Uncertainty That Counts

Dany Maslowski and Jef Wijsen

Université de Mons, Mons, Belgium
{dany.maslowski,jef.wijsen}@umons.ac.be

Abstract. Uncertainty is modeled by a multibase $(\mathbf{db}, \mu)$ where $\mathbf{db}$ is a database with zero or more primary key violations, and μ associates a multiplicity (a positive integer) to each fact of $\mathbf{db}$. In data integration, the multiplicity of a fact g can indicate the number of data sources in which g was found. In planning databases, facts with the same primary key value are alternatives for each other, and the multiplicity of a fact g can denote the number of people in favor of g.

A repair of $\mathbf{db}$ is obtained by selecting a maximal number of facts without ever selecting two distinct facts of the same relation that agree on their primary key. Every repair has a support count, which is the product of the multiplicities of its facts.

For a fixed Boolean query q, we define $\sigma\mathsf{CERTAINTY}(q)$ as the following counting problem: Given a multibase $(\mathbf{db}, \mu)$, determine the weighted number of repairs of $\mathbf{db}$ that satisfy q. Here, every repair is weighted by its support count. We illustrate the practical significance of this problem by means of examples.

For conjunctive queries q without self-join, we provide a syntactic characterization of the class of queries q such that $\sigma\mathsf{CERTAINTY}(q)$ is in $\mathbf{P}$; for queries not in this class, $\sigma\mathsf{CERTAINTY}(q)$ is $\natural\mathbf{P}$-hard (and hence highly intractable).

1 Motivation

Many database applications require integrating data from multiple sources. Some data sources may store inaccurate or outdated values for the same entity [7]. Users of the database often do not know which value is the correct one. This leads to uncertainty.

For example, Figure 1 shows a database resulting from the integration of three distinct source databases. Primary keys are underlined: the business rules impose that every department has a single budget and manager; every employee has a single first name, last name, and address. Unfortunately, the three source databases contained conflicting information. The *multiplicity* column μ indicates the number of source databases recording a given fact: two source databases recorded that the Toys department is managed by employee 456, while one database recorded that Toys is managed by employee 123. Two source databases recorded that Shoes is managed by employee 123; the remaining database did not store this department. Also, the source databases did not agree on the address of Ann Smith: the address 8 Corn St. was found in two databases, while

H. Christiansen et al. (Eds.): FQAS 2011, LNAI 7022, pp. 25–36, 2011.

6 Main St. was found once. Luckily, the three source databases agreed on John Kipling's address.

A *repair* of a database **db** is a maximal subset of **db** that satisfies the primary key constraints. The integrated database of Figure 1 has four repairs, because there are two choices for the manager of Toys, and two choices for the address of Ann Smith. Figure 2 shows the four repairs. Significantly, if we have equal trust (or suspicion) in each source database, then these four repairs do not have the same likelihood. The repair $\mathbf{r}_1$ stores the most agreed upon values for the Toys' manager and for Ann Smith's address, while $\mathbf{r}_4$ stores values for these items that were found only once. This can be quantified by the *support count* of a repair, which is obtained by multiplying the multiplicities of its facts. The repairs $\mathbf{r}_1$ and $\mathbf{r}_4$ have support counts 24 and 6, respectively. Alternatively, these numbers could be conveniently represented as a fraction of the total sum of support counts of all repairs; in this example, the support counts sum up to $54 = 24 + 12 + 12 + 6$.

DEPT

DName	Budget	Mgr	μ
Toys	10K	456	2
Toys	10K	123	1
Shoes	12K	123	2

EMP

E#	FName	LName	Address	μ
123	Ann	Smith	8 Corn St.	2
123	Ann	Smith	6 Main St.	1
456	John	Kipling	7 River St.	3

Fig. 1. Integrated company database

Given a Boolean query q, we want to know the weighted number of repairs in which q evaluates to true; here, every repair is weighted by its support count. For example, the following query q_1 asks whether the manager of the Toys department lives in '6 Main St.' The query q_1 is only true in $\mathbf{r}_4$ with support count 6.

$$q_1 = \exists x \exists y \exists z \exists u \big(\mathbf{DEPT}(\underline{\text{'Toys'}}, x, y) \wedge \mathbf{EMP}(\underline{y}, z, u, \text{'6 Main St.'})\big)$$

The following query q_2 asks whether the Toys' manager lives in '7 River St.' The query q_2 is true in $\mathbf{r}_1$ and $\mathbf{r}_2$, whose support counts sum up to $36 = 24 + 12$.

$$q_2 = \exists x \exists y \exists z \exists u \big(\mathbf{DEPT}(\underline{\text{'Toys'}}, x, y) \wedge \mathbf{EMP}(\underline{y}, z, u, \text{'7 River St.'})\big)$$

These figures could be conveniently presented as fractions of the total sum of support counts: the support fractions of q_1 and q_2 are $\frac{6}{54}$ and $\frac{36}{54}$, respectively.

Multiplicities can arise in many applications. In a conference planning database, for example, the multiplicities may indicate the number of steering committee members that are in favor of a given conference location. In the database of Figure 3, eight members are favorable to organizing FQAS 2015 in Seattle. The support counts of all repairs sum up to 100. The following query q_3 asks the support for organizing FQAS in North-America in some year:

$$q_3 = \exists x \exists y \big(\mathbf{CONF}(\underline{\text{'FQAS'}}, x, y, \text{'North-America'})\big)$$

DEPT

DName	Budget	Mgr	μ
Toys	10K	456	2
Shoes	12K	123	2

EMP

SS#	FName	LName	Address	μ
123	Ann	Smith	8 Corn St.	2
456	John	Kipling	7 River St.	3

Repair $\mathbf{r}_1$ with support count $24 = 2 \times 2 \times 2 \times 3$

DEPT

DName	Budget	Mgr	μ
Toys	10K	456	2
Shoes	12K	123	2

EMP

SS#	FName	LName	Address	μ
123	Ann	Smith	6 Main St.	1
456	John	Kipling	7 River St.	3

Repair $\mathbf{r}_2$ with support count $12 = 2 \times 2 \times 1 \times 3$

DEPT

DName	Budget	Mgr	μ
Toys	10K	123	1
Shoes	12K	123	2

EMP

SS#	FName	LName	Address	μ
123	Ann	Smith	8 Corn St.	2
456	John	Kipling	7 River St.	3

Repair $\mathbf{r}_3$ with support count $12 = 1 \times 2 \times 2 \times 3$

DEPT

DName	Budget	Mgr	μ
Toys	10K	123	1
Shoes	12K	123	2

EMP

SS#	FName	LName	Address	μ
123	Ann	Smith	6 Main St.	1
456	John	Kipling	7 River St.	3

Repair $\mathbf{r}_4$ with support count $6 = 1 \times 2 \times 1 \times 3$

Fig. 2. Four repairs $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$, $\mathbf{r}_4$ with their support counts

For the example table, there is only one repair in which FQAS is *not* organized in North-America; the support count of this repair is $18 = 2 \times 9$. Consequently, the support count for organizing FQAS in North-America is 82; the support fraction of q_3 is thus $\frac{82}{100}$.

CONF

Conf	Year	Town	Continent	μ
FQAS	2015	Seattle	North-America	8
FQAS	2015	Berlin	Europe	2
FQAS	2016	Paris	Europe	9
FQAS	2016	Dallas	North-America	1

Fig. 3. A conference planning table

Formally, for a given Boolean query q, we define $\sigma\mathsf{CERTAINTY}(q)$ as the following problem: given a database **db** with primary key violations and multiplicities for all facts, determine the weighted number of repairs in which q evaluates to true. In this article, we study $\sigma\mathsf{CERTAINTY}(q)$ for queries q that belong to the class SJFCQ, which is the class of Boolean conjunctive queries that are self-join-free (that is, in which no relation name occurs more than once). Unfortunately, there are queries q in this class for which the data complexity of $\sigma\mathsf{CERTAINTY}(q)$ is highly intractable (in particular, $\natural\mathbf{P}$-hard). In this article, we provide a syn-

tactic characterization of the SJFCQ queries q such that σCERTAINTY(q) is tractable (that is, in **P**).

Recall that the class ♮**P** contains the counting variant of problems in **NP**. By Toda's theorem [12], every problem in the polynomial-time hierarchy can be solved in polynomial time given an oracle that solves a ♮**P**-complete problem. Thus, ♮**P**-hardness suggests a higher level of intractability than **NP**-hardness, insofar decision problems and counting problems can be compared.

In summary, the contribution made by this article is twofold:

1. We propose to model uncertainty by primary key violations plus multiplicities. Multiplicities may be useful in practice, because they allow to model the support for a given database fact. They arise, for example, as the result of a voting process.
2. We give a sound and complete syntactic characterization of the self-join-free conjunctive queries whose data complexity is tractable.

The remainder of this article is organized as follows. Section 2 formally defines our data model and the problem of interest. We focus on Boolean conjunctive queries in which each relation name is used at most once. Section 3 discusses related work. Section 4 determines a sufficient and necessary condition, called safety[1], on queries q under which σCERTAINTY(q) is tractable. Section 5 concludes the article. Several proofs are available in a separate appendix.

2 Preliminaries

2.1 Basic Notions

Data Model We define $\mathbb{N} = \{0, 1, 2, \ldots\}$. Each relation name R of arity n, $n \geq 1$, has a unique *primary key* which is a set $\{1, 2, \ldots, k\}$ where $1 \leq k \leq n$. We say that R has *signature* $[n, k]$ if R has arity n and primary key $\{1, 2, \ldots, k\}$. Elements of the primary key are called *primary-key positions*, while $k+1$, $k+2$, $\ldots$, n are *non-primary-key positions*. For all positive integers n, k such that $1 \leq k \leq n$, we assume denumerably many relation names with signature $[n, k]$.

We assume a denumerable set **dom** of *constants*, disjoint from a denumerable set **vars** of *variables*. If R is a relation name of signature $[n, k]$, and $s_1, \ldots, s_n$ are variables or constants, then $R(\underline{s_1, \ldots, s_k}, s_{k+1}, \ldots, s_n)$ is an *R-goal* (or simply *goal* if R is understood). Notice that primary key positions are underlined. An *R-fact* (or simply *fact*) is a goal in which no variable occurs. Two R-facts g and h are *key-equal* if they agree on all primary-key positions. Every fact is key-equal to itself.

A *database* **db** is a finite set of facts. Such database may violate primary keys, and so capture uncertainty. Given a database **db**, we write **adom**(**db**) for the set of constants that occur in **db**.

[1] This notion is unrelated to the notion of safety that guarantees domain independence in relational calculus [1, page 75].

A database $\mathbf{db}$ is called *consistent* if it contains no two distinct, key-equal facts. A *repair* $\mathbf{r}$ of a database $\mathbf{db}$ is a maximal subset of $\mathbf{db}$ that is consistent. If g is a fact of $\mathbf{db}$, then $\mathsf{block}(g, \mathbf{db})$ is the subset of $\mathbf{db}$ containing each fact that is key-equal to g. The sets $\mathsf{block}(g, \mathbf{db})$ are also called *blocks*. Intuitively, repairs are obtained by choosing exactly one fact from each block.

A *multibase* is a pair $(\mathbf{db}, \mu)$ where $\mathbf{db}$ is a database and μ is a total function $\mu : \mathbf{db} \to \mathbb{N} \setminus \{0\}$. The *support count* of a repair $\mathbf{r}$, denoted $\sigma(\mathbf{r}, \mathbf{db}, \mu)$, is defined by:

$$\sigma(\mathbf{r}, \mathbf{db}, \mu) = \prod \{\mu(g) \mid g \in \mathbf{r}\} \ .$$

For every $g \in \mathbf{db}$, we define the support count of its block as follows:

$$\sigma\mathsf{block}(g, \mathbf{db}, \mu) = \sum \{\mu(h) \mid h \in \mathsf{block}(g, \mathbf{db})\} \ .$$

Notice the use of $\prod$ in the case of repairs, and $\sum$ in the case of blocks: the facts in a repair are mutually independent, while the facts in a block are mutually exclusive.

Queries. A *Boolean conjunctive query* q is a finite set of goals. A Boolean conjunctive query $q = \{g_1, g_2, \ldots, g_n\}$ represents the first-order logic sentence $\exists x_1 \ldots \exists x_m (g_1 \wedge g_2 \ldots \wedge g_n)$, where $x_1, \ldots, x_m$ are all variables occurring in q. Such query is *self-join-free* if it contains no two distinct goals with the same relation name. Thus, every relation name occurs at most once in a self-join-free query. The class of self-join-free Boolean conjunctive queries is denoted by SJFCQ.

Let V be a finite set of variables. A *valuation* over V is a mapping $\theta : \mathbf{vars} \cup \mathbf{dom} \to \mathbf{vars} \cup \mathbf{dom}$ such that for every $x \in V$, $\theta(x) \in \mathbf{dom}$, and for every $s \notin V$, $\theta(s) = s$. Valuations extend to goals and queries in the straightforward way.

If $\boldsymbol{s}$ is a sequence of variables and constants, then $\mathsf{Vars}(\boldsymbol{s})$ denotes the set of variables that occur in $\boldsymbol{s}$. If g is a goal, then $\mathsf{Vars}(g)$ denotes the set of variables that occur in g, and $\mathsf{KVars}(g)$ denotes the subset of $\mathsf{Vars}(g)$ containing each variable that occurs at a primary-key position. If q is a query, then $\mathsf{Vars}(q)$ denotes the set of variables that occur in q.

If q is a conjunctive query, $x \in \mathsf{Vars}(q)$, and $a \in \mathbf{dom}$, then $q_{x \mapsto a}$ is the query obtained from q by replacing all occurrences of x with a.

A database $\mathbf{db}$ is said to *satisfy* Boolean conjunctive query q, denoted $\mathbf{db} \models q$, if there exists a valuation θ over $\mathsf{Vars}(q)$ such that $\theta(q) \subseteq \mathbf{db}$.

Counting Repairs. For some fixed $q \in \mathsf{SJFCQ}$, $\sigma\mathsf{CERTAINTY}(q)$ is the following problem: Given a multibase $(\mathbf{db}, \mu)$, determine the total sum of support counts of repairs of $\mathbf{db}$ that satisfy q.

Let $(\mathbf{db}, \mu)$ be a multibase and q a Boolean query. We write $\mathsf{rset}(\mathbf{db})$ for the set of repairs of $\mathbf{db}$, and $\mathsf{rset}(\mathbf{db}, q)$ for the subset of $\mathsf{rset}(\mathbf{db})$ containing each repair that satisfies q. Furthermore, we define:

$$\sigma\mathsf{rset}(\mathbf{db}, \mu) = \sum \{\sigma(\mathbf{r}, \mathbf{db}, \mu) \mid \mathbf{r} \in \mathsf{rset}(\mathbf{db})\}$$

CONF

Conf	Year	Town	Continent	P
FQAS	2015	Seattle	North-America	0.8
FQAS	2015	Berlin	Europe	0.2
FQAS	2016	Paris	Europe	0.9
FQAS	2016	Dallas	North-America	0.1

Fig. 4. The conference planning table as a probabilistic table

$$\sigma\mathsf{rset}(\mathbf{db}, \mu, q) = \sum\{\sigma(\mathbf{r}, \mathbf{db}, \mu) \mid \mathbf{r} \in \mathsf{rset}(\mathbf{db}, q)\}$$
$$\sigma\mathsf{frac}(\mathbf{db}, \mu, q) = \frac{\sigma\mathsf{rset}(\mathbf{db}, \mu, q)}{\sigma\mathsf{rset}(\mathbf{db}, \mu)}$$

Thus, for a fixed query $q \in \mathsf{SJFCQ}$, $\sigma\mathsf{CERTAINTY}(q)$ is the problem that takes as input a multibase $(\mathbf{db}, \mu)$ and asks to determine $\sigma\mathsf{rset}(\mathbf{db}, \mu, q)$.

2.2 Counts Versus Fractions

We show that the choice to work with counts or fractions throughout this article is not fundamental. In [3], it is illustrated that a database with $2n$ facts can have 2^n repairs. That is, the number of repairs can be exponential in the size of the database. Nevertheless, the following lemma implies that the total sum of support counts of all repairs can be computed in polynomial time.

Lemma 1. *Let* $(\mathbf{db}, \mu)$ *be a multibase. Let* $\mathbf{r}$ *be a repair of* $\mathbf{db}$*. Then,*

$$\sigma\mathsf{rset}(\mathbf{db}, \mu) = \prod_{g \in \mathbf{r}} \sigma\mathsf{block}(g, \mathbf{db}, \mu) \ ,$$

where, as usual, the empty product is defined to be equal to 1*.*

It follows that $\sigma\mathsf{rset}(\mathbf{db}, \mu)$ can be determined in time $\mathcal{O}(n \log n)$ where n is the cardinality of $\mathbf{db}$: sort each relation of $\mathbf{db}$ on its primary key values; for each block, determine the support count of that block, and multiply these numbers. Since $\sigma\mathsf{rset}(\mathbf{db}, \mu, q) = \sigma\mathsf{frac}(\mathbf{db}, \mu, q) \times \sigma\mathsf{rset}(\mathbf{db}, \mu)$, if we can determine $\sigma\mathsf{frac}(\mathbf{db}, \mu, q)$ in time $\mathcal{O}(f(n))$, then we can determine $\sigma\mathsf{rset}(\mathbf{db}, \mu, q)$ in time $\mathcal{O}(f(n) + n \log n)$. In particular, $\sigma\mathsf{CERTAINTY}(q)$ is in $\mathbf{P}$ if for each multibase $(\mathbf{db}, \mu)$, $\sigma\mathsf{frac}(\mathbf{db}, \mu, q)$ can be determined in polynomial time in the size of $(\mathbf{db}, \mu)$. For that reason, we can focus on determining fractions $\sigma\mathsf{frac}(\mathbf{db}, \mu, q)$ instead of counts $\sigma\mathsf{rset}(\mathbf{db}, \mu, q)$.

3 Related Work

The current article generalizes [10] by allowing multiplicities. The data model in [10] has no multiplicities, which is tantamount to setting $\mu(g) = 1$ for every database fact g. We believe that multiplicities are useful in many applications, as illustrated in Section 1.

Block-independent-disjoint probabilistic databases [4,5] use probabilities instead of multiplicities, as illustrated by Figure 4. If one requires that the probabilities within each block sum up to 1, then the difference between multiplicities and probabilities turns out to be irrelevant. It should be noted, however, that the authors of [4,5] do not require that the probabilities in a block sum up to 1, in which case a nonempty database can have an empty repair. This is different from our data model, in which a repair cannot be empty unless the original database is empty. This difference is significant. For example, Dalvi et al. [6,5] obtain an intractability result for the query $q = \{R(\underline{x}, y), S(\underline{y})\}$, whereas $\sigma\mathsf{CERTAINTY}(q)$ is tractable in our setting.

Our work can also be viewed as a variant of consistent query answering [2], which deals with the problem of computing answers to queries on databases that violate integrity constraints. In our data model, the only constraints are primary keys. Fuxman and Miller [8] were the first ones to focus on consistent conjunctive query answering under primary key violations. Their results have been extended and improved in recent works [13,14,15,11]. These works on primary key violations have focused on the question whether a query is true in every repair; in the current article, we ask to determine the weighted number of repairs in which the query is true.

Counting the fraction of repairs that satisfy a query is also studied by Greco et al. in [9]. The constraints in that work are functional dependencies, and the repairs are obtained by updates. Greco et al. present an approach for computing approximate probabilistic answers in polynomial time. We, on the other hand, characterize queries for which exact fractions can be obtained in polynomial time.

4 The Boundary of Tractability

We define a syntactically restricted class of SJFCQ queries, called *safe* queries, and show that for every safe SJFCQ query q, the problem $\sigma\mathsf{CERTAINTY}(q)$ is in **P**. Moreover, we show that for every unsafe SJFCQ query q, it is the case that $\sigma\mathsf{CERTAINTY}(q)$ is $\natural\mathbf{P}$-hard. The outline is as follows: Lemmas 2–6 first provide arithmetic expressions for $\sigma\mathsf{frac}(\mathbf{db}, \mu, q)$ under particular syntactic conditions on q; these lemmas are then combined in an algorithm that defines the safe queries.

The following lemma implies that we can compute in polynomial time the total sum of support counts of repairs that contain a given fact.

Lemma 2 (SE0a). *Let g be a fact. Let $q = \{g\}$, an* SJFCQ *query. Then, for every multibase $(\mathbf{db}, \mu)$,*

$$\sigma\mathsf{frac}(\mathbf{db}, \mu, q) = \begin{cases} 0 & \text{if } g \notin \mathbf{db} \\ \frac{\mu(g)}{\sigma\mathsf{block}(g, \mathbf{db}, \mu)} & \text{if } g \in \mathbf{db} \end{cases}$$

Lemma 3 deals with queries that are either satisfied or falsified by all repairs. An example of such query is $q_0 = \{R(\underline{x, x, y}, z)\}$. Trivially, a database $\mathbf{db}$ satisfies

Algorithm *IsSafe*(q)
Input: SJFCQ query q
Output: Boolean in $\{\mathbf{true}, \mathbf{false}\}$
1. (∗ SE0a ∗)
2. **if** $q = \{g\}$ with $\mathsf{Vars}(g) = \emptyset$ **then return true**
3.
4. (∗ SE0b ∗)
5. **if** $[\![q]\!] = \emptyset$ **then return true**
6.
7. (∗ SE1 ∗)
8. **if** $q = q_1 \uplus q_2$ **and** $q_1 \neq \emptyset \neq q_2$ **and** $\mathsf{Vars}(q_1) \cap \mathsf{Vars}(q_2) = \emptyset$
9. **then return** $(\mathit{IsSafe}(q_1) \wedge \mathit{IsSafe}(q_2))$
10.
11. (∗ SE2 ∗)
12. **if** $[\![q]\!] \neq \emptyset$ **and** $\exists x \forall g \in [\![q]\!](x \in \mathsf{KVars}(g))$ **then return** $\mathit{IsSafe}(q_{x \mapsto a})$
13. (∗ a is any fixed constant ∗)
14.
15. (∗ SE3 ∗)
16. **if** $\exists x \exists g \in q(\mathsf{KVars}(g) = \emptyset, x \in \mathsf{Vars}(g))$ **then return** $\mathit{IsSafe}(q_{x \mapsto a})$
17.
18. (∗ Otherwise ∗)
19. **if** none of the above **then return false**

Fig. 5. Algorithm *IsSafe*

q_0 if and only if **db** contains an R-fact of the form $R(\underline{a, a, b, c})$, for some constants a, b, c. Moreover, if **db** contains an atom of this form, then every repair of **db** will contain an atom of this form, and thus satisfy q_0. Inversely, if **db** contains no atom of this form, then no repair of **db** will satisfy q_0.

Definition 1. *Let q be an* SJFCQ *query. A variable $x \in \mathsf{Vars}(q)$ is called a* liaison variable *if x has at least two occurrences in q.*[2] *A variable $y \in \mathsf{Vars}(q)$ is called an* orphan variable *if y occurs only once in q and the only occurrence of y is at a non-primary-key position.*

The complex part *of an* SJFCQ *query q, denoted $[\![q]\!]$, contains every goal $g \in q$ such that some non-primary-key position in g contains a liaison variable or a constant.*

Example 1. Let $q = \{R(\underline{x}, y), S(\underline{y}, z), T(\underline{y}, u, a)\}$. Then, y is a liaison variable because y occurs more than once in q. The variables u and z are orphan, because they occur only once in q at a non-primary-key position. The variable x is neither liaison nor orphan. The complex part of q is $[\![q]\!] = \{R(\underline{x}, y), T(\underline{y}, u, a)\}$.

The complex part of a query q can be empty, as is the case for $q = \{R(\underline{x, y}), S(\underline{x}, u), T(\underline{y}, w)\}$, in which each position of R is a primary-key position. Lemma 3 implies that if the complex part of an SJFCQ query q is empty, then the problem σCERTAINTY(q) is tractable.

[2] Liaison variables are sometimes called join variables in the literature.

Algorithm $Eval(\mathbf{db}, \mu, q)$
Input: multibase $(\mathbf{db}, \mu)$, safe SJFCQ query q
Output: $\sigma\mathsf{frac}(\mathbf{db}, \mu, q)$

(∗ Evaluation SE0a ∗)
if $q = \{g\}$ with $\mathsf{Vars}(g) = \emptyset$
  then if $g \in \mathbf{db}$
    then return $\mu(g)/\sigma\mathsf{block}(g, \mathbf{db}, \mu)$
    else return 0

(∗ Evaluation SE0b ∗)
if $[\![q]\!] = \emptyset$
  then if $\mathbf{db} \models q$
    then return 1
    else return 0

(∗ Evaluation SE1 ∗)
if $q = q_1 \uplus q_2$ **and** $q_1 \neq \emptyset \neq q_2$ **and** $\mathsf{Vars}(q_1) \cap \mathsf{Vars}(q_2) = \emptyset$
  then return $Eval(\mathbf{db}, \mu, q_1) \times Eval(\mathbf{db}, \mu, q_2)$

(∗ Evaluation SE2 ∗)
if $[\![q]\!] \neq \emptyset$ **and** $\exists x \forall g \in [\![q]\!](x \in \mathsf{KVars}(g))$
  then return $1 - \left(\prod_{a \in \mathbf{adom(db)}} \left(1 - Eval(\mathbf{db}, \mu, q_{x \mapsto a})\right)\right)$

(∗ Evaluation SE3 ∗)
if $\exists x \exists g \in q(\mathsf{KVars}(g) = \emptyset, x \in \mathsf{Vars}(g))$
  then return $\sum_{a \in \mathbf{adom(db)}} Eval(\mathbf{db}, \mu, q_{x \mapsto a})$

Fig. 6. Algorithm *Eval*

Lemma 3 (SE0b). *Let q be an SJFCQ query. If $[\![q]\!] = \emptyset$, then for every multibase $(\mathbf{db}, \mu)$,*

$$\sigma\mathsf{frac}(\mathbf{db}, \mu, q) = \begin{cases} 0 \text{ if } \mathbf{db} \not\models q \\ 1 \text{ if } \mathbf{db} \models q \end{cases}$$

Proof. Straightforward.

Lemma 4 deals with queries q that can be partitioned into two subqueries, say q_1 and q_2, such that q_1 and q_2 have no variables in common. The lemma implies that if $\sigma\mathsf{CERTAINTY}(q_1)$ and $\sigma\mathsf{CERTAINTY}(q_2)$ are both tractable, then so is $\sigma\mathsf{CERTAINTY}(q)$.

Lemma 4 (SE1). *Let q, q_1, q_2 be SJFCQ queries such that $q = q_1 \cup q_2$, $q_1 \cap q_2 = \emptyset$, and $\mathsf{Vars}(q_1) \cap \mathsf{Vars}(q_2) = \emptyset$. Then, for every multibase $(\mathbf{db}, \mu)$,*

$$\sigma\mathsf{frac}(\mathbf{db}, \mu, q) = \sigma\mathsf{frac}(\mathbf{db}, \mu, q_1) \times \sigma\mathsf{frac}(\mathbf{db}, \mu, q_2).$$

Lemma 5 treats queries q for which there exists a variable x such that x occurs at a primary-key position in every goal of the complex part of q. An example is the query $\{R(\underline{x}, y), S(\underline{x, y}, z), T(\underline{z}, u)\}$, whose complex part is $\{R(\underline{x}, y),$

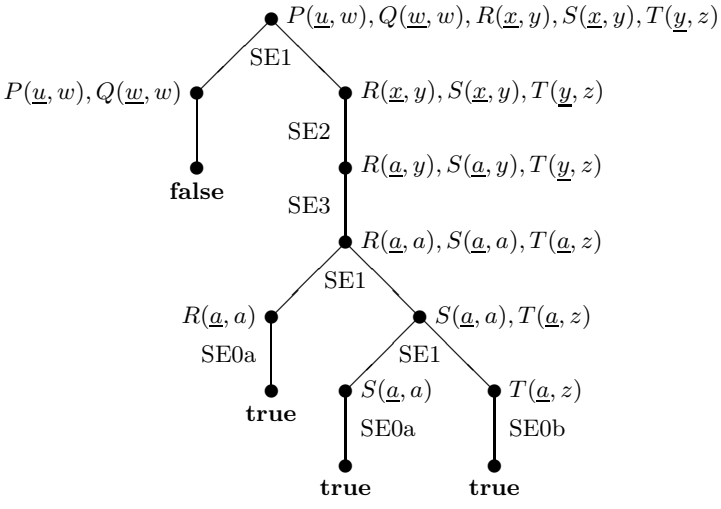

Fig. 7. Tree representation of an execution of *IsSafe*. Vertices are labeled by queries. Each edge label indicates the rule that, applied on the parent, results in the children. Since some leaf vertex is **false**, the query at the root node is unsafe. Notice that the subquery $R(\underline{x}, y), S(\underline{x}, y), T(\underline{y}, z)$ is safe, because all its descendant leaf vertices are **true**.

$S(\underline{x, y}, z)\}$; the variable x occurs at a primary-key position in each goal of the complex part.

Lemma 5 (SE2). *Let q be an SJFCQ query such that $[\![q]\!] \neq \emptyset$ and for some variable x, $x \in \bigcap_{g \in [\![q]\!]} \mathsf{KVars}(g)$. Then, for every multibase $(\mathbf{db}, \mu)$,*

$$\sigma\mathsf{frac}(\mathbf{db}, \mu, q) = 1 - \Big(\prod_{a \in \mathbf{adom}(\mathbf{db})} \big(1 - \sigma\mathsf{frac}(\mathbf{db}, \mu, q_{x \mapsto a})\big)\Big).$$

Lemma 6 treats queries with a goal g such that all primary-key positions in g are occupied by constants, and at least one non-primary-key position is occupied by a variable.

Lemma 6 (SE3). *Let q be an SJFCQ query such that for some goal $g \in q$, for some variable x, $\mathsf{KVars}(g) = \emptyset$ and $x \in \mathsf{Vars}(g)$. Then, for every multibase $(\mathbf{db}, \mu)$,*

$$\sigma\mathsf{frac}(\mathbf{db}, \mu, q) = \sum_{a \in \mathbf{adom}(\mathbf{db})} \sigma\mathsf{frac}(\mathbf{db}, \mu, q_{x \mapsto a}).$$

Lemmas 2–6 are now bundled in a recursive algorithm, called *IsSafe*, which takes as input an SJFCQ query q and returns **true** if for every multibase $(\mathbf{db}, \mu)$,

$\sigma\mathsf{frac}(\mathbf{db}, \mu, q)$ can be obtained by recursive application of Lemmas 2–6; otherwise *IsSafe* returns **false**. Algorithm *IsSafe* is shown in Figure 5. To simplify the notation, we write $A \uplus B$ for the *disjoint union* of sets A and B, where A and B are understood to be disjoint. An execution of *IsSafe* is illustrated in Figure 7. Algorithm *IsSafe* always terminates since every recursive call has an argument query that contains either less goals or less variables.

Definition 2. *An* SJFCQ *query q is called* safe *if the algorithm IsSafe returns* ***true*** *on input q; if IsSafe returns* ***false****, then q is called* unsafe.

Figure 6 shows algorithm *Eval*, which takes as input a multibase $(\mathbf{db}, \mu)$ and a safe SJFCQ query q, and returns $\sigma\mathsf{frac}(\mathbf{db}, \mu, q)$ in polynomial time in the size of $\mathbf{db}$.

Theorem 1. *If q is a safe* SJFCQ *query, then*

1. *for each multibase $(\mathbf{db}, \mu)$, algorithm Eval correctly computes $\sigma\mathsf{frac}(\mathbf{db}, \mu, q)$;*
2. *σ*CERTAINTY*(q) is in $\mathbf{P}$.*

Finally, we show that $\sigma\mathsf{CERTAINTY}(q)$ is $\natural\mathbf{P}$-hard (and hence highly intractable) for queries q that violate safety.

Theorem 2. *For every unsafe* SJFCQ *query q, the problem $\sigma\mathsf{CERTAINTY}(q)$ is $\natural\mathbf{P}$-hard under polynomial-time Turing reductions.*

Proof. From [10], it follows that $\sigma\mathsf{CERTAINTY}(q)$ is already $\natural\mathbf{P}$-hard if all multiplicities are 1.

Consequently, given an SJFCQ query q, we can test by means of algorithm *IsSafe* whether q is safe. If q is not safe, then computing $\sigma\mathsf{frac}(\mathbf{db}, \mu, q)$ for a given database $(\mathbf{db}, \mu)$ has highly intractable data complexity, unless $\mathbf{P} = \mathbf{NP}$. On the other hand, if q is safe, then algorithm *Eval* computes $\sigma\mathsf{frac}(\mathbf{db}, \mu, q)$ in polynomial time in the size of $(\mathbf{db}, \mu)$.

5 Conclusion

Uncertainty is modeled in our data model by means of primary key violations. Each database fact g has a multiplicity, indicating how often g occurs in the database. We have argued that multiplicities may be useful in practice.

For a Boolean query q, the problem $\sigma\mathsf{CERTAINTY}(q)$ asks the weighted number of repairs (or possible worlds) in which q evaluates to true. Intuitively, a query q is more certain if it is true in more repairs. We have given a sound and complete syntactic characterization of the self-join-free conjunctive queries q for which $\sigma\mathsf{CERTAINTY}(q)$ can be solved in polynomial time (data complexity).

An open problem is to determine tractability boundaries for richer query languages, like unions of conjunctive queries or conjunctive queries with self-joins.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: PODS, pp. 68–79. ACM Press, New York (1999)
3. Arenas, M., Bertossi, L.E., Chomicki, J., He, X., Raghavan, V., Spinrad, J.: Scalar aggregation in inconsistent databases. Theor. Comput. Sci. 296(3), 405–434 (2003)
4. Dalvi, N.N., Ré, C., Suciu, D.: Probabilistic databases: diamonds in the dirt. Commun. ACM 52(7), 86–94 (2009)
5. Dalvi, N.N., Re, C., Suciu, D.: Queries and materialized views on probabilistic databases. J. Comput. Syst. Sci. 77(3), 473–490 (2011)
6. Dalvi, N.N., Suciu, D.: Management of probabilistic data: foundations and challenges. In: Libkin, L. (ed.) PODS, pp. 1–12. ACM, New York (2007)
7. Fan, W., Geerts, F., Wijsen, J.: Determining the currency of data. In: Lenzerini, M., Schwentick, T. (eds.) PODS, pp. 71–82. ACM, New York (2011)
8. Fuxman, A., Miller, R.J.: First-order query rewriting for inconsistent databases. J. Comput. Syst. Sci. 73(4), 610–635 (2007)
9. Greco, S., Molinaro, C.: Approximate probabilistic query answering over inconsistent databases. In: Li, Q., Spaccapietra, S., Yu, E.S.K., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 311–325. Springer, Heidelberg (2008)
10. Maslowski, D., Wijsen, J.: On counting database repairs. In: Proceedings of the 4th International Workshop on Logic in Databases, LID 2011, pp. 15–22. ACM, New York (2011), http://doi.acm.org/10.1145/1966357.1966361
11. Pema, E., Kolaitis, P.G., Tan, W.C.: On the tractability and intractability of consistent conjunctive query answering. In: Proceedings of the 2011 Joint EDBT/ICDT Ph.D. Workshop, PhD 2011, pp. 38–44. ACM, New York (2011), http://doi.acm.org/10.1145/1966874.1966881
12. Toda, S.: PP is as hard as the polynomial-time hierarchy. SIAM J. Comput. 20(5), 865–877 (1991)
13. Wijsen, J.: On the consistent rewriting of conjunctive queries under primary key constraints. Inf. Syst. 34(7), 578–601 (2009)
14. Wijsen, J.: On the first-order expressibility of computing certain answers to conjunctive queries over uncertain databases. In: Paredaens, J., Gucht, D.V. (eds.) PODS, pp. 179–190. ACM, New York (2010)
15. Wijsen, J.: A remark on the complexity of consistent conjunctive query answering under primary key violations. Inf. Process. Lett. 110(21), 950–955 (2010)