A Survey of the Data Complexity of Consistent Query Answering under Key Constraints

Jef Wijsen

Université de Mons, Mons, Belgium
jef.wijsen@umons.ac.be

Abstract. This paper adopts a very elementary representation of uncertainty. A relational database is called uncertain if it can violate primary key constraints. A repair of an uncertain database is obtained by selecting a maximal number of tuples without selecting two distinct tuples of the same relation that agree on their primary key. For any Boolean query q, CERTAINTY(q) is the problem that takes an uncertain database **db** on input, and asks whether q is true in every repair of **db**. The complexity of these problems has been particularly studied for q ranging over the class of Boolean conjunctive queries. A research challenge is to solve the following complexity classification task: given q, determine whether CERTAINTY(q) belongs to complexity classes **FO**, **P**, or **coNP**-complete.

The counting variant of CERTAINTY(q), denoted ♯CERTAINTY(q), asks to determine the exact number of repairs that satisfy q. This problem is related to query answering in probabilistic databases.

This paper motivates the problems CERTAINTY(q) and ♯CERTAINTY(q), surveys the progress made in the study of their complexity, and lists open problems. We also show a new result comparing complexity boundaries of both problems with one another.

1 Motivation

Uncertainty shows up in a variety of forms and representations. In this paper, we consider a very elementary representation of uncertainty. We model uncertainty in the relational database model by primary key violations. A *block* is a maximal set of tuples of the same relation that agree on the primary key of that relation. Tuples of a same block are mutually exclusive alternatives for each other. In each block, only one (and exactly one) tuple can be true, but we do not know which one. We will refer to databases as "uncertain databases" to stress that such databases can violate primary key constraints.

Primary keys are underlined in the conference planning database of Fig. 1. Blocks are separated by dashed lines. There is uncertainty about the city of ICDT 2016 (Rome or Paris), about the rank of KDD (A or B), and about the frequency of ICDT (biennial or annual).

There can be several reasons why a database is uncertain. On the positive side, it allows one to represent several possible future scenarios. In Fig. 1, the relation **C** represents that there are still two candidate cities for hosting ICDT

C. Beierle and C. Meghini (Eds.): FoIKS 2014, LNCS 8367, pp. 62–78, 2014.

C

conf	year	city	country
ICDT	2016	Rome	Italy
ICDT	2016	Paris	France
KDD	2017	Rome	Italy

R

conf	rank	frequency
ICDT	A	biennial
ICDT	A	annual
KDD	A	annual
KDD	B	annual
DBPL	B	biennial
BDA	B	annual

Fig. 1. Uncertain database

2016. On the reverse side, inconsistency may be an undesirable but inescapable consequence of data integration. The relation **R** may result from integrating data from different web sites that contradict one another.

A *repair* (or possible world) of an uncertain database is obtained by selecting exactly one tuple from each block. In general, the number of repairs of an uncertain database **db** is exponential in the size of **db**. For instance, if an uncertain database contains n blocks with two tuples each, then it contains $2n$ tuples and has 2^n repairs.

There are three natural semantics for answering Boolean queries q on an uncertain database. Under the *possibility semantics*, the question is whether the query evaluates to true on some repair. Under the *certainty semantics*, the question is whether the query evaluates to true on every repair. More generally, under the *counting semantics*, the question is to determine the number of repairs on which the query evaluates to true. In this paper, we consider the certainty and counting semantics. The certainty semantics adheres to the paradigm of *consistent query answering* [1,3], which introduces the notion of database repair with respect to general integrity constraints. In this work, repairing is exclusively with respect to primary key constraints, one per relation.

Example 1. The uncertain database of Fig. 1 has eight repairs. The Boolean first-order query $\exists x \exists y \exists z \exists w \left(\mathbf{C}(\underline{x, y}, \text{'Rome'}, z) \wedge \mathbf{R}(\underline{x}, \text{'A'}, w)\right)$ (Will Rome host some A conference?) is true in six repairs.

For any Boolean query q, the decision problem $\mathsf{CERTAINTY}(q)$ is the following.

PROBLEM:	$\mathsf{CERTAINTY}(q)$
INPUT:	uncertain database **db**
QUESTION:	Does every repair of **db** satisfy q?

Two comments are in place. First, the Boolean query q is not part of the input. Every Boolean query q gives thus rise to a new problem. Since the input to $\mathsf{CERTAINTY}(q)$ is an uncertain database, the problem complexity is *data complexity*. Second, we will assume that every relation name in q or **db** has a fixed known arity and primary key. The primary key constraints are thus implicitly present in all problems.

The complexity of CERTAINTY(q) has gained considerable research attention in recent years. A challenging question is to distinguish queries q for which the problem CERTAINTY(q) is tractable from queries for which the problem is intractable. Further, if CERTAINTY(q) is tractable, one may ask whether it is first-order definable. We will refer to these questions as the *complexity classification task for* CERTAINTY(q).

For any Boolean query q, the counting problem ♯CERTAINTY(q) is defined as follows.

PROBLEM:	♯CERTAINTY(q)
INPUT:	uncertain database **db**
QUESTION:	How many repairs of **db** satisfy q?

The complexity classification task for ♯CERTAINTY(q) is then to determine the complexity of ♯CERTAINTY(q) for varying q.

In this paper, we review known results in the aforementioned complexity classification tasks. We also contribute a new result relating the complexity classifications for CERTAINTY(q) and ♯CERTAINTY(q). We discuss variations and extensions of the basic problems, and review existing systems that implement algorithms for consistent query answering under primary keys.

This paper is organized as follows. Section 2 introduces the basic concepts and terminology. Section 3 discusses *consistent first-order rewriting*, which consists in solving CERTAINTY(q) in first-order logic. Section 4 reviews known dichotomy theorems for CERTAINTY(q) and ♯CERTAINTY(q). Section 5 contains our new result. From Section 6 on, we present a number of variations and extensions of the basic framework. Section 6 introduces the notion of *nucleus* of an uncertain database **db** relative to a class $\mathcal{C}$ of Boolean queries. Intuitively, a nucleus is a new (consistent) database that "summarizes" all repairs of **db** such that it returns certain answers to all queries in $\mathcal{C}$. Section 7 relates ♯CERTAINTY(q) to query evaluation in probabilistic databases. Section 9 discusses practical implementations. Finally, Section 10 lists some questions for future research.

2 Preliminaries

In this section, we first introduce basic notions and terminology. We then recall a number of complexity classes that will occur in the complexity classification tasks mentioned in Section 1.

2.1 Data and Query Model

We assume disjoint sets of *variables* and *constants*. If $\boldsymbol{x}$ is a sequence containing variables and constants, then vars($\boldsymbol{x}$) denotes the set of variables that occur in $\boldsymbol{x}$. A *valuation* over a set U of variables is a total mapping θ from U to the set of constants. Such a valuation θ is extended to be the identity on constants and on variables not in U.

Atoms and Key-Equal Facts. 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 $[n, k]$.

If R is a relation name with signature $[n, k]$, then $R(s_1, \ldots, s_n)$ is called an *R-atom* (or simply atom), where each s_i is either a constant or a variable ($1 \leq i \leq n$). Such an atom is commonly written as $R(\underline{\boldsymbol{x}}, \boldsymbol{y})$ where the primary key value $\boldsymbol{x} = s_1, \ldots, s_k$ is underlined and $\boldsymbol{y} = s_{k+1}, \ldots, s_n$. A *fact* is an atom in which no variable occurs. Two facts $R_1(\underline{\boldsymbol{a}_1}, \boldsymbol{b}_1), R_2(\underline{\boldsymbol{a}_2}, \boldsymbol{b}_2)$ are *key-equal* if $R_1 = R_2$ and $\boldsymbol{a}_1 = \boldsymbol{a}_2$.

We will use letters F, G, H for atoms. For an atom $F = R(\underline{\boldsymbol{x}}, \boldsymbol{y})$, we denote by $\mathsf{key}(F)$ the set of variables that occur in $\boldsymbol{x}$, and by $\mathsf{vars}(F)$ the set of variables that occur in F, that is, $\mathsf{key}(F) = \mathsf{vars}(\boldsymbol{x})$ and $\mathsf{vars}(F) = \mathsf{vars}(\boldsymbol{x}) \cup \mathsf{vars}(\boldsymbol{y})$.

Uncertain Database, Blocks, and Repairs. A *database schema* is a finite set of relation names. All constructs that follow are defined relative to a fixed database schema.

An *uncertain database* is a finite set $\mathbf{db}$ of facts using only the relation names of the schema. We write $\mathbf{adom}(\mathbf{db})$ for the active domain of $\mathbf{db}$ (i.e., the set of constants that occur in $\mathbf{db}$). A *block* of $\mathbf{db}$ is a maximal set of key-equal facts of $\mathbf{db}$. An uncertain database $\mathbf{db}$ is *consistent* if it does not contain two distinct facts that are key-equal (i.e., if every block of $\mathbf{db}$ is a singleton). A *repair* of $\mathbf{db}$ is a maximal (with respect to set containment) consistent subset of $\mathbf{db}$.

Boolean Conjunctive Query. A *Boolean query* is a mapping q that associates a Boolean (true or false) to each uncertain database, such that q is closed under isomorphism [22]. We write $\mathbf{db} \models q$ to denote that q associates true to $\mathbf{db}$, in which case $\mathbf{db}$ is said to *satisfy* q. A *Boolean first-order query* is a Boolean query that can be defined in first-order logic. A *Boolean conjunctive query* is a finite set $q = \{R_1(\underline{\boldsymbol{x}_1}, \boldsymbol{y}_1), \ldots, R_n(\underline{\boldsymbol{x}_n}, \boldsymbol{y}_n)\}$ of atoms. By $\mathsf{vars}(q)$, we denote the set of variables that occur in q. The set q represents the first-order sentence

$$\exists u_1 \cdots \exists u_k \left(R_1(\underline{\boldsymbol{x}_1}, \boldsymbol{y}_1) \wedge \cdots \wedge R_n(\underline{\boldsymbol{x}_n}, \boldsymbol{y}_n) \right),$$

where $\{u_1, \ldots, u_k\} = \mathsf{vars}(q)$. This query q is satisfied by uncertain database $\mathbf{db}$ if there exists a valuation θ over $\mathsf{vars}(q)$ such that for each $i \in \{1, \ldots, n\}$, $R_i(\underline{\boldsymbol{a}}, \boldsymbol{b}) \in \mathbf{db}$ with $\boldsymbol{a} = \theta(\boldsymbol{x}_i)$ and $\boldsymbol{b} = \theta(\boldsymbol{y}_i)$.

If q is a Boolean conjunctive query, $\boldsymbol{x} = \langle x_1, \ldots, x_\ell \rangle$ is a sequence of distinct variables that occur in q, and $\boldsymbol{a} = \langle a_1, \ldots, a_\ell \rangle$ is a sequence of constants, then $q_{[\boldsymbol{x} \mapsto \boldsymbol{a}]}$ denotes the query obtained from q by replacing all occurrences of x_i with a_i, for all $1 \leq i \leq \ell$.

Computational Problems. The decision problem $\mathsf{CERTAINTY}(q)$ and the counting problem $\sharp\mathsf{CERTAINTY}(q)$ have been defined in Section 1.

2.2 Restrictions on Conjunctive Queries

The class of Boolean conjunctive queries can be further restricted by adding syntactic constraints.

Acyclicity. A Boolean conjunctive query q is *acyclic* if it has a *join tree* [2]. A *join tree* for q is an undirected tree whose vertices are the atoms of q such that for every variable x in $\mathsf{vars}(q)$, the set of vertices in which x occurs induces a connected subtree.

No Self-joins. We say that a Boolean conjunctive query q has a *self-join* if some relation name occurs more than once in q. If q has no self-join, then it is called *self-join-free*.

Restrictions on Signatures. Let R be a relation name with signature $[n, k]$. The relation name R is *simple-key* if $k = 1$. The relation name R is *all-key* if $n = k$.

We introduce names for some classes of special interest:

- BCQ denotes the class of Boolean conjunctive queries;
- SJFBCQ denotes the class of self-join-free Boolean conjunctive queries; and
- ACYSJFBCQ denotes the class of acyclic self-join-free Boolean conjunctive queries.

2.3 Complexity Classes

The following complexity classes will occur in the complexity classification tasks for $\mathsf{CERTAINTY}(q)$ and $\sharp\mathsf{CERTAINTY}(q)$.

- **FO**, the class of first-order definable problems. In particular, for a given Boolean query q, $\mathsf{CERTAINTY}(q)$ is in **FO** if there exists a Boolean first-order query φ such that for every uncertain database **db**, every repair of **db** satisfies q if and only if **db** satisfies φ. Such a φ, if it exists, is called a *consistent first-order rewriting of* q.
- **P**, the class of decision problems that can be solved in deterministic polynomial time.
- **NP**, the class of decision problems whose "yes" instances have succinct certificates that can be verified in deterministic polynomial time.
- **coNP**, the class of decision problems whose "no" instances have succinct disqualifications that can be verified in deterministic polynomial time. In particular, $\mathsf{CERTAINTY}(q)$ is in **coNP** for every Boolean first-order query q, because if q is not true in every repair of **db**, then a succinct disqualification is a repair of **db** that falsifies q. Indeed, repair checking (i.e., given **rep** and **db**, check whether **rep** is a repair of **db**) is in polynomial time, and so is the data complexity of first-order queries.

- **FP**, the class of function problems that can be solved in deterministic polynomial time. In particular, for a given Boolean query q, $\sharp\mathsf{CERTAINTY}(q)$ is in **FP** if there exists a polynomial-time algorithm that takes any uncertain database **db** on input, and returns the number of repairs of **db** that satisfy q.
- $\sharp$**P**, the class of counting problems associated with decision problems in **NP**. Given an instance of a decision problem in **NP**, the associated counting problem instance asks to determine the number of succinct certificates of its being a "yes" instance. The following problem is obviously in **NP** for every Boolean first-order query q: given an uncertain database **db** on input, determine whether some repair of **db** satisfies q. Its associated counting problem, called $\sharp\mathsf{CERTAINTY}(q)$, is thus in $\sharp$**P**.

Concerning the latter item, notice that the decision variant of $\sharp\mathsf{CERTAINTY}(q)$ is *not* $\mathsf{CERTAINTY}(q)$, which is a decision problem in **coNP**. The problem $\sharp\mathsf{CERTAINTY}(q)$ might better have been named $\sharp\mathsf{POSSIBILITY}(q)$ or so, because its decision variant asks, given an uncertain database on input, whether some repair satisfies q.

3 Consistent First-Order Rewriting

The detailed investigation of $\mathsf{CERTAINTY}(q)$ was pioneered by Fuxman and Miller [14,15]. This initial research focused on determining classes of Boolean conjunctive queries q for which $\mathsf{CERTAINTY}(q)$ is in **FO**, and hence solvable by a single Boolean first-order query, which is called a consistent first-order rewriting of q. The practical significance is that such a consistent first-order rewriting can be directly implemented in SQL. A concrete example is given next.

Example 2. Let q_0 be the query $\exists z\mathbf{R}(\text{'ICDT'}, \text{'A'}, z)$ (Is ICDT a conference of rank A?). Clearly, q_0 is true in every repair of some uncertain database **db** if and only if **db** contains an **R**-fact stating that ICDT has rank A, and contains no **R**-fact mentioning a different rank for ICDT. These conditions are expressed by the following Boolean first-order query (call it φ_0):

$$\exists z\mathbf{R}(\text{'ICDT'}, \text{'A'}, z) \wedge \forall y \forall z \left(\mathbf{R}(\text{'ICDT'}, y, z) \rightarrow y = \text{'A'}\right).$$

If φ_0 evaluates to true on **db**, then every repair of **db** satisfies q_0; if φ_0 evaluates to false on **db**, then some repair of **db** falsifies q_0. Thus, φ_0 a consistent first-order rewriting of q_0, and solves $\mathsf{CERTAINTY}(q_0)$ without any need for enumerating repairs.

Fuxman and Miller defined a class $\mathcal{C}_{forest} \subseteq$ SjfBcq such that for every $q \in \mathcal{C}_{forest}$, $\mathsf{CERTAINTY}(q)$ is in **FO**. Their results were improved by Wijsen [34], as follows.

Theorem 1 ([34]). *Given $q \in$ AcySjfBcq, it is decidable (in quadratic time in the size of q) whether $\mathsf{CERTAINTY}(q)$ is in* **FO**. *If $\mathsf{CERTAINTY}(q)$ is in* **FO**, *then a consistent first-order rewriting of q can be effectively constructed.*

To be precise, the class $\mathcal{C}_{forest}$ contains some conjunctive queries that are cyclic in some very restricted way (see [9, Theorem 2.4]). On the other hand, there exist queries $q \in$ AcySjfBcq such that CERTAINTY(q) is in **FO** but $q \notin \mathcal{C}_{forest}$. For Boolean conjunctive queries with self-joins, some sufficient conditions for the existence of consistent first-order rewritings appear in [32].

4 Complexity Dichotomy Theorems

A further research challenge is to distinguish Boolean conjunctive queries q for which the problem CERTAINTY(q) is tractable from queries for which the problem is intractable.

In general, we say that a class $\mathcal{P}$ of decision problems *exhibits a* **P-coNP**-*dichotomy* if all problems in $\mathcal{P}$ are either in **P** or **coNP**-hard. We say that $\mathcal{P}$ *exhibits an effective* **P-coNP**-*dichotomy* if in addition it is decidable whether a given problem in $\mathcal{P}$ is in **P** or **coNP**-hard. Likewise, we say that a class $\mathcal{P}$ of function problems *exhibits an* **FP**-♯**P**-*dichotomy* if all problems in $\mathcal{P}$ are either in **FP** or ♯**P**-hard (under polynomial-time Turing reductions). We say that $\mathcal{P}$ *exhibits an effective* **FP**-♯**P**-*dichotomy* if in addition it is decidable whether a given problem in $\mathcal{P}$ is in **FP** or ♯**P**-hard. We use the term *complexity dichotomy theorem* to refer to a theorem that establishes a **P-coNP**-dichotomy in a class of decision problems, or an **FP**-♯**P**-dichotomy in a class of counting problems. By Ladner's theorem [21], if **P** $\neq$ **NP**, or **FP** $\neq$ ♯**P**, then no such complexity dichotomy theorems exist for **coNP** or ♯**P**.

Let $\mathcal{C}$ be a class of Boolean queries. We write CERTAINTY[$\mathcal{C}$] to denote the class of decision problems that contains CERTAINTY(q) for every $q \in \mathcal{C}$. Likewise, we write ♯CERTAINTY[$\mathcal{C}$] for the class of counting problems that contains ♯CERTAINTY(q) for every $q \in \mathcal{C}$.

The following result by Kolaitis and Pema was chronologically the first complexity dichotomy theorem for consistent query answering under primary keys.

Theorem 2 ([17]). *Let $\mathcal{C}$ be the class of self-join-free Boolean conjunctive queries that contain at most two atoms. Then,* CERTAINTY[$\mathcal{C}$] *exhibits an effective* **P-coNP**-*dichotomy.*

Clearly, every Boolean conjunctive query with at most two atoms, is acyclic. The following generalization of Theorem 2 was conjectured in both [26] and [35]. What is remarkable is that both works have independently conjectured exactly the same boundary between tractable and intractable problems in the class CERTAINTY[AcySjfBcq]. The exposition of this boundary is involved and will not be given here.

Conjecture 1. The problem class CERTAINTY[AcySjfBcq] exhibits an effective **P-coNP**-dichotomy.

Recently, Koutris and Suciu showed the following complexity dichotomy theorem.

Theorem 3 ([19,20]). *Let $\mathcal{C}$ be the class of self-join-free Boolean conjunctive queries in which each relation name is either simple-key or all-key. Then,* $\mathsf{CERTAINTY}[\mathcal{C}]$ *exhibits an effective* **P**-**coNP**-*dichotomy.*

In summary, for Boolean conjunctive queries q, the complexity classification task for $\mathsf{CERTAINTY}(q)$ is far from accomplished. More is known about the complexity classification of $\sharp\mathsf{CERTAINTY}(q)$, as becomes clear from the following two theorems.

Theorem 4 ([24]). *The problem class* $\sharp\mathsf{CERTAINTY}[\textsc{SjfBcq}]$ *exhibits an effective* **FP**-$\sharp$**P**-*dichotomy.*

The following is one of the few complexity dichotomies known for conjunctive queries with self-joins.

Theorem 5 ([25]). *Let $\mathcal{C}$ be the class of Boolean conjunctive queries in which all relation names are simple-key. Then,* $\sharp\mathsf{CERTAINTY}[\mathcal{C}]$ *exhibits an effective* **FP**-$\sharp$**P**-*dichotomy.*

5 Comparing Complexity Boundaries

Theorems 1-5 contribute to the complexity classification tasks for $\mathsf{CERTAINTY}(q)$ and $\sharp\mathsf{CERTAINTY}(q)$. These theorems guarantee the existence of effective procedures to classify problems in different complexity classes. We will not expose these effective procedures in detail, but provide some examples for queries in SjfBcq.

1. For $q_1 = \{R(\underline{x}, y), S(\underline{y}, z)\}$, we have that $\mathsf{CERTAINTY}(q_1)$ is in **FO**, and $\sharp\mathsf{CERTAINTY}(q_1)$ is in **FP**. A consistent first-order rewriting of q_1 is $\exists x \exists y \left(R(\underline{x}, y) \wedge \forall y \left(R(\underline{x}, y) \rightarrow \exists z S(\underline{y}, z)\right)\right)$.
2. For $q_2 = \{R(\underline{x}, y), S(\underline{y}, a)\}$, where a is a constant, $\mathsf{CERTAINTY}(q_2)$ is in **FO**, but $\sharp\mathsf{CERTAINTY}(q_2)$ is already $\sharp$**P**-hard. A consistent first-order rewriting of q_2 is $\exists x \exists y \left(R(\underline{x}, y) \wedge \forall y \left(R(\underline{x}, y) \rightarrow \left(S(\underline{y}, a) \wedge \forall z \left(S(\underline{y}, z) \rightarrow z = a\right)\right)\right)\right)$.
3. For $q_3 = \{R(\underline{x}, y), S(\underline{y}, x)\}$, we have that $\mathsf{CERTAINTY}(q_3)$ is in $\mathbf{P} \setminus \mathbf{FO}$ [33], and $\sharp\mathsf{CERTAINTY}(q_3)$ is $\sharp$**P**-hard.
4. For $q_4 = \{R(\underline{x}, y), S(\underline{z}, y)\}$, we have that $\mathsf{CERTAINTY}(q_4)$ is **coNP**-complete, and $\sharp\mathsf{CERTAINTY}(q_4)$ is $\sharp$**P**-hard.

We have no example of a query $q \in \textsc{SjfBcq}$ such that $\mathsf{CERTAINTY}(q)$ is in $\mathbf{P} \setminus \mathbf{FO}$ and $\sharp\mathsf{CERTAINTY}(q)$ is in **FP**. The following theorem implies that no such query exists unless $\mathbf{FP} = \sharp\mathbf{P}$. The proof is in the Appendix.

Theorem 6. *For every $q \in \textsc{SjfBcq}$, if* $\mathsf{CERTAINTY}(q)$ *is not in* **FO**, *then* $\sharp\mathsf{CERTAINTY}(q)$ *is* $\sharp$**P**-*hard.*

Clearly, the number of repairs of an uncertain database **db** can be computed in polynomial time in the size of **db**, by multiplying the cardinalities of all blocks. Therefore, for every Boolean query q, if $\sharp\mathsf{CERTAINTY}(q)$ is in **FP**, then

$\mathbf{R}'$	conf	rank	frequency
	ICDT	A	ℓ_1
	KDD	ℓ_2	annual
	DBPL	B	biennial
	BDA	B	annual
	ℓ_3	B	ℓ_1
	ℓ_4	ℓ_2	ℓ_1

Fig. 2. BCQ-nucleus for the relation **R** of Fig. 1. The symbols ℓ_1, ℓ_2, ℓ_3, ℓ_4 are new distinct constants that cannot be used in queries.

CERTAINTY(q) must be in **P**. That is, tractability of CERTAINTY(q) could in principle be established from tractability of ♯CERTAINTY(q). We notice, however, that Theorem 4 does not give us new tractable cases of CERTAINTY(q) in this way for $q \in$ ACYSJFBCQ (i.e., Theorem 4 does not help to prove Conjecture 1). By Theorem 1, we can already distinguish in ACYSJFBCQ the queries q that have a consistent first-order rewriting from those that have not. If a query $q \in$ ACYSJFBCQ has no consistent first-order rewriting, then ♯CERTAINTY(q) is ♯**P**-hard by Theorem 6.

6 Nucleus

Solving CERTAINTY(q) consists in developing an algorithm that takes any uncertain database **db** on input, and checks whether every repair of **db** satisfies the Boolean query q. As indicated in Section 3, for some q, such an algorithm can be expressed in first-order logic. A natural question is whether algorithms for solving CERTAINTY(q) could benefit from some database preprocessing. An approach proposed in [29] consists in "rewriting" an uncertain database **db** into a new database $\mathbf{db}'$ such that for all queries q in some query class, the answer to CERTAINTY(q) on input **db** is obtained by executing q on $\mathbf{db}'$. This is formalized next.

Nucleus. Let $\mathcal{C}$ be a class of Boolean queries. A *$\mathcal{C}$-nucleus* of an uncertain database **db** is a database $\mathbf{db}'$ such that for every query $q \in \mathcal{C}$, every repair of **db** satisfies q if and only if $\mathbf{db}'$ satisfies q.

It follows from [29] that every uncertain database **db** has a BCQ-nucleus. To be precise, this result supposes the existence of some special constants, called *labeled nulls*, that can occur in uncertain databases, but not in queries.

Example 3. A BCQ-nucleus for the relation **R** of Fig. 1 is shown in Fig. 2, where ℓ_1, ℓ_2, ℓ_3, ℓ_4 are distinct labeled nulls. The atom $\mathbf{R}'(\underline{\ell_3}, \mathrm{B}, \ell_1)$, for example, expresses that in every repair, some conference of rank B has the same frequency as ICDT. The query $\exists z \mathbf{R}(\text{'ICDT'}, \text{'A'}, z)$ evaluates to true on every repair of **R**,

R	conf	rank	frequency	P
	ICDT	A	biennial	0.3
	ICDT	A	annual	0.6
	KDD	A	annual	0.5
	KDD	B	annual	0.5
	DBPL	B	biennial	0.7
	BDA	B	annual	1.0

Fig. 3. Representation of a BID probabilistic database

and evaluates to true on the BCQ-nucleus. The query $\exists y\mathbf{R}(\text{'ICDT'}, y, \text{'annual'})$ evaluates to false on some repair of $\mathbf{R}$, and evaluates to false on the BCQ-nucleus.

The notion of $\mathcal{C}$-nucleus is closely related to the notion of *universal repair* in [28]. Clearly, since $\mathsf{CERTAINTY}(q)$ is **coNP**-hard for some $q \in \text{BCQ}$, any algorithm that takes an uncertain database **db** on input and computes a BCQ-nucleus of **db**, must be exponential-time (unless $\mathbf{P} = \mathbf{coNP}$).

7 Probabilistic Databases

For any fixed Boolean query q, the problem $\sharp\mathsf{CERTAINTY}(q)$ is a special case of probabilistic query answering. Let N be the total number of repairs of a given uncertain database **db**. If a fact A (or, by extension, a Boolean query) evaluates to true in m repairs, then its probability, denoted $P(A)$, is m/N. For example, in Fig. 1, the probability of the fact $\mathbf{R}(\underline{\text{ICDT}}, \text{A}, \text{biennial})$ is $4/8$, because it belongs to 4 repairs out of 8. It can now be easily verified that for all distinct facts A, B of **db**, the following hold:

- If the facts A and B belong to a same block, then $P(A \wedge B) = 0$. In probabilistic terms, distinct facts of the same block represent *disjoint* (i.e., exclusive) events.
- If the facts A and B belong to distinct blocks, then $P(A \wedge B) = P(A) \cdot P(B)$. In probabilistic terms, facts of distinct blocks are *independent*.

Probabilistic databases satisfying the above two properties have been coined *block-independent-disjoint* (BID) by Dalvi, Ré, and Suciu [6]. BID probabilistic databases can be represented by listing the probability of each fact, as illustrated in Fig. 3. The main differences between uncertain databases and BID probabilistic databases are twofold:

- In an uncertain database, all facts of a same block have the same probability. In BID probabilistic databases, facts of a same block need not have the same probability. For example, in the BID probabilistic database of Fig. 3, the two facts about ICDT have distinct probabilities (0.3 and 0.6).

– In an uncertain database, the probabilities of facts in a same block sum up to 1. In BID probabilistic databases, this sum can be strictly less than 1. The BID probabilistic database of Fig. 3 admits a possible world with non-zero probability in which ICDT does not occur.

A detailed comparison of both data models can be found in [35]. The difference between both data models is further diminished in [23], where some positive integer multiplicity is associated to every tuple of an uncertain database.

The tractability/intractability frontier of evaluating SjfBcq queries on BID probabilistic databases has been revealed by Dalvi et al. [7]. Theorem 4 settles this frontier for uncertain databases. For conjunctive queries with self-joins, no analogue of Theorem 5 is currently known for BID probabilistic databases.

The situation is different for tuple-independent probabilistic databases. In such a database, there is no notion of block and all tuples represent independent events. The tractability/intractability frontier of evaluating unions of conjunctive queries (possibly with self-joins) on tuple-independent probabilistic databases has been revealed by Dalvi and Suciu [8].

8 Integrity Constraints on Uncertain Databases

Integrity constraints allow to restrict the set of legal databases. Although uncertainty is modeled by primary key violations in our approach, this does not mean that constraints should be given up altogether. Some constraints, including some primary keys, could still be enforced.

Example 4. The uncertain database of Fig. 1 satisfies the functional dependency $\mathbf{R} : \mathbf{city} \to \mathbf{country}$, the inclusion dependency $\mathbf{C}[\mathbf{conf}] \subseteq \mathbf{R}[\mathbf{conf}]$, and the join dependency $\mathbf{C} :\bowtie [\{\mathbf{conf}, \mathbf{rank}\}, \{\mathbf{conf}, \mathbf{frequency}\}]$. This join dependency expresses that, given a conference, the uncertainties in rank and frequency are independent [31].

The problem CERTAINTY(q) has been generalized to account for constraints that are satisfied by the uncertain databases that are input to the problem [16], as follows. Let q be a Boolean query, and let Σ be a set of first-order constraints referring exclusively to the relation names in **db**. Then CERTAINTY(q, Σ) is the following decision problem.

Problem:	CERTAINTY(q, Σ)
Input:	uncertain database **db** that satisfies Σ
Question:	Does every repair of **db** satisfy q?

Clearly, if $\Sigma = \emptyset$, then CERTAINTY(q, Σ) is the same as CERTAINTY(q). At another extreme, if Σ contains all primary key constraints, then the input to CERTAINTY(q, Σ) is restricted to consistent databases without primary key violations.

Concerning the following theorem, a join dependency $R :\bowtie [K_1, \ldots, K_\ell]$ is called a *key join dependency* (KJD) if for all $1 \leq i < j \leq \ell$, the intersection

$K_i \cap K_j$ is exactly the primary key of R. The join dependency in Example 4 is a key join dependency.

Theorem 7 ([16]). *Given a query $q \in$ AcySjfBcq and a set Σ of functional dependencies and KJDs containing at most one KJD for every relation name in q, it is decidable whether* CERTAINTY(q, Σ) *is in* **FO**. *If* CERTAINTY(q, Σ) *is in* **FO**, *then a first-order definition of it can be effectively constructed.*

9 Non-boolean Queries and Implemented Systems

In this section, we discuss some systems that have implemented consistent query answering under primary key constraints. In practice, non-Boolean queries are more prevalent than Boolean queries, on which we have focused so far. Nevertheless, most results can be easily extended to non-Boolean queries, as follows.

Henceforth, we will write $q(x_1, \ldots, x_\ell)$ to indicate that q is a (domain independent) first-order query with free variables $x_1, \ldots, x_\ell$. The function problem CERTAINTY$(q(\boldsymbol{x}))$ takes on input an uncertain database **db**, and asks to return all certain answers to q, i.e., all sequences $\boldsymbol{a}$ of constants (of the same length as $\boldsymbol{x}$) such that $q(\boldsymbol{a})$ is true in every repair of **db**. A first-order formula $\varphi(\boldsymbol{x})$ is called a *consistent first-order rewriting of* $q(\boldsymbol{x})$ if for every uncertain database **db**, for all sequences $\boldsymbol{a}$ of constants, $q(\boldsymbol{a})$ is true in every repair of **db** if and only if $\varphi(\boldsymbol{a})$ is true in **db**.

Notice that the number of sequences $\boldsymbol{a}$ that consist exclusively of constants in **db**, is polynomially bounded in the size of **db**. Therefore, the non-Boolean case is at most polynomially more difficult than the Boolean one. Further, $q(\boldsymbol{x})$ has a consistent first-order rewriting if and only if the Boolean query $q_{[\boldsymbol{x} \mapsto \boldsymbol{c}]}$ has a consistent first-order rewriting, where $\boldsymbol{c}$ is a sequence of new constants.

ConQuer [13] and EQUIP [18] are two systems for solving CERTAINTY$(q(\boldsymbol{x}))$ where $q(\boldsymbol{x})$ is a conjunctive query. ConQuer applies to a class of conjunctive queries $q(\boldsymbol{x})$ for which CERTAINTY$(q(\boldsymbol{x}))$ is known to be in **FO**.[1] ConQuer rewrites such a query $q(\boldsymbol{x})$ into a new SQL query Q that gives the certain answers on any uncertain database. The query Q can then be executed in any commercial DBMS. Notice that Q does not depend on the data.

EQUIP applies to all conjunctive queries $q(\boldsymbol{x})$. When an uncertain database **db** is given as the input of the problem CERTAINTY$(q(\boldsymbol{x}))$, EQUIP transforms the database and the query into a Binary Integer Program (BIP) that computes the certain answers. The BIP can then be executed by any existing BIP solver. Since the BIP depends on the database **db**, a new BIP has to be generated whenever the database changes.

Extensive experiments [18,26] show that if CERTAINTY$(q(\boldsymbol{x}))$ is in **FO**, then encoding the problem in SQL (like in ConQuer) is always preferable to binary integer programming. This is not surprising, because binary integer programming is **NP**-hard, while the data complexity of "first-order" SQL is **FO**. A main

[1] ConQuer also deals with aggregation, but we will not consider queries with aggregation here.

conclusion of [26] is that *consistent first-order rewriting should be used whenever possible*. In this respect, Theorems 1 and 7 are of practical importance, because they tell us when exactly consistent first-order rewriting is possible, i.e., when the problem can be solved in SQL. The viability of consistent first-order rewriting was also demonstrated in [10].

10 Open Problems

Notwithstanding active research, the complexity classification of $\mathsf{CERTAINTY}(q)$ for Boolean conjunctive queries q is far from completed. The case of self-joins remains largely unexplored. Furthermore, existing research has exclusively focused on complexity classes **FO**, **P**, and **coNP**-complete. A more fine-grained classification could be pursued. For example, can we characterize Boolean conjunctive queries q for which $\mathsf{CERTAINTY}(q)$ is **P**-complete?

Fontaine [12] has established a number of results relating the complexities of consistent query answering and the constraint satisfaction problem (CSP). One result is the following. Let DisBcq be the class of Boolean first-order queries that can be expressed as disjunctions of Boolean conjunctive queries (possibly with constants and self-joins). Then, a **P-coNP** dichotomy in $\mathsf{CERTAINTY}[\text{DisBcq}]$ implies Bulatov's dichotomy theorem for conservative CSP [4]. Since the proof of the latter theorem is highly involved, it is a major challenge to establish a **P-coNP** dichotomy in $\mathsf{CERTAINTY}[\text{DisBcq}]$. A further natural question is whether complexities in $\sharp\mathsf{CERTAINTY}[\text{DisBcq}]$ can be related to the effective dichotomy theorem for counting CSP proved in [5,11]. Conversely, can complexity results for CSP be used in the complexity classification of $\mathsf{CERTAINTY}(q)$ and $\sharp\mathsf{CERTAINTY}(q)$?

The concept of nucleus has not yet been studied in depth. Can we determine a large class of queries $\mathcal{C} \subseteq \text{Bcq}$ such that a $\mathcal{C}$-nucleus of any uncertain database **db** can be computed in polynomial time in the size of **db**? Some preliminary results appear in [30].

Currently, no dichotomy is known in the complexity of evaluating conjunctive queries with self-joins on BID probabilistic databases. Can the dichotomy of Theorem 5 be extended to BID probabilistic databases?

It remains an open task to gain a better understanding of the role of Σ in the complexity of $\mathsf{CERTAINTY}(q, \Sigma)$. Currently, we have only studied the case where Σ is a set of functional dependencies and join dependencies, the latter of a restricted form. The set Σ of satisfied constraints could be used, for example, to limit the amount of uncertainty by restricting the number of tuples per block.

References

1. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: PODS, pp. 68–79. ACM Press (1999)
2. Beeri, C., Fagin, R., Maier, D., Yannakakis, M.: On the desirability of acyclic database schemes. J. ACM 30(3), 479–513 (1983)

3. Bertossi, L.E.: Database Repairing and Consistent Query Answering. In: Synthesis Lectures on Data Management. Morgan & Claypool Publishers (2011)
4. Bulatov, A.A.: Complexity of conservative constraint satisfaction problems. ACM Trans. Comput. Log. 12(4), 24 (2011)
5. Bulatov, A.A.: The complexity of the counting constraint satisfaction problem. J. ACM 60(5), 34 (2013)
6. Dalvi, N.N., Ré, C., Suciu, D.: Probabilistic databases: diamonds in the dirt. Commun. ACM 52(7), 86–94 (2009)
7. Dalvi, N.N., Re, C., Suciu, D.: Queries and materialized views on probabilistic databases. J. Comput. Syst. Sci. 77(3), 473–490 (2011)
8. Dalvi, N.N., Suciu, D.: The dichotomy of probabilistic inference for unions of conjunctive queries. J. ACM 59(6), 30 (2012)
9. Decan, A.: Certain Query Answering in First-Order Languages. PhD thesis, Université de Mons (2013)
10. Decan, A., Pijcke, F., Wijsen, J.: Certain conjunctive query answering in SQL. In: Hüllermeier, E., Link, S., Fober, T., Seeger, B. (eds.) SUM 2012. LNCS, vol. 7520, pp. 154–167. Springer, Heidelberg (2012)
11. Dyer, M.E., Richerby, D.: An effective dichotomy for the counting constraint satisfaction problem. SIAM J. Comput. 42(3), 1245–1274 (2013)
12. Fontaine, G.: Why is it hard to obtain a dichotomy for consistent query answering? In: LICS, pp. 550–559. IEEE Computer Society (2013)
13. Fuxman, A., Fazli, E., Miller, R.J.: ConQuer: Efficient management of inconsistent databases. In: Özcan, F. (ed.) SIGMOD Conference, pp. 155–166. ACM (2005)
14. Fuxman, A.D., Miller, R.J.: First-order query rewriting for inconsistent databases. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 337–351. Springer, Heidelberg (2005)
15. Fuxman, A., Miller, R.J.: First-order query rewriting for inconsistent databases. J. Comput. Syst. Sci. 73(4), 610–635 (2007)
16. Greco, S., Pijcke, F., Wijsen, J.: Certain query answering in partially consistent databases. PVLDB 7(5) (2014)
17. Kolaitis, P.G., Pema, E.: A dichotomy in the complexity of consistent query answering for queries with two atoms. Inf. Process. Lett. 112(3), 77–85 (2012)
18. Kolaitis, P.G., Pema, E., Tan, W.-C.: Efficient querying of inconsistent databases with binary integer programming. PVLDB 6(6), 397–408 (2013)
19. Koutris, P., Suciu, D.: A dichotomy on the complexity of consistent query answering for atoms with simple keys. CoRR, abs/1212.6636 (2012)
20. Koutris, P., Suciu, D.: A dichotomy on the complexity of consistent query answering for atoms with simple keys. In: Schweikardt [27]
21. Ladner, R.E.: On the structure of polynomial time reducibility. J. ACM 22(1), 155–171 (1975)
22. Libkin, L.: Elements of Finite Model Theory. Springer (2004)
23. Maslowski, D., Wijsen, J.: Uncertainty that counts. In: Christiansen, H., De Tré, G., Yazici, A., Zadrozny, S., Andreasen, T., Larsen, H.L. (eds.) FQAS 2011. LNCS, vol. 7022, pp. 25–36. Springer, Heidelberg (2011)
24. Maslowski, D., Wijsen, J.: A dichotomy in the complexity of counting database repairs. J. Comput. Syst. Sci. 79(6), 958–983 (2013)
25. Maslowski, D., Wijsen, J.: Counting database repairs that satisfy conjunctive queries with self-joins. In: Schweikardt ed. [27]
26. Pema, E.: Consistent Query Answering of Conjunctive Queries under Primary Key Constraints. PhD thesis, University of California Santa Cruz (2013)

27. Schweikardt, N. (ed.): 17th International Conference on Database Theory, ICDT 2014, Athens, Greece, March 24-28. ACM (2014)
28. ten Cate, B., Fontaine, G., Kolaitis, P.G.: On the data complexity of consistent query answering. In: Deutsch, A. (ed.) ICDT, pp. 22–33. ACM (2012)
29. Wijsen, J.: Database repairing using updates. ACM Trans. Database Syst. 30(3), 722–768 (2005)
30. Wijsen, J.: On condensing database repairs obtained by tuple deletions. In: DEXA Workshops, pp. 849–853. IEEE Computer Society (2005)
31. Wijsen, J.: Project-join-repair: An approach to consistent query answering under functional dependencies. In: Larsen, H.L., Pasi, G., Ortiz-Arroyo, D., Andreasen, T., Christiansen, H. (eds.) FQAS 2006. LNCS (LNAI), vol. 4027, pp. 1–12. Springer, Heidelberg (2006)
32. Wijsen, J.: On the consistent rewriting of conjunctive queries under primary key constraints. Inf. Syst. 34(7), 578–601 (2009)
33. Wijsen, J.: A remark on the complexity of consistent conjunctive query answering under primary key violations. Inf. Process. Lett. 110(21), 950–955 (2010)
34. Wijsen, J.: Certain conjunctive query answering in first-order logic. ACM Trans. Database Syst. 37(2), 9 (2012)
35. Wijsen, J.: Charting the tractability frontier of certain conjunctive query answering. In: Hull, R., Fan, W. (eds.) PODS, pp. 189–200. ACM (2013)

A Appendix: Proof of Theorem 6

We first expose the tractability/intractability boundary of Theorem 4.

Complex Part of a Boolean Conjunctive Query. Let q be a Boolean conjunctive query. A variable $x \in \mathsf{vars}(q)$ is called a *liaison variable* if x has at least two occurrences in q.[2] The *complex part* of a Boolean conjunctive query q, denoted $[\![q]\!]$, contains every atom $F \in q$ such that some non-primary-key position in F contains a liaison variable or a constant.

Example 5. The variable y is the only liaison variable in $q = \{R(\underline{x}, y),\ R(\underline{y}, z),\ S(\underline{y}, u, a)\}$, in which a is a constant. The complex part of q is $[\![q]\!] = \{R(\underline{x}, y),\ S(\underline{y}, u, a)\}$. The complex part of $\{R(\underline{y}, w),\ R(\underline{x}, u),\ T(\underline{x, y})\}$, where T is all-key, is empty.

If some atom $F = R(\underline{\boldsymbol{x}}, y_1, \ldots, y_\ell)$ of a Boolean conjunctive query q does *not* belong to q's complex part, then $y_1, \ldots, y_\ell$ are distinct variables that have only one occurrence in q. Intuitively, such variables can be disregarded when evaluating the query q, because they do not impose any join condition.

Function IsSafe takes a query $q \in$ SjfBcq on input, and always terminates with either **true** or **false**. The function is recursive. The base rules (SE0a and SE0b) apply if q consists of a single fact, or if the complex part of q is empty.

[2] Liaison variables are sometimes called "join variables" in the literature. Notice nevertheless that in the singleton query $\{R(\underline{x}, x)\}$, which is not a genuine join, the variable x is a liaison variable.

Function. IsSafe(q) Determine whether q is safe

Input: A query q in SjfBcq.
Result: Boolean in {**true**, **false**}.
begin

SE0a: **if** $|q| = 1$ *and* $\mathsf{vars}(q) = \emptyset$ **then**
  return true;

SE0b: **if** $[\![q]\!] = \emptyset$ **then**
  return true;

SE1: **if** $q = q_1 \cup q_2$ *with* $q_1 \neq \emptyset \neq q_2$, $\mathsf{vars}(q_1) \cap \mathsf{vars}(q_2) = \emptyset$ **then**
  return *IsSafe*(q_1) $\wedge$ *IsSafe*(q_2);

/* a is an arbitrary constant */

SE2: **if** $[\![q]\!] \neq \emptyset$ *and* $\bigcap_{F \in [\![q]\!]} \mathsf{key}(F) \neq \emptyset$ **then**
  select $x \in \bigcap_{F \in [\![q]\!]} \mathsf{key}(F)$;
  return *IsSafe*($q_{[x \mapsto a]}$);

SE3: **if** *there exists* $F \in q$ *such that* $\mathsf{key}(F) = \emptyset \neq \mathsf{vars}(F)$ **then**
  select $F \in q$ such that $\mathsf{key}(F) = \emptyset \neq \mathsf{vars}(F)$;
  select $x \in \mathsf{vars}(F)$;
  return *IsSafe*($q_{[x \mapsto a]}$);

if *none of the above* **then**
  return false;

The recursive rule SE1 applies if q can be partitioned into two subqueries which have no variables in common. The recursive rule SE2 applies if for some variable x, all atoms in the complex part of q contain x at some of their primary-key positions. The recursive rule SE3 applies if all primary-key positions of some atom are occupied by constants and some non-primary-key position contains a variable.

A query $q \in$ SjfBcq is called *safe* if Function IsSafe returns **true** on input q; otherwise q is *unsafe*. The following result refines Theorem 4.

Theorem 8 ([24]). *For every* $q \in$ SjfBcq*,*

1. *if* q *is safe, then* $\sharp\mathsf{CERTAINTY}(q)$ *is in* **FP***; and*
2. *if* q *is unsafe, then* $\sharp\mathsf{CERTAINTY}(q)$ *is* $\sharp$**P***-hard.*

We use the following helping lemma.

Lemma 1. *For every* $q \in$ SjfBcq*, if* q *is safe, then* $\mathsf{CERTAINTY}(q)$ *is in* **FO**.

Proof. Let $q \in$ SjfBcq such that q is safe. The proof runs by induction on the execution of Function IsSafe. Some rule among SE0a, SE0b, SE1, SE2, or SE3 must apply to q.

Case SE0a Applies. If q consists of a single fact, then $\mathsf{CERTAINTY}(q)$ is obviously in **FO**.

Case SE0b Applies. If $[\![q]\!] = \emptyset$, then for any given uncertain database $\mathbf{db}$, we have that q evaluates to true on every repair of $\mathbf{db}$ if and only if q evaluates to true on $\mathbf{db}$. It follows that CERTAINTY(q) is in **FO**.

Case SE1 Applies. Let $q = q_1 \cup q_2$ with $q_1 \neq \emptyset \neq q_2$ and $\mathsf{vars}(q_1) \cap \mathsf{vars}(q_2) = \emptyset$. Since q is safe, q_1 and q_2 are safe by definition of safety. By the induction hypothesis, there exists a consistent first-order rewriting φ_1 of q_1, and a consistent first-order rewriting φ_2 of q_2. Obviously, $\varphi_1 \wedge \varphi_2$ is a consistent first-order rewriting of q.

Case SE2 Applies. Assume variable x such that for every $F \in [\![q]\!] \neq \emptyset$, $x \in \mathsf{key}(F)$. We first show that for every uncertain database $\mathbf{db}$, the following are equivalent:

1. every repair of $\mathbf{db}$ satisfies q; and
2. for some constant a, every repair of $\mathbf{db}$ satisfies $q_{[x \mapsto a]}$.

$\boxed{2 \Longrightarrow 1}$ Trivial. $\boxed{1 \Longrightarrow 2}$ Proof by contraposition. Assume that for every constant a, there exists a repair $\mathbf{rep}_a$ of $\mathbf{db}$ such that $\mathbf{rep}_a \not\models q_{[x \mapsto a]}$. Assume without loss of generality that $[\![q]\!] = \{R_1(\underline{x, \boldsymbol{x}_1}, \boldsymbol{y}_1), \ldots, R_n(\underline{x, \boldsymbol{x}_n}, \boldsymbol{y}_n)\}$. For every $a \in \mathbf{adom}(\mathbf{db})$, let $\mathbf{rep}'_a$ be the subset of $\mathbf{rep}_a$ that contains each R_i-fact whose leftmost position is occupied by a, for all $1 \leq i \leq n$. Let $\mathbf{db}'$ be the subset of $\mathbf{db}$ that contains each fact F whose relation name is not among $R_1, \ldots, R_n$. Let $\mathbf{rep}'$ be a repair of $\mathbf{db}'$. It can now be easily seen that $\mathbf{rep}' \cup \left(\bigcup_{a \in \mathbf{adom}(\mathbf{db})} \mathbf{rep}'_a \right)$ is a repair of $\mathbf{db}$ that falsifies q.

By definition of safety, $q_{[x \mapsto a]}$ is safe. By the induction hypothesis, the problem CERTAINTY($q_{[x \mapsto a]}$) is in **FO**. Let φ be a consistent first-order rewriting of $q_{[x \mapsto c]}$, where we assume without loss of generality that c is a constant that does not occur in q. Let $\varphi(x)$ be the first-order formula obtained from φ by replacing each occurrence of c with x. By the equivalence shown in the previous paragraph, $\exists x \varphi(x)$ is a consistent first-order rewriting of q.

Case SE3 Applies. Assume $F \in q$ such that $\mathsf{key}(F) = \emptyset$ and $\mathsf{vars}(F) \neq \emptyset$. Let $\boldsymbol{x}$ be a sequence of distinct variables such that $\mathsf{vars}(\boldsymbol{x}) = \mathsf{vars}(F)$. Let $\boldsymbol{a} = \langle a, a, \ldots, a \rangle$ be a sequence of length $|\boldsymbol{x}|$. By definition of safety, $q_{[\boldsymbol{x} \mapsto \boldsymbol{a}]}$ is safe. By the induction hypothesis, CERTAINTY($q_{[\boldsymbol{x} \mapsto \boldsymbol{a}]}$) is in **FO**. From Lemma 8.6 in [34], it follows that CERTAINTY(q) is in **FO**. This concludes the proof of Lemma 1. □

The proof of Theorem 6 can now be given.

Proof (of Theorem 6). Assume that CERTAINTY(q) is not in **FO**. By Lemma 1, q is unsafe. By Theorem 8, ♯CERTAINTY(q) is ♯**P**-hard. □