $E_{q,S} - \Delta_1 \leq |[\![q]\!]_G| \leq E_{q,S} + \Delta_2$. Furthermore, we have that $\Delta_1 \leq \Delta_2$, and so

$$\mathsf{Pr}(Y_{err} \leq \varepsilon) = \mathsf{Pr}(|Y_{ans} - E_{q,S}| \leq \Delta_2) = \mathsf{Pr}(|Y_{ans} - E_{q,S}| < \Delta_1 \text{ or } E_{q,S} + \Delta_1 \leq Y_{ans} \leq E_{q,S} \cdot \varepsilon).$$

Please note that $|Y_{ans} - E_{q,S}| < \Delta_1$ and $E_{q,S} + \Delta_1 \leq Y_{ans} \leq E_{q,S} \cdot \varepsilon$ are independent events, and so

$$\mathsf{Pr}(|Y_{ans} - E_{q,S}| < \Delta_1 \text{ or } E_{q,S} + \Delta_1 \leq Y_{ans} \leq E_{q,S} \cdot \varepsilon) = \mathsf{Pr}(|Y_{ans} - E_{q,S}| < \Delta_1) + \mathsf{Pr}(E_{q,S} + \Delta_1 \leq Y_{ans} \leq E_{q,S} \cdot \varepsilon).$$

But then, we can bound from below the probability that the q-error is at most ε as follows:

$$\mathsf{Pr}(Y_{err} \leq \varepsilon) = \mathsf{Pr}(|Y_{ans} - E_{q,S}| < \Delta_1) + \mathsf{Pr}(E_{q,S} + \Delta_1 \leq Y_{ans} \leq E_{q,S} \cdot \varepsilon) \geq \mathsf{Pr}(|Y_{ans} - E_{q,S}| < \Delta_1). \tag{10}$$

By Chebyschev's inequality, for $\Delta_1 = k \cdot \sigma_{q,S}$, we have that

$$\mathsf{Pr}(|Y_{ans} - E_{q,S}| < \Delta_1) > 1 - \left(\frac{\sigma_{q,S}}{\Delta_1}\right)^2.$$

Since $\Delta_1 = \frac{E_{q,S} \cdot (\varepsilon - 1)}{\varepsilon}$, we then have that

$$\mathsf{Pr}(|Y_{ans} - E_{q,S}| < \Delta_1) > 1 - \left(\frac{\varepsilon \cdot \sigma_{q,S}}{(\varepsilon - 1) \cdot E_{q,S}} \right)^2.$$

By Equation (10), we then have that

$$\mathsf{Pr}(Y_{err} \leq \varepsilon) > 1 - \left(\frac{\varepsilon \cdot \sigma_{q,S}}{(\varepsilon - 1) \cdot E_{q,S}} \right)^2,$$

as required. □

PROPOSITION 7.1. *The q-error of any CQ q is less than $\varepsilon \geq 1$ with probability at least $\wp$ if $\frac{\sigma_{q,S}}{E_{q,S}} < (1 - \frac{1}{\varepsilon})\sqrt{1 - \wp}$.*

PROOF. Let $S = \langle H, w, \mu \rangle$ be a consistent summary, and let q be a CQ such that $\mathsf{res}(q) \subseteq \mathsf{dom}(\mu)$. Furthermore, let ε be an arbitrary rational number with $\varepsilon \geq 1$, and let $\wp$ be an arbitrary real number such that $0 \leq \wp \leq 1$. We next show that the probability that the q-error is less than ε is at least $\wp$ if $\frac{\sigma_{q,S}}{E_{q,S}} \leq (1 - \frac{1}{\varepsilon})(\sqrt{1 - \wp})$. By Lemma D.1, the probability that the q-error of q on a randomly chosen graph $G \in R_S$ is less than ε is at least $\wp$ if

$$1 - \left(\frac{\varepsilon \cdot \sigma_{q,S}}{(\varepsilon - 1) \cdot E_{q,S}} \right)^2 > \wp.$$

By manipulating the inequality using the standard identities from arithmetic, we obtain that

$$\left(\frac{\sigma_{q,S}}{E_{q,S}} \right)^2 < \left(1 - \frac{1}{\varepsilon} \right)^2 \cdot (1 - \wp).$$

Thus, by taking the square root on both sides of the inequality, we have that

$$\left(\frac{\sigma_{q,S}}{E_{q,S}} \right) < \left(1 - \frac{1}{\varepsilon} \right) \cdot \sqrt{1 - \wp},$$

and the proposition holds. □