Certain Query Answering in Partially Consistent Databases

Sergio Greco
University of Calabria
Italy
greco@deis.unical.it

Fabian Pijcke
University of Mons
Belgium
fabian.pijcke@umons.ac.be

Jef Wijsen
University of Mons
Belgium
jef.wijsen@umons.ac.be

ABSTRACT

A database is called uncertain if two or more tuples of the same relation are allowed to agree on their primary key. Intuitively, such tuples act as alternatives for each other. A repair (or possible world) of such uncertain database is obtained by selecting a maximal number of tuples without ever selecting two tuples of the same relation that agree on their primary key. For a Boolean query q, the problem CERTAINTY(q) takes as input an uncertain database **db** and asks whether q evaluates to true on every repair of **db**. In recent years, the complexity of CERTAINTY(q) has been studied under different restrictions on q. These complexity studies have assumed no restrictions on the uncertain databases that are input to CERTAINTY(q). In practice, however, it may be known that these input databases are partially consistent, in the sense that they satisfy some dependencies (e.g., functional dependencies). In this article, we introduce the problem CERTAINTY(q) in the presence of a set Σ of dependencies. The problem CERTAINTY(q, Σ) takes as input an uncertain database **db** *that satisfies* Σ, and asks whether every repair of **db** satisfies q.

We focus on the complexity of CERTAINTY(q, Σ) when q is an acyclic conjunctive query without self-join, and Σ is a set of functional dependencies and join dependencies, the latter of a particular form. We provide an algorithm that, given q and Σ, decides whether CERTAINTY(q, Σ) is first-order expressible. Moreover, we show how to effectively construct a first-order definition of CERTAINTY(q, Σ) if it exists.

1. MOTIVATION

An uncertain database is a database that can contain zero, one, or more primary key violations. Two or more tuples of the same relation that agree on their primary key act as alternatives for each other and thus represent uncertainty. We will always assume exactly one primary key per relation. For example, consider the relations R and S in Figure 1, where primary keys are underlined. Employees are uniquely identified by their first and last name. The relation S gives touristic appreciations for cities in terms of Michelin stars. There is uncertainty about the salary and the residence of Ed Smith, about the birth year of An Allen, about the touristic value of the city of Acri, and about the country of Mons. Such uncertainty may result from data integration, or may reflect divergent opinions, e.g., concerning touristic value. In planning databases, key-equal tuples can be useful to model different possible choices.[1]

Uncertainty could be solved by removing duplicates. In practice, however, it may not be clear which tuples should be deleted. Instead of making arbitrary deletions, a more principled approach is to cope with all possible ways of restoring consistency, as explained next.

A repair (or possible world) of an uncertain database **db** is a maximal consistent subset of **db**. In general, the number of repairs of **db** is exponential in the size of **db**. To answer queries on uncertain databases, we follow the paradigm of *consistent query answering* [3, 5], which we also refer to as *certain query answering*. Given a Boolean query q, the problem CERTAINTY(q) takes as input an uncertain database **db** and asks whether q evaluates to true on every repair of **db**. In the last few years, the complexity of this problem has been studied in particular for q ranging over the class of Boolean conjunctive queries without self-join [16, 17, 30, 31, 32, 33]. In this case, the complexity varies from $\mathbf{AC}^0$ to **coNP**-complete, depending on q.

All existing works on CERTAINTY(q) have assumed that primary keys are the only integrity constraints involved, and that they can be violated at any one time. However, in practice, some primary keys or some other constraints may well be satisfied. This happens, for example, when some (but not all) constraints are enforced by the database system. In this article, we study the problem CERTAINTY(q) *in the presence of a set* Σ *of functional and join dependencies.* The problem CERTAINTY(q, Σ) takes as input an uncertain database **db** *that satisfies* Σ, and asks whether q evaluates to true on every repair of **db**. Thus, the only constraints that can be violated are primary keys not implied by Σ. We next illustrate the interest of this problem by two examples.

Example 1. Although the relation R in Figure 1 violates its primary key, it can be observed that the functional dependency (FD) $R : \mathbf{City} \rightarrow \mathbf{Country}$ (call it σ_1) holds (but $S : \mathbf{City} \rightarrow \mathbf{Country}$ is violated).

[1]For this reason, we prefer the term uncertain database over inconsistent database.

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/3.0/. Obtain permission prior to any use beyond those covered by the license. Contact copyright holder by emailing info@vldb.org. Articles from this volume were invited to present their results at the 40th International Conference on Very Large Data Bases, September 1st - 5th 2014, Hangzhou, China.
Proceedings of the VLDB Endowment, Vol. 7, No. 5
Copyright 2014 VLDB Endowment 2150-8097/14/01.

R	First	Last	Birth	Sal	City	Country
	Ed	Smith	1960	50K	Acri	Italy
	Ed	Smith	1960	50K	Mons	Belgium
	Ed	Smith	1960	60K	Acri	Italy
	Ed	Smith	1960	60K	Mons	Belgium
	An	Allen	1970	40K	Mons	Belgium
	An	Allen	1971	40K	Mons	Belgium

S	City	Country	Stars
	Acri	Italy	**
	Acri	Italy	***
	Mons	Belgium	***
	Mons	France	***

Figure 1: Uncertain database satisfying functional dependency R : City $\rightarrow$ Country and join dependency $R :\bowtie [\{\textbf{First}, \textbf{Last}, \textbf{Birth}\}, \{\textbf{First}, \textbf{Last}, \textbf{Sal}\}, \{\textbf{First}, \textbf{Last}, \textbf{City}, \textbf{Country}\}]$.

Consider the conjunctive query q_1 asking whether some employees live in three-star cities:

$$q_1 = \exists u \exists v \exists w \exists x \exists y \exists z \big(R(\underline{u, v}, w, x, y, z) \wedge S(\underline{y}, z, \text{`***'})\big).$$

Earlier work [17] implies that $\mathsf{CERTAINTY}(q_1)$ is **coNP**-complete. Nevertheless, the results in the current article will imply that $\mathsf{CERTAINTY}(q_1, \{\sigma_1\})$ is first-order expressible and hence in the low complexity class $\mathbf{AC}^0$. In practice, this means that the problem can be solved by a single SQL query. Thus, in the presence of an FD that is satisfied, the complexity of certain query answering can significantly decrease, from intractable to highly tractable, even if that FD is not a key dependency.

Example 2. The relation R of Figure 1 stores three kinds of information about employees: year of birth (attribute **Birth**), salary (attribute **Sal**), and residence (composite attribute composed of **City** and **Country**). In a consistent database, these attributes are single-valued for each employee. On the other hand, the inconsistent relation R of Figure 1 stores two distinct salaries and two distinct residences for Ed Smith. It can be observed though that all four combinations of these conflicts occur in R. More generally, it can be observed that the relation R satisfies the following join dependency (JD):

$$R :\bowtie \left[\begin{array}{l} \{\textbf{First}, \textbf{Last}, \textbf{Birth}\}, \\ \{\textbf{First}, \textbf{Last}, \textbf{Sal}\}, \\ \{\textbf{First}, \textbf{Last}, \textbf{City}, \textbf{Country}\} \end{array} \right].$$

The three components of the above join dependency share the primary key $\{\textbf{First}, \textbf{Last}\}$ of R. We will use the term *key join dependency* (KJD) for join dependencies in which all components share the primary key (and share no other attribute). KJDs automatically arise in relations that are obtained by joining source relations on some common key. In the example, the relation R was obtained by integrating three sources: a first source containing birth years, a second source with salaries, and a third source with residences. Otherwise, if conflicts in different attributes can be assumed to be independent, then it may be sensible to induce KJD satisfaction by combining conflicting values in all possible ways [28].

In this article, we study the complexity of the problem $\mathsf{CERTAINTY}(q, \Sigma)$ when q is an acyclic Boolean conjunctive query without self-join, and Σ is a set of FDs and KJDs, containing at most one KJD per relation name. In particular, we show that it is decidable to determine whether $\mathsf{CERTAINTY}(q, \Sigma)$ is first-order expressible (and hence in the low complexity class $\mathbf{AC}^0$). This contribution is of practical relevance, because it tells us which cases of $\mathsf{CERTAINTY}(q, \Sigma)$ can be solved by standard "first-order" SQL capabilities.

Conceptually, our work adds flexibility to consistent query answering [3, 5]. Most existing works in this field have assumed that all constraints can be potentially violated. In our approach, we distinguish two classes of constraints: those that may be violated (primary keys in our setting), and those that are known to be satisfied. In practice, one may learn the satisfied constraints in various ways: one may simply look at the data and "mine" dependencies that hold; one may know that the database system enforces some constraints; or one may have partially cleansed the data to satisfy some constraints. In any way, the knowledge about satisfied constraints can be favorably exploited in certain query answering.

This article is organized as follows. Section 2 introduces the theoretical framework and the problem statement. Section 3 discusses more related work in addition to the works already discussed in the current section. Section 4 recalls the notion of attack graph [32], which will turn out to be an important tool in the complexity study of $\mathsf{CERTAINTY}(q, \Sigma)$. Section 5 extends the notion of attack graph to deal with FDs and KJDs, and shows that it is decidable, given q and Σ, whether $\mathsf{CERTAINTY}(q, \Sigma)$ is first-order expressible. Section 6 explains how to effectively construct a first-order definition of $\mathsf{CERTAINTY}(q, \Sigma)$ if it exists. Section 7 discusses the practical implications of our theoretical contribution. Finally, Section 8 concludes the article. The missing proofs of lemmas and theorems can be found in the full version of this paper.

2. PROBLEM STATEMENT

We assume disjoint sets of *variables* and *constants*. Variables and constants are *symbols*. If $\vec{x}$ is a sequence of symbols, then $\mathsf{Vars}(\vec{x})$ is the set of variables that occur in $\vec{x}$. If S is a set of symbols, then $\mathsf{Vars}(S)$ is the set of variables that belong to S.

Let U be a set of variables. A *valuation over* U is a total mapping θ from U to the set of constants. Such valuation θ is often extended to be the identity on constants and on variables not in U.

Atoms and key-equal facts. Every *relation name* R has a unique *signature*, which is a pair $[n, k]$ with $n \geq k \geq 1$: the integer n is the *arity* of the relation name and $\{1, 2, \ldots, k\}$ is the *primary key*. The relation name R is *all-key* if $n = k$. If R is a relation name with signature $[n, k]$, then $R(s_1, \ldots, s_n)$ is an *R-atom* (or simply atom), where each s_i is a constant or a variable ($1 \leq i \leq n$). Such atom is commonly written as $R(\underline{\vec{x}}, \vec{y})$ where $\vec{x} = s_1, \ldots, s_k$ and $\vec{y} = s_{k+1}, \ldots, s_n$. Thus, the positions of the primary key will be underlined. An *R-fact* (or simply fact) is an R-atom in which no vari-

able occurs. Two facts $R_1(\underline{\vec{a}_1}, \vec{b}_1), R_2(\underline{\vec{a}_2}, \vec{b}_2)$ are *key-equal* if $R_1 = R_2$ and $\vec{a}_1 = \vec{a}_2$.

Uncertain database and repair. A *database schema* is a finite set of relation names. An *uncertain database* **db** over a given database schema is a finite set of facts using only the relation names of the schema. Importantly, an uncertain database is allowed to violate primary key constraints. An uncertain database **db** is *consistent* if it does not contain two distinct facts that are key-equal. A *repair* of an uncertain database **db** is a maximal (under set inclusion) consistent subset of **db**. We write $\mathsf{rset}(\mathbf{db})$ for the set of repairs of **db**.

Boolean conjunctive query. A *Boolean conjunctive query* is a finite set $q = \{R_1(\underline{\vec{x}_1}, \vec{y}_1), \ldots, R_n(\underline{\vec{x}_n}, \vec{y}_n)\}$ of atoms, representing the first-order sentence

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

where $u_1, \ldots, u_k$ are all the variables that have an occurrence in $\vec{x}_1\vec{y}_1 \ldots \vec{x}_n\vec{y}_n$. The *schema of* q, denoted $\mathsf{schema}(q)$, is the set of relation names that occur in q. We write $\mathsf{Vars}(q)$ for the set of variables that occur in q.

The query q is *satisfied* by an uncertain database **db**, denoted $\mathbf{db} \models q$, if there exists a valuation θ over $\mathsf{Vars}(q)$ such that for each $i \in \{1, \ldots, n\}$, $R_i(\theta(\underline{\vec{x}_i}), \theta(\vec{y}_i)) \in \mathbf{db}$. The restriction to Boolean queries simplifies the technical treatment, but is not fundamental as explained in Section 6.

We will use letters A, B, C for database facts, and F, G, H, I for query atoms. For $F = R(\underline{\vec{x}}, \vec{y})$, we denote by $\mathsf{KVars}(F)$ the set of variables that occur in $\vec{x}$, and by $\mathsf{Vars}(F)$ the set of variables that occur in F, that is, $\mathsf{KVars}(F) = \mathsf{Vars}(\vec{x})$ and $\mathsf{Vars}(F) = \mathsf{Vars}(\vec{x}) \cup \mathsf{Vars}(\vec{y})$.

We say that query q has a *self-join* if some relation name occurs more than once in q. If q has no self-join, it is called *self-join-free*. We denote by SJFCQ the class of self-join-free conjunctive queries.

A Boolean conjunctive query q is *acyclic* if it has a *join tree* [4]. 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. It is common to label, in a join tree, each edge with the set of variables common to its end points. We write $F \overset{L}{\frown} G$ to denote an edge between F and G with label L, where $L = \mathsf{Vars}(F) \cap \mathsf{Vars}(G)$.

Example 3. A join tree is shown in Figure 2 (left).

Certain query answering. Let q be an acyclic Boolean SJFCQ query. The problem of certain query answering is the following [30].

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

It is well known that the problem $\mathsf{CERTAINTY}(q)$ is in **coNP**, and that its complexity varies for varying q from $\mathbf{AC}^0$ to **coNP**-complete.

Certain query answering in the presence of constraints. Let R be a relation name with signature $[n, k]$. A *key join dependency* (KJD) has the form $R :\bowtie [K_1, \ldots, K_\ell]$ with $\ell \geq 1$ such that

1. $\bigcup_{i=1}^{\ell} K_i = \{1, \ldots, n\}$;
2. for every $1 \leq i < j \leq \ell$, we have $K_i \neq K_j$ and $K_i \cap K_j = \{1, \ldots, k\}$.

A *functional dependency* (FD) has the form

$$R : i_1, i_2, \ldots, i_m \to \ell,$$

where $1 \leq i_1 < i_2 < \cdots < i_m \leq n$ and $\ell \in \{1, \ldots, n\}$, $\ell \notin \{i_1, \ldots, i_m\}$. We will assume $m \geq 1$, although the technical treatment can be easily extended to treat FDs with an empty left-hand side. Notice that in the running example of Section 1, for readability reasons, attributes were denoted by names instead of positions.

The following definition of satisfaction of KJDs and FDs is standard (see, e.g., [1, page 159]). Let **db** be an uncertain database. Two R-facts $R(\underline{a_1, \ldots, a_k}, a_{k+1}, \ldots, a_n)$ and $R(\underline{b_1, \ldots, b_k}, b_{k+1}, \ldots, b_n)$ are said to *agree on position* i if $a_i = b_i$, where $i \in \{1, \ldots, n\}$. We say that **db** *satisfies* $R :\bowtie [K_1, \ldots, K_\ell]$ if whenever $A_1, \ldots, A_\ell$ are key-equal R-facts of **db**, then there exists an R-fact $B \in \mathbf{db}$ such that for all $i \in \{1, \ldots, \ell\}$, B and A_i agree on all positions in K_i. We say that **db** *satisfies* $R : i_1, i_2, \ldots, i_m \to \ell$ if for all R-facts $A, B \in \mathbf{db}$, if A and B agree on all positions among $i_1, \ldots, i_m$, then they agree on position ℓ. If σ is an FD or a KJD, then we write $\mathbf{db} \models \sigma$ to denote that **db** satisfies σ.

We say that a set Σ of KJDs and FDs is *jd-singular* if it does not contain two distinct KJDs with the same relation name; the number of FDs per relation name is not restricted. It will always be assumed that all relation names in Σ belong to $\mathsf{schema}(q)$ for some query q that is clear from the context.

Let q be an acyclic Boolean SJFCQ query. Let Σ be a jd-singular set of KJDs and FDs. The problem of certain query answering in the presence of constraints is the following.

PROBLEM:	$\mathsf{CERTAINTY}(q, \Sigma)$
INPUT:	uncertain database **db** such that $\mathbf{db} \models \Sigma$
QUESTION:	does every repair of **db** satisfy q?

If $\Sigma = \emptyset$, then $\mathsf{CERTAINTY}(q, \Sigma)$ is the same problem as $\mathsf{CERTAINTY}(q)$. The problem $\mathsf{CERTAINTY}(q, \Sigma)$ is in **coNP**, because if the answer to $\mathsf{CERTAINTY}(q, \Sigma)$ is "no," then a "no"-certificate is a repair of **db** that satisfies Σ and falsifies q. We are interested in deciding its complexity for varying q and Σ, in particular:

1. Is it decidable to determine, given q and Σ, whether $\mathsf{CERTAINTY}(q, \Sigma)$ is first-order expressible?
2. Is it decidable to determine, given q and Σ, whether $\mathsf{CERTAINTY}(q, \Sigma)$ is in **P**?
3. Is it decidable to determine, given q and Σ, whether $\mathsf{CERTAINTY}(q, \Sigma)$ is **coNP**-hard?

Saying that $\mathsf{CERTAINTY}(q, \Sigma)$ is first-order expressible is tantamount to saying that there exists a first-order sentence φ such that for every uncertain database **db** that satisfies Σ, the following are equivalent:

- Every repair of **db** satisfies q.
- $\mathbf{db} \models \varphi$.

Such a first-order sentence φ is called a first-order definition of $\mathsf{CERTAINTY}(q, \Sigma)$, or alternatively, a *consistent first-order rewriting of q relative to Σ*. Its practical interest is obvious: φ can be encoded in SQL and executed on any uncertain database by means of standard database technology.

The aforementioned questions 2 and 3 are open, even for $\Sigma = \emptyset$. Question 1 has been answered for $\Sigma = \emptyset$ in [30, 32]. In the current article, we answer question 1 affirmatively, as follows.

THEOREM 1. *For a given q and Σ, it is decidable whether $\mathsf{CERTAINTY}(q, \Sigma)$ is first-order expressible, where*

- *q is an acyclic Boolean* SJFCQ *query, and*
- *Σ is a jd-singular set of KJDs and FDs.*

Moreover, if $\mathsf{CERTAINTY}(q, \Sigma)$ is first-order expressible, then a first-order definition of $\mathsf{CERTAINTY}(q, \Sigma)$ can be effectively constructed.

Theorem 1 is a fairly deep result of practical interest. Its proof will be developed from Section 5 on. We briefly discuss the remaining restrictions on queries and constraints. The restriction to conjunctive queries that are acyclic and self-join-free allows us to use results proved in [32]. It is known since the early 1980s that the class of acyclic conjunctive queries has several elegant and useful characterizations [4]. The proof of Theorem 1 also relies on the hypothesis that Σ contains at most one KJD per relation name. This KJD can express that a relation is equal to a join of multiple relations that share the same primary key, as illustrated in Example 2.

3. RELATED WORK

Database repairing and consistent query answering were introduced in [3] for dealing with inconsistency in databases. They provide an elegant and principled alternative for data cleansing. Different notions of repair have been proposed in the literature, among others, symmetric-difference repairs [3], subset repairs [9], cardinality repairs [2, 20], update-based repairs [6, 7, 27], project-join repairs [28]... See also the survey [5]. If the only constraints are primary keys, then there is no difference between symmetric-difference, subset, and cardinality repairs. Most works have assumed that all constraints can potentially be violated. Notable exceptions are [6] and [19]. In [6], all primary keys are assumed to be satisfied, and only non-key attributes are subject to modification. The framework of [19] allows one to specify that some primary keys are satisfied.

For any fixed repair notion, there are two semantics for answering a Boolean query q. Under the certain semantics, the question is whether q is true in every repair. Under the possible semantics, the question is whether q is true in some repair. In this paper, we adopt the certain semantics.

Repairing has been studied with respect to different types and combinations of constraints, among others, denial constraints [6], inclusion dependencies [8], functional dependencies and foreign keys [22], tuple-generating and equality-generating dependencies [25], aggregate constraints [13]... The problem $\mathsf{CERTAINTY}(q)$ assumes that the constraints consist of one primary key per relation.

The detailed investigation of $\mathsf{CERTAINTY}(q)$ was pioneered by Fuxman and Miller [15, 16], who defined a class of self-join-free conjunctive queries q for which $\mathsf{CERTAINTY}(q)$ is first-order expressible. Since then, the following complexity classification problem has gained considerable research interest: Given a conjunctive query q on input, determine the complexity classes to which the problem $\mathsf{CERTAINTY}(q)$ belongs, or does not belong. For conjunctive queries with self-joins, this complexity classification problem remains largely open, which is likely due to the difficulty of treating self-joins. For self-join-free conjunctive queries, the following are known (all queries are assumed Boolean):

- Given an acyclic self-join-free conjunctive query q, it is decidable whether or not $\mathsf{CERTAINTY}(q)$ is first-order expressible [32].
- For each self-join-free conjunctive query q with exactly two atoms, $\mathsf{CERTAINTY}(q)$ is either in **P** or **coNP**-complete, and it is decidable which of the two cases applies [17]. The sufficient condition for **coNP**-completeness has later on been generalized to more than two atoms [33].
- For each self-join-free conjunctive query q in which all primary keys are either simple or contain all attributes of the relation, $\mathsf{CERTAINTY}(q)$ is either in **P** or **coNP**-complete, and it is decidable which of the two cases applies [19].

It remains an intriguing open conjecture that for each self-join-free conjunctive query q, $\mathsf{CERTAINTY}(q)$ is either in **P** or **coNP**-complete. Existing systems for solving the problem $\mathsf{CERTAINTY}(q)$ will be discussed in Section 7.

All aforementioned results assume queries without self-join. For queries q with self-joins, only fragmentary results about the complexity of $\mathsf{CERTAINTY}(q)$ are known [9, 29].

The counting variant of $\mathsf{CERTAINTY}(q)$, which has been denoted $\sharp\mathsf{CERTAINTY}(q)$, takes as input an uncertain database **db** and asks to determine the number of repairs of **db** that satisfy Boolean query q. As shown in [33], this problem is intimately related to query answering in block-independent-disjoint (BID) probabilistic databases [10, 11]. Maslowski and Wijsen [21] have proved that for each Boolean SJFCQ query q, the counting problem $\sharp\mathsf{CERTAINTY}(q)$ is either in **FP** or $\sharp$**P**-complete, and it is decidable which of the two is the case.

4. ATTACK GRAPH

In this section, we recall the construct of attack graph, which was introduced in [32]. The attack graph is a directed graph that is defined for every acyclic Boolean SJFCQ query. The main result of [32] is that for every acyclic Boolean SJFCQ query q, $\mathsf{CERTAINTY}(q)$ is first-order expressible if and only if q's attack graph is acyclic (see Theorem 2 hereinafter). Attack graphs will also be a useful tool in the study of $\mathsf{CERTAINTY}(q, \Sigma)$.

The primary key of an atom F gives rise to a functional dependency among the variables that occur in F. For example, $R(\underline{x, y}, z, u)$ gives rise to $\{x, y\} \to \{x, y, z, u\}$, which will be abbreviated as $xy \to xyzu$ (and which is equivalent to $xy \to zu$). Notice that the functional dependencies of the form $X \to Y$, with X, Y sets of variables, are always relative to a given query, and are thus different in nature from the FDs among positions (of the form $R : i_1, \ldots, i_m \to \ell$) considered in Section 2. The set $\mathcal{K}(q)$ defined next collects all functional dependencies that arise in atoms of the query q.

Figure 2: Join tree τ_2 (left) and attack graph (right) of query q_2.

Definition 1. Let q be a Boolean conjunctive query. We define $\mathcal{K}(q)$ as the following set of functional dependencies.

$$\mathcal{K}(q) = \{\mathsf{KVars}(F) \to \mathsf{Vars}(F) \mid F \in q\}$$

Concerning the following definition, recall from relational database theory [26, page 387] that if Σ is a set of functional dependencies over a set U of attributes and $X \subseteq U$, then the attribute closure of X (with respect to Σ) is the set $\{A \in U \mid \Sigma \models X \to A\}$.

Definition 2. Let q be a Boolean conjunctive query. For every $F \in q$, we define $F^{+,q}$ as the following set of variables.

$$F^{+,q} = \{x \in \mathsf{Vars}(q) \mid \mathcal{K}(q \setminus \{F\}) \models \mathsf{KVars}(F) \to x\}$$

In words, $F^{+,q}$ is the attribute closure of the set $\mathsf{KVars}(F)$ with respect to the set of functional dependencies that arise in the atoms of $q \setminus \{F\}$.

Example 4. Consider the query $q_2 = \{R(\underline{u,a},x), S(\underline{y},x,z), T(\underline{x},y), U(\underline{x},z)\}$, where a is a constant. A join tree for this query is shown in Figure 2 (left). To shorten notation, let $F = R(\underline{u,a},x)$, $G = S(\underline{y},x,z)$, $H = T(\underline{x},y)$, and $I = U(\underline{x},z)$, as indicated in the figure. We have the following.

$$\mathcal{K}(q_2 \setminus \{F\}) = \{y \to xyz, x \to xy, x \to xz\}$$
$$\mathsf{KVars}(F) = \{u\} \text{ and } F^{+,q_2} = \{u\}$$

$$\mathcal{K}(q_2 \setminus \{G\}) = \{u \to ux, x \to xy, x \to xz\}$$
$$\mathsf{KVars}(G) = \{y\} \text{ and } G^{+,q_2} = \{y\}$$

$$\mathcal{K}(q_2 \setminus \{H\}) = \{u \to ux, y \to xyz, x \to xz\}$$
$$\mathsf{KVars}(H) = \{x\} \text{ and } H^{+,q_2} = \{x,z\}$$

$$\mathcal{K}(q_2 \setminus \{I\}) = \{u \to ux, y \to xyz, x \to xy\}$$
$$\mathsf{KVars}(I) = \{x\} \text{ and } I^{+,q_2} = \{x,y,z\}$$

Definition 3. Let q be a Boolean conjunctive query that is acyclic. Let τ be a join tree for q. The *attack graph* of τ is a directed graph whose vertices are the atoms of q. There is a directed edge from F to G if F, G are distinct atoms such that for every label L on the unique path that links F and G in τ, we have $L \not\subseteq F^{+,q}$.

We write $F \overset{\tau}{\rightsquigarrow} G$ if the attack graph of τ contains a directed edge from F to G. The directed edge $F \overset{\tau}{\rightsquigarrow} G$ is also called an *attack from F to G*. If $F \overset{\tau}{\rightsquigarrow} G$, we say that F *attacks* G (or that G is attacked by F).

Example 5. This is a continuation of Example 4. Figure 2 (left) shows a join tree τ_2 for query q_2. The attack graph of τ_2 is shown in Figure 2 (right) and is computed as follows.

Let us first compute the attacks outgoing from F. The path from F to G in the join tree is $F \overset{\{x\}}{\frown} G$. Since the label $\{x\}$ is not contained in F^{+,q_2}, the attack graph contains a directed edge from F to G, i.e., $F \overset{\tau_2}{\rightsquigarrow} G$. The path from F to H in the join tree is $F \overset{\{x\}}{\frown} G \overset{\{x,y\}}{\frown} H$. Since no label on that path is contained in F^{+,q_2}, the attack graph contains a directed edge from F to H. In the same way, one finds that F attacks I.

Let us next compute the attacks outgoing from H. The path from H to G in the join tree is $H \overset{\{x,y\}}{\frown} G$. Since the label $\{x,y\}$ is not contained in H^{+,q_2}, the attack graph contains a directed edge from H to G, i.e., $H \overset{\tau_2}{\rightsquigarrow} G$. The path from H to F in the join tree is $H \overset{\{x,y\}}{\frown} G \overset{\{x\}}{\frown} F$. Since the label $\{x\}$ is contained in H^{+,q_2}, the attack graph contains no directed edge from H to F. And so on. The complete attack graph is shown in Figure 2 (right).

It was shown in [32] that if τ_1 and τ_2 are distinct join trees for the same conjunctive query q, then the attack graph of τ_1 is identical to the attack graph of τ_2. This motivates the following definition.

Definition 4. Let q be a Boolean conjunctive query that is acyclic. The attack graph of q is the attack graph of τ for any join tree τ for q. We write $F \overset{q}{\rightsquigarrow} G$ (or simply $F \rightsquigarrow G$ if q is clear from the context) to indicate that the attack graph of q contains a directed edge from F to G.

The attack graph of an acyclic Boolean query q can be computed in quadratic time in the length of q [32]. The main result in [32] is the following.

THEOREM 2 ([32]). *The following are equivalent for all acyclic Boolean* SJFCQ *queries q:*

1. *The attack graph of q is acyclic.*
2. $\mathsf{CERTAINTY}(q)$ *is first-order expressible.*

Example 6. Figure 3 shows the attack graph of q_1 introduced in Section 1. Let $F = R(\underline{u,v},w,x,y,z)$ and $G = S(\underline{y,z},\text{`}***\text{'})$. We have $F^{+,q_1} = \{u,v\}$ and $G^{+,q_1} = \{y\}$.

Figure 3: Attack graph of query q_1.

Since the shared variable z does not occur in F^{+,q_1} or G^{+,q_1}, the atoms F and G mutually attack one another. Since the attack graph is cyclic, CERTAINTY(q_1) is not first-order expressible.

5. PRESENCE OF KJDS AND FDS

In this section, given an acyclic Boolean SJFCQ query q and a set Σ of dependencies, we compute a new query denoted $q \otimes \Sigma$. This new query will also be conjunctive, acyclic, and self-join-free. The main result will be that CERTAINTY(q, Σ) is first-order expressible if and only if the attack graph of $q \otimes \Sigma$ is acyclic (Theorem 3).

The operator $\otimes$ transforms query atoms and database facts according to FDs and KJDs. We provide an example before giving the technical definition.

Example 7. Assume an atom $F = R(\underline{a_1, a_2}, a_3, a_4, a_5)$ with signature $[5, 2]$.

- An FD $R : 2, 3 \to 4$ (call it σ) will add to F two new atoms $R_1^\sigma(\underline{a_2, a_3}, a_4)$ and $R_2^\sigma(\underline{a_2, a_3}, a_4)$. Here, R_1^σ and R_2^σ are two new relation names which depend on σ. The two atoms differ in their relation names but are otherwise identical.
- A KJD $R :\bowtie [\{1, 2, 3\}, \{1, 2, 4, 5\}]$ will replace the atom F with three atoms $\widehat{R}(\underline{a_1, a_2, a_3, a_4, a_5})$, $R_1^{\bowtie}(\underline{a_1, a_2}, a_3)$, and $R_2^{\bowtie}(\underline{a_1, a_2}, a_4, a_5)$. Here, $\widehat{R}$ is a new relation name that is all-key, and $R_1^{\bowtie}$, $R_2^{\bowtie}$ are new relation names corresponding to the first and the second component of the KJD.

Definition 5. Let q be an acyclic Boolean SJFCQ query. Let **db** be an uncertain database such that all relation names of **db** belong to schema(q).

Let $\Sigma = \Sigma_1 \cup \Sigma_2$ where Σ_1 is a jd-singular set of KJDs and Σ_2 is a set of FDs (with zero or more FDs per relation name). The SJFCQ query $q \otimes \Sigma$ and the uncertain database $\mathbf{db} \otimes \Sigma$ are defined as follows. For every atom $F = R(\underline{s_1, \ldots, s_k}, s_{k+1}, \ldots, s_n)$ of q,

1. If Σ_1 contains no KJD for R, then $q \otimes \Sigma$ contains F and $\mathbf{db} \otimes \Sigma$ contains all R-facts of **db**.
2. If Σ_1 contains a KJD for R, then $q \otimes \Sigma$ contains the atom $\widehat{R}(\underline{s_1, \ldots, s_n})$ where $\widehat{R}$ is a new relation name with signature $[n, n]$. That is, $\widehat{R}$ is all-key.

   For every $R(\underline{a_1, \ldots, a_k}, a_{k+1}, \ldots, a_n)$ of **db**, it is the case that $\mathbf{db} \otimes \Sigma$ contains $\widehat{R}(\underline{a_1, \ldots, a_n})$.
3. If Σ_1 contains KJD $R :\bowtie [K_1, \ldots, K_\ell]$, then for each $i \in \{1, \ldots, \ell\}$, the SJFCQ query $q \otimes \Sigma$ contains a new $R_i^{\bowtie}$-atom. If $K_i = \{1, \ldots, k, j_1, \ldots, j_m\}$ where $k < j_1 < \cdots < j_m \leq n$, then the new $R_i^{\bowtie}$-atom is $R_i^{\bowtie}(\underline{s_1, \ldots, s_k}, s_{j_1}, \ldots, s_{j_m})$, where $R_i^{\bowtie}$ is a new relation name of signature $[k + m, k]$.

   For every $R(\underline{a_1, \ldots, a_k}, a_{k+1}, \ldots, a_n)$ of **db**, it is the case that $\mathbf{db} \otimes \Sigma$ contains $R_i^{\bowtie}(\underline{a_1, \ldots, a_k}, a_{j_1}, \ldots, a_{j_m})$.
4. If Σ_2 contains FD $R : i_1, i_2, \ldots, i_m \to \ell$ (call it σ), then $q \otimes \Sigma$ contains two new atoms $R_1^\sigma(\underline{s_{i_1}, \ldots, s_{i_m}}, s_\ell)$ and $R_2^\sigma(\underline{s_{i_1}, \ldots, s_{i_m}}, s_\ell)$, where R_1^σ and R_2^σ are two new relation names with signature $[m + 1, m]$.

   For every $R(\underline{a_1, \ldots, a_k}, a_{k+1}, \ldots, a_n)$ of **db**, it is the case that $\mathbf{db} \otimes \Sigma$ contains the facts $R_1^\sigma(\underline{a_{i_1}, \ldots, a_{i_m}}, a_\ell)$ and $R_2^\sigma(\underline{a_{i_1}, \ldots, a_{i_m}}, a_\ell)$.
5. $q \otimes \Sigma$ contains no other atoms than those specified in items 1–4; $\mathbf{db} \otimes \Sigma$ contains no other facts than those specified in items 1–4.

We refer to relation names $\widehat{R}$, $R_i^{\bowtie}$, R_1^σ, and R_2^σ as *spurious relation names*. Atoms that contain a spurious relation name are called *spurious atoms*.

Example 8. Let $q_3 = \{R(\underline{x}, y, z), S(\underline{y}, u, x, z)\}$. Let Σ_3 be the set of dependencies containing KJD $R :\bowtie [\{1, 2\}, \{1, 3\}]$ and FD $S : 3, 4 \to 1$ (call it σ_3). The query $q_3 \otimes \Sigma_3$ contains the following atoms:

- $\widehat{R}(\underline{x, y, z})$ where $\widehat{R}$ has signature $[3, 3]$;
- $R_1^{\bowtie}(\underline{x}, y)$ where $R_1^{\bowtie}$ has signature $[2, 1]$ and corresponds to the first component of the KJD;
- $R_2^{\bowtie}(\underline{x}, z)$ where $R_2^{\bowtie}$ has signature $[2, 1]$ and corresponds to the second component of the KJD;
- $S(\underline{y}, u, x, z)$. Notice that Σ_3 contains no KJD for S; and
- $S_1^{\sigma_3}(\underline{x, z}, y)$ and $S_2^{\sigma_3}(\underline{x, z}, y)$ where $S_1^{\sigma_3}$ and $S_2^{\sigma_3}$ both have signature $[3, 2]$.

A join tree for $q_3 \otimes \Sigma_3$ is shown in Figure 4 (left).

The following lemma states that the operator $\otimes$ is first-order expressible.

LEMMA 1. *Let q be an acyclic Boolean* SJFCQ *query. Let Σ be a jd-singular set of KJDs and FDs. For every spurious relation name S in $q \otimes \Sigma$, there exists a first-order query ψ_S such that for every uncertain database* **db** *over* schema(q)*, the set of S-facts of $\mathbf{db} \otimes \Sigma$ is equal to $\psi_S(\mathbf{db})$.*

PROOF. For every spurious relation name S, the query ψ_S is a projection. □

Example 8 showed that the query $q_3 \otimes \Sigma_3$ has a join tree. The following lemma shows that in general, if q is acyclic, then so is $q \otimes \Sigma$.

LEMMA 2. *If q is an acyclic Boolean* SJFCQ *query, and Σ is a jd-singular set of KJDs and FDs, then $q \otimes \Sigma$ is an acyclic Boolean* SJFCQ *query.*

LEMMA 3. *Let q be an acyclic Boolean* SJFCQ *query. Let $\Sigma = \Sigma_1 \cup \Sigma_2$ where Σ_1 is a jd-singular set of KJDs and Σ_2 is a set of FDs. The following statements are equivalent for every uncertain database* **db** *over* schema(q) *that satisfies Σ:*

1. *Every repair of* **db** *satisfies q.*
2. *Every repair of $\mathbf{db} \otimes \Sigma$ satisfies $q \otimes \Sigma$.*

The following examples illustrate that in Lemma 3, it is important to require that $\mathbf{db} \models \Sigma$.

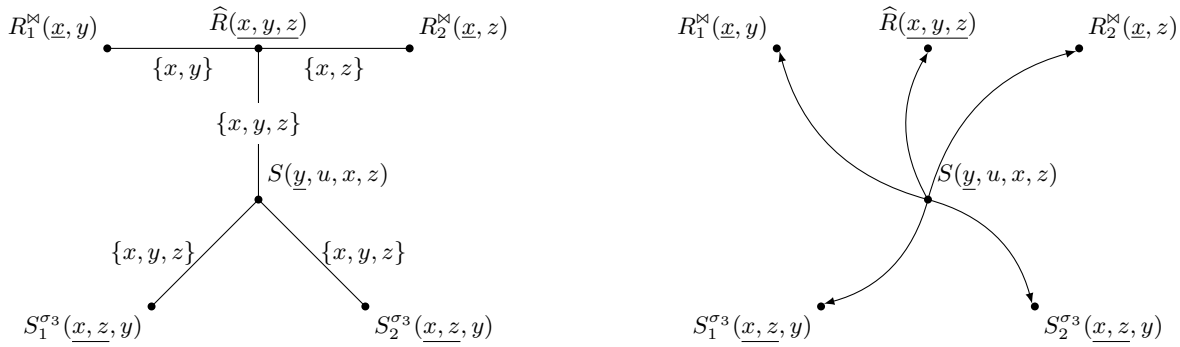

Figure 4: Join tree (left) and attack graph (right) of query $q_3 \otimes \Sigma_3$ of Example 8, where $\sigma_3 = S : 3, 4 \to 1$.

Example 9. Let $q = \{R(\underline{x}, y, z)\}$ and let Σ be the singleton containing KJD $R :\bowtie [\{1, 2\}, \{1, 3\}]$. We have $q \otimes \Sigma = \{\widehat{R}(\underline{x, y, z}),\ R_1^{\bowtie}(\underline{x}, y),\ R_2^{\bowtie}(\underline{x}, z)\}$.

Let $\mathbf{db} = \{R(\underline{a}, b_1, b_2), R(\underline{a}, c_1, c_2)\}$, which falsifies the KJD in Σ. We have $\mathbf{db} \otimes \Sigma = \{\widehat{R}(\underline{a, b_1, b_2}),\ \widehat{R}(\underline{a, c_1, c_2}),\ R_1^{\bowtie}(\underline{a}, b_1),\ R_1^{\bowtie}(\underline{a}, c_1),\ R_2^{\bowtie}(\underline{a}, b_2),\ R_2^{\bowtie}(\underline{a}, c_2)\}$.

Clearly, every repair of $\mathbf{db}$ satisfies q. However, the set $\{\widehat{R}(\underline{a, b_1, b_2}),\ \widehat{R}(\underline{a, c_1, c_2}),\ R_1^{\bowtie}(\underline{a}, b_1),\ R_2^{\bowtie}(\underline{a}, c_2)\}$ is a repair of $\mathbf{db} \otimes \Sigma$ that falsifies $q \otimes \Sigma$.

Example 10. Let $q = \{R(\underline{x}, y, z)\}$ and let Σ be the singleton containing FD $R : 2 \to 3$ (call it σ). We have $q \otimes \Sigma = \{R(\underline{x}, y, z), R_1^{\sigma}(\underline{y}, z), R_2^{\sigma}(\underline{y}, z)\}$.

Let $\mathbf{db} = \{R(\underline{a}, c, d), R(\underline{b}, c, e)\}$, which falsifies σ. We have $\mathbf{db} \otimes \Sigma = \{R(\underline{a}, c, d),\ R(\underline{b}, c, e),\ R_1^{\sigma}(\underline{c}, d),\ R_1^{\sigma}(\underline{c}, e),\ R_2^{\sigma}(\underline{c}, d),\ R_2^{\sigma}(\underline{c}, e)\}$.

Clearly, every repair of $\mathbf{db}$ satisfies q. However, $\{R(\underline{a}, c, d),\ R(\underline{b}, c, e),\ R_1^{\sigma}(\underline{c}, d),\ R_2^{\sigma}(\underline{c}, e)\}$ is a repair of $\mathbf{db} \otimes \Sigma$ that falsifies $q \otimes \Sigma$.

THEOREM 3. *Let q be an acyclic Boolean* SJFCQ *query. Let Σ be a jd-singular set of KJDs and FDs. The following statements are equivalent:*

1. $\mathsf{CERTAINTY}(q, \Sigma)$ *is first-order expressible.*
2. *The attack graph of $q \otimes \Sigma$ is acyclic.*

PROOF. $\boxed{1 \Rightarrow 2}$ Proof by contraposition. Assume the attack graph of $q \otimes \Sigma$ is cyclic. By [32, Lemma 7.3], the attack graph of $q \otimes \Sigma$ contains two atoms, say $\tilde{F}$ and $\tilde{G}$, that mutually attack each other. It can be easily verified that:

1. if $\tilde{F}$ is an $S_i^{\bowtie}$-atom and $\tilde{G}$ a $T_j^{\bowtie}$-atom, where relation names $S_i^{\bowtie}$ and $T_j^{\bowtie}$ come from KJDs for S and T respectively, then $S \neq T$;
2. the attack graph of $q \otimes \Sigma$ contains no attacks starting from an $\widehat{R}$-atom, where relation name $\widehat{R}$ comes from a KJD for R, because $\widehat{R}$ is all-key; and
3. the attack graph of $q \otimes \Sigma$ contains no attacks starting from R_i^{σ}-atoms, where relation name R_i^{σ} comes from an FD $\sigma \in \Sigma$ and $i \in \{1, 2\}$.

Consequently, we can assume distinct relation names S, T and positive integers i, j such that $\tilde{F}$ is either an S-atom or an $S_i^{\bowtie}$-atom, and $\tilde{G}$ is either a T-atom or a $T_j^{\bowtie}$-atom.

Let F and G be the S-atom and T-atom of q respectively. Notice that F and $\tilde{F}$ agree on all primary-key positions. Likewise, G and $\tilde{G}$ agree on all primary-key positions. Let τ be a join tree for q. It can be easily seen that for every label L on the unique path in τ between F and G, there exists $u, w \in L$ such that $u \notin \tilde{F}^{+,q\otimes\Sigma}$ and $w \notin \tilde{G}^{+,q\otimes\Sigma}$.

Note incidentally that from $F^{+,q} \subseteq \tilde{F}^{+,q\otimes\Sigma}$ and $G^{+,q} \subseteq \tilde{G}^{+,q\otimes\Sigma}$, it follows that F and G mutually attack each other in the attack graph of q.

Let φ be a first-order sentence over $\mathsf{schema}(q)$. Build $\mathbf{db}_{yes}$ and $\mathbf{db}_{no}$ as in the proof of [32, Theorem 5.1] with the difference that $F^{+,q}$ is replaced with $\tilde{F}^{+,q\otimes\Sigma}$, and $G^{+,q}$ with $\tilde{G}^{+,q\otimes\Sigma}$. It suffices to show the following:

1. $\mathbf{db}_{yes} \models \Sigma$ and $\mathbf{db}_{no} \models \Sigma$;
2. $\mathbf{db}_{yes} \in \mathsf{CERTAINTY}(q)$ and $\mathbf{db}_{no} \notin \mathsf{CERTAINTY}(q)$;
3. $\mathbf{db}_{yes} \models \varphi \iff \mathbf{db}_{no} \models \varphi$.

The proofs of the last two items are exactly as in the proof of [32, Theorem 5.1]. We show next that $\mathbf{db}_{yes} \models \Sigma$ (the proof of $\mathbf{db}_{no} \models \Sigma$ is analogous).

First assume Σ contains $R :\bowtie [K_1, \ldots, K_\ell]$ where R has signature $[n, k]$. Two cases can occur.

Case $R = S$ or $R = T$. Assume $R = S$ (the case $R = T$ is analogous). Assume that the S-atom of q is $F = S(\underline{s_1, \ldots, s_k}, s_{k+1}, \ldots, s_n)$. We can assume $h \in \{1, \ldots, \ell\}$ such that that $\tilde{F}$ is an $S_h^{\bowtie}$-atom. Notice that $\mathsf{KVars}(F) = \mathsf{KVars}(\tilde{F})$.

Let $i \in \{1, \ldots, \ell\}$ such that $i \neq h$, and let $K_i = \{1, \ldots, k, j_1, \ldots, j_m\}$ with $k < j_1 < \cdots < j_m \leq n$. Since the set $\mathcal{K}((q \otimes \Sigma) \setminus \{\tilde{F}\})$ contains the FD

$$\mathsf{KVars}(F) \to \mathsf{Vars}(\{s_{j_1}, \ldots, s_{j_m}\})$$

and since $\mathsf{KVars}(F) \subseteq \tilde{F}^{+,q\otimes\Sigma}$ is obvious, we have

$$\mathsf{Vars}(\{s_1, \ldots, s_k, s_{j_1}, \ldots, s_{j_m}\}) \subseteq \tilde{F}^{+,q\otimes\Sigma}.$$

The construction in the proof of [32, Theorem 5.1] will ensure that whenever two S-atoms of $\mathbf{db}_{yes}$ agree on all positions among $\{1, \ldots, k\}$, then they agree on all positions in K_i. That is, $\mathbf{db}_{yes} \models S : \{1, \ldots, k\} \to \{j_1, \ldots, j_m\}$.

To conclude, for each $i \in \{1, \ldots, \ell\}$ such that $i \neq h$, we have that $\mathbf{db}_{yes} \models S : \{1, \ldots, k\} \to K_i$. By Heath's theorem, $\mathbf{db}_{yes} \models S :\bowtie [K_1, \ldots, K_\ell]$.

Case $S \neq R \neq T$. Since the construction in the proof of [32, Theorem 5.1] ensures that no two R-facts of $\mathbf{db}_{yes}$ will be key-equal, it is obvious that $\mathbf{db}_{yes} \models R :\bowtie [K_1, \ldots, K_\ell]$.

Next assume Σ contains FD $R : i_1, \ldots, i_m \to \ell$ (call it σ) where R has signature $[n, k]$. Assume the R-atom of q is $R(\underline{s_1, \ldots, s_k}, s_{k+1}, \ldots, s_n)$. Let $X = \mathsf{Vars}(\{s_{i_1}, \ldots, s_{i_m}\})$ and $Y = \mathsf{Vars}(\{s_\ell\})$. Since the query $q \otimes \Sigma$ contains the atom $R_1^\sigma(\underline{s_{i_1}, \ldots, s_{i_m}}, s_\ell)$, we have that $\mathcal{K}((q \otimes \Sigma) \setminus \{\tilde{F}\})$ and $\mathcal{K}((q \otimes \Sigma) \setminus \{\tilde{G}\})$ both contain $X \to Y$ (recall that neither $\tilde{F}$ nor $\tilde{G}$ is an R_1^σ-atom). Two cases can occur.

Case $X \subseteq \tilde{F}^{+,q\otimes\Sigma} \cup \tilde{G}^{+,q\otimes\Sigma}$**.** We distinguish two subcases.

Subcase $X \subseteq \tilde{F}^{+,q\otimes\Sigma}$ **or** $X \subseteq \tilde{G}^{+,q\otimes\Sigma}$**.** Assume that $X \subseteq \tilde{F}^{+,q\otimes\Sigma}$ (the case $X \subseteq \tilde{G}^{+,q\otimes\Sigma}$ is analogous). Then $X \cup Y \subseteq \tilde{F}^{+,q\otimes\Sigma}$. The construction ensures that if two R-atoms of $\mathbf{db}_{yes}$ agree on all positions among $i_1, \ldots, i_m$, then they agree on position ℓ. It follows $\mathbf{db}_{yes} \models \sigma$.

Subcase $X \not\subseteq \tilde{F}^{+,q\otimes\Sigma}$ **and** $X \not\subseteq \tilde{G}^{+,q\otimes\Sigma}$**.** We can assume indices $g, h \in \{i_1, \ldots, i_m\}$ such that $s_g, s_h \in X$, $s_g \in \tilde{F}^{+,q\otimes\Sigma} \setminus \tilde{G}^{+,q\otimes\Sigma}$ and $s_h \in \tilde{G}^{+,q\otimes\Sigma} \setminus \tilde{F}^{+,q\otimes\Sigma}$. The construction ensures that no two R-atoms of $\mathbf{db}_{yes}$ agree on both positions g and h. It follows $\mathbf{db}_{yes} \models \sigma$.

Case $X \not\subseteq \tilde{F}^{+,q\otimes\Sigma} \cup \tilde{G}^{+,q\otimes\Sigma}$**.** We can assume the existence of $g \in \{i_1, \ldots, i_m\}$ such that $s_g \in X$ and $s_g \notin \tilde{F}^{+,q\otimes\Sigma} \cup \tilde{G}^{+,q\otimes\Sigma}$. The construction ensures that no two R-atoms of $\mathbf{db}_{yes}$ agree on position g. It follows $\mathbf{db}_{yes} \models \sigma$.

$\boxed{2 \Rightarrow 1}$ Assume the attack graph of $q \otimes \Sigma$ is acyclic. By Theorem 2, $\mathsf{CERTAINTY}(q \otimes \Sigma)$ is first-order expressible. We can assume a first-order formula ψ such that for every uncertain database $\tilde{\mathbf{db}}$ over $\mathsf{schema}(q \otimes \Sigma)$, we have that ψ evaluates to true on $\tilde{\mathbf{db}}$ if and only if every repair of $\tilde{\mathbf{db}}$ satisfies $q \otimes \Sigma$.

For every uncertain database $\mathbf{db}$ over $\mathsf{schema}(q)$, it is the case that $\mathbf{db} \otimes \Sigma$ is a uncertain database over $\mathsf{schema}(q \otimes \Sigma)$. It is correct to conclude that for every uncertain database $\mathbf{db}$ over $\mathsf{schema}(q)$, we have that ψ evaluates to true on $\mathbf{db} \otimes \Sigma$ if and only if every repair of $\mathbf{db} \otimes \Sigma$ satisfies $q \otimes \Sigma$.

By Lemma 1, $\mathbf{db} \otimes \Sigma$ is first-order computable from $\mathbf{db}$. Consequently, there exists a first-order formula $\tilde{\psi}$ such that for every uncertain database $\mathbf{db}$ over $\mathsf{schema}(q)$, we have that $\tilde{\psi}$ evaluates to true on $\mathbf{db}$ if and only if every repair of $\mathbf{db} \otimes \Sigma$ satisfies $q \otimes \Sigma$. In particular, for every uncertain database $\mathbf{db}$ that satisfies Σ, the following are equivalent:

1. $\tilde{\psi}$ evaluates to true on $\mathbf{db}$.
2. Every repair of $\mathbf{db} \otimes \Sigma$ satisfies $q \otimes \Sigma$.

Then, by Lemma 3, for every uncertain database $\mathbf{db}$ that satisfies Σ, the following are equivalent:

1. $\tilde{\psi}$ evaluates to true on $\mathbf{db}$.
2. Every repair of $\mathbf{db}$ satisfies q.

Hence $\mathsf{CERTAINTY}(q, \Sigma)$ is first-order expressible. □

Importantly, it happens that the attack graph of q is cyclic, and the attack graph of $q \otimes \Sigma$ is acyclic. Thus, by Theorems 2 and 3, there are cases where $\mathsf{CERTAINTY}(q, \Sigma)$ is first-order expressible, but $\mathsf{CERTAINTY}(q)$ is not. We illustrate this by two examples.

Example 11. Figure 3 shows the cyclic attack graph of q_1. Figure 5 (right) shows the acyclic attack graph of $q_1 \otimes \{\sigma_1\}$ where $\sigma_1 = R : 5 \to 6$. From Theorems 2 and 3, it follows that $\mathsf{CERTAINTY}(q_1)$ is not first-order expressible, but that $\mathsf{CERTAINTY}(q_1, \{\sigma_1\})$ is first-order expressible.

Example 12. Consider again the query $q_3 = \{R(\underline{x}, y, z), S(\underline{y}, u, x, z)\}$ introduced in Example 8. The attack graph of q_3 (not shown) is cyclic. Figure 4 (right) shows the attack graph of $q_3 \otimes \Sigma_3$ with $\Sigma_3 = \{R :\bowtie [\{1, 2\}, \{1, 3\}], S : 3, 4 \to 1\}$. The latter attack graph is acyclic.

Incidentally, one can easily verify that if we delete the KJD and/or the FD from Σ_3, then the attack graph remains cyclic. That is, both the KJD and the FD are needed to attain an acyclic attack graph.

The proof of Theorem 1 can now be given.

Proof of Theorem 1. Theorem 3 tells us that the problem $\mathsf{CERTAINTY}(q, \Sigma)$ is first-order expressible if and only if the attack graph of $q \otimes \Sigma$ is acyclic. Acyclicity of attack graphs can be tested in quadratic time [32]. Furthermore, the proof of Theorem 3 is constructive, meaning that it constructs a first-order definition of $\mathsf{CERTAINTY}(q, \Sigma)$ if it exists. □

6. CONSTRUCTING FIRST-ORDER DEFINITIONS

Assume $\mathsf{CERTAINTY}(q, \Sigma)$ is first-order expressible. The proof of Theorem 3 implies that a first-order definition of $\mathsf{CERTAINTY}(q, \Sigma)$ can be constructed in two steps: first, construct a first-order definition ψ of $\mathsf{CERTAINTY}(q \otimes \Sigma)$, and then substitute away from ψ all atoms with spurious relation names. In this section, we show a more efficient approach.

6.1 Attack Graph of (q, Σ)

The following definition extends the notion of attack graph to deal with the presence of KJDs and FDs.

Definition 6. Let q be an acyclic Boolean SJFCQ query. Let Σ be a jd-singular set of KJDs and FDs. The *attack graph of* (q, Σ) is a directed graph whose vertices are the atoms of q. If the attack graph of $q \otimes \Sigma$ contains an attack $F \overset{q\otimes\Sigma}{\rightsquigarrow} G$ where F is an R-atom or an $R_i^{\bowtie}$-atom, and G is an S-atom or an $S_j^{\bowtie}$-atom, then the attack graph of (q, Σ) contains a directed edge from the R-atom of q to the S-atom of q.

Example 13. From the attack graph of $q_3 \otimes \Sigma_3$ in Figure 4 (right), one finds that the attack graph of (q_3, Σ_3) consists of a single directed edge from $S(\underline{y}, u, x, z)$ to $R(\underline{x}, y, z)$.

Clearly, if $\Sigma = \emptyset$, then the attack graph of (q, Σ) is the same graph as the attack graph of q. The following lemma states that the attack graphs of $q \otimes \Sigma$ and (q, Σ) are either both cyclic or both acyclic.

Lemma 4. *Let q be an acyclic Boolean* SJFCQ *query. Let Σ be a jd-singular set of KJDs and FDs. The following are equivalent:*

1. *The attack graph of (q, Σ) is acyclic.*
2. *The attack graph of $q \otimes \Sigma$ is acyclic.*

6.2 Free Variables

So far, we have assumed that all queries are Boolean. In the construction of first-order definitions, we will have to treat queries with free variables. In the following, the

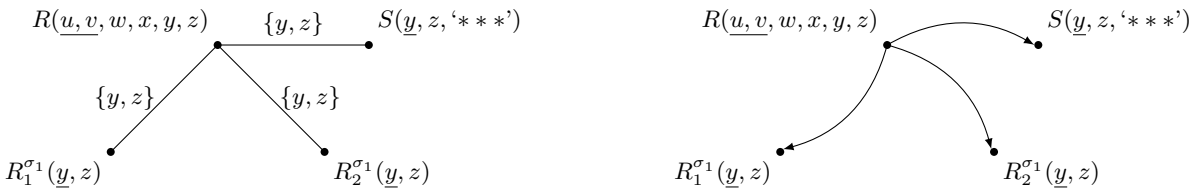

Figure 5: Join tree (left) and attack graph (right) of query $q_1 \otimes \{\sigma_1\}$ with $\sigma_1 = R : 5 \to 6$.

notation $q(\vec{u})$, where $\vec{u}$ is a sequence of distinct variables, indicates that the variables in $\vec{u}$ are free in the query q. The problem of certain query answering naturally extends to queries with free variables, as follows.

Definition 7. Let $q(\vec{u})$ be a conjunctive query with free variables $\vec{u}$. Let Σ be a set of first-order constraints. The function problem $\mathsf{CERTAINTY}(q(\vec{u}), \Sigma)$ takes on input an uncertain database $\mathbf{db}$ that satisfies Σ, and asks to return all *certain answers to q on* $\mathbf{db}$, i.e., all sequences $\vec{a}$ of constants (of the same length as $\vec{u}$) such that $q(\vec{a})$ is true in every repair of $\mathbf{db}$.

A *first-order definition of* $\mathsf{CERTAINTY}(q(\vec{u}), \Sigma)$ is a first-order formula $\varphi(\vec{u})$ such that for every sequence $\vec{a}$ of constants (of the same length as $\vec{u}$), for every uncertain database $\mathbf{db}$ that satisfies Σ, the following are equivalent:

1. Every repair of $\mathbf{db}$ satisfies $q(\vec{a})$.
2. $\mathbf{db} \models \varphi(\vec{a})$.

We explain that the addition of free variables is not fundamental. Assume we are asked to determine a first-order definition of $\mathsf{CERTAINTY}(q(\vec{u}), \Sigma)$ where $\vec{u} = \langle u_1, \ldots, u_m \rangle$. Let $c_1, \ldots, c_m$ be distinct constants not occurring in q, and let $\vec{c} = \langle c_1, \ldots, c_m \rangle$. Let $q_{\vec{u} \mapsto \vec{c}}$ be the query obtained from q by replacing each occurrence of u_i with c_i, for $i \in \{1, \ldots, m\}$. Let φ be a first-order definition of $\mathsf{CERTAINTY}(q_{\vec{u} \mapsto \vec{c}}, \Sigma)$. Clearly, the query $q_{\vec{u} \mapsto \vec{c}}$ is Boolean. Since first-order definitions treat all constants in a generic fashion, a first-order definition of $\mathsf{CERTAINTY}(q(\vec{u}), \Sigma)$ is obtained from φ by replacing each c_i with u_i. This is tantamount to saying that free variables are treated as constants. Likewise, in the computation of join trees and attack graphs, free variables are to be treated as constants.

6.3 The Rewrite Function

We introduce a function $\mathsf{Rewrite}$ that returns consistent first-order rewritings. We provide an intuitive example before giving the technical definition.

Example 14. Let $q = \exists x \exists y R(\underline{x}, a, x, y, y)$, where a is a constant. Obviously, q is true in every repair of an uncertain database $\mathbf{db}$ if and only if $\mathbf{db}$ contains an R-fact of the form $R(\underline{x}, a, x, y, y)$ and for all z_1, z_2, z_3, z_4, if the key-equal R-fact $R(\underline{x}, z_1, z_2, z_3, z_4)$ belongs to $\mathbf{db}$, then it is the case that $z_1 = a$, $z_2 = x$, and $z_3 = z_4$.

Notice that we can rename z_3 into y in the previous sentence, which results in the condition: for all z_1, z_2, y, z_4, if the key-equal R-fact $R(\underline{x}, z_1, z_2, y, z_4)$ belongs to $\mathbf{db}$, then it is the case that $z_1 = a$, $z_2 = x$, and $y = z_4$.

Figure 6: The attack graph of $(q_1, \{R : 5 \to 6\})$.

Thus, q is true in every repair of an uncertain database $\mathbf{db}$ if and only if $\mathbf{db}$ satisfies the following first-order sentence.

$$\exists x \exists y \Big(R(\underline{x}, a, x, y, y) \wedge \forall z_1 \forall z_2 \forall y \forall z_4 \big(R(\underline{x}, z_1, z_2, y, z_4) \to \left[\begin{array}{r} z_1 = a \\ \wedge z_2 = x \\ \wedge y = z_4 \end{array} \right] \big) \Big)$$

Definition 8. Let $q(\vec{u})$ be an acyclic SJFCQ query. Let Σ be a jd-singular set of KJDs and FDs such that the attack graph of $(q(\vec{u}), \Sigma)$ is acyclic. We define $\mathsf{Rewrite}(q(\vec{u}), \Sigma)$ recursively as follows.

Case $q = \emptyset$ (hence $\Sigma = \emptyset$). Then $\mathsf{Rewrite}(q, \Sigma) = \mathbf{true}$.

Case $q \neq \emptyset$. Choose an atom $F = R(\underline{\vec{x}}, y_1, \ldots, y_m)$ that is unattacked in the attack graph of (q, Σ). Let $z_1, \ldots, z_m$ be distinct variables and let C be a conjunction of equalities constructed as follows. For $i \in \{1, \ldots, m\}$,

- if y_i is a variable that does not occur in $\vec{u}$ nor in $\langle \vec{x}, y_1, \ldots, y_{i-1} \rangle$, then z_i is the same variable as y_i;
- otherwise z_i is a new variable and C contains $z_i = y_i$. Notice that this case applies if y_i is a constant, if y_i is a (free) variable in $\vec{u}$, or if y_i is a variable occurring in $\langle \vec{x}, y_1, \ldots, y_{i-1} \rangle$.

Let $\vec{v}$ be a sequence of variables that contains exactly once each variable that occurs in $\langle \vec{x}, y_1, \ldots, y_m \rangle$ and that does not occur in $\vec{u}$. Then

$$\begin{aligned} \mathsf{Rewrite}(q(\vec{u}), \Sigma) \quad = \quad & \exists \vec{v} \Big(R(\underline{\vec{x}}, y_1, \ldots, y_m) \wedge \\ & \forall z_1 \cdots \forall z_m \big(R(\underline{\vec{x}}, z_1, \ldots, z_m) \to \\ & \quad C \wedge \mathsf{Rewrite}(q'(\vec{u}, \vec{v}), \Sigma') \big) \Big). \end{aligned}$$

Here, $q' = q \setminus \{R(\underline{\vec{x}}, y_1, \ldots, y_m)\}$ and Σ' is the restriction of Σ to the KJDs and FDs that do not involve the relation name R. Notice that the variables of $\vec{u}$ remain free in $\mathsf{Rewrite}(q(\vec{u}), \Sigma)$.

Notice that from [32, Lemmas C.1 and C.2] and Lemma 4, it follows that the attack graph of $(q'(\vec{u}, \vec{v}), \Sigma')$ is acyclic and hence the recursive call $\mathsf{Rewrite}(q'(\vec{u}, \vec{v}), \Sigma')$ is well defined.

THEOREM 4. *Let $q(\vec{u})$ and Σ be as in Definition 8. If $\mathsf{CERTAINTY}(q(\vec{u}), \Sigma)$ is first-order expressible, then a first-order definition of it is given by $\mathsf{Rewrite}(q(\vec{u}), \Sigma)$.*

R	First	Last	Birth	Sal	City	Country
	Ed	Smith	1960	50K	Mons	Belgium
	An	Allen	1970	40K	Mons	France

S	City	Country	Stars
	Mons	Belgium	* * *
	Mons	France	* * *

Figure 7: Uncertain database falsifying $R : \mathbf{City} \to \mathbf{Country}$.

```
SELECT R.SAL FROM R, S
WHERE R.CITY = S.CITY
  AND R.COUNTRY = S.COUNTRY
  AND S.STARS = '***'
```

Figure 8: Original SQL query in the experiment, asking for salaries of employees who live in three-star cities.

Example 15. We can now explain all technical details behind Example 1 introduced in Section 1. Figure 3 shows the attack graph of q_1. Since the attack graph is cyclic, we conclude (by Theorem 2) that $\mathsf{CERTAINTY}(q_1)$ is not first-order expressible. Figure 5 shows the join tree (left) and the attack graph (right) of $q_1 \otimes \{R : 5 \to 6\}$. Since the attack graph is acyclic, we conclude (by Theorem 3) that $\mathsf{CERTAINTY}(q_1, \{R : 5 \to 6\})$ is first-order expressible.

The attack graph of $(q_1, \{R : 5 \to 6\})$ is shown in Figure 6. Based on this attack graph, $\mathsf{Rewrite}(q_1, \{R : 5 \to 6\})$ yields the following first-order sentence.

$$\varphi_1 = \exists u \exists v \exists w \exists x \exists y \exists z \Big(R(\underline{u, v}, w, x, y, z) \wedge \forall w \forall x \forall y \forall z \Big(R(\underline{u, v}, w, x, y, z) \to \big(S(\underline{y}, z, \text{`}***\text{'}) \wedge \forall z_1 \forall z_2 (S(\underline{y}, z_1, z_2) \to z_1 = z \wedge z_2 = \text{`}***\text{'}) \big) \Big) \Big)$$

By Theorem 4, the sentence φ_1 is a first-order definition of $\mathsf{CERTAINTY}(q_1, \{R : 5 \to 6\})$. Thus, for every uncertain database **db** that satisfies $R : 5 \to 6$, it is the case that φ_1 evaluates to true on **db** if and only if q_1 evaluates to true on every repair of **db**.

Unsurprisingly, φ_1 does not provide certain answers on uncertain databases that *falsify* $R : 5 \to 6$. For example, φ_1 evaluates to false on the uncertain database of Figure 7, even though all repairs of this database satisfy q_1.

It is fairly straightforward to translate a first-order definition of $\mathsf{CERTAINTY}(q, \Sigma)$ into SQL. The performance of such SQL queries has been studied in [12] and will be further discussed in the next section.

7. EXPERIMENTAL EVIDENCE

ConQuer [14] and EQUIP [18] are two systems for solving the problem $\mathsf{CERTAINTY}(q(\vec{u}))$ where $q(\vec{u})$ is a conjunctive query. ConQuer applies only to conjunctive queries $q(\vec{u})$ for which $\mathsf{CERTAINTY}(q(\vec{u}))$ is first-order expressible. ConQuer rewrites such a query $q(\vec{u})$ into a new SQL query Q that yields 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 $q(\vec{u})$. When an uncertain database **db** is given as the input of the problem $\mathsf{CERTAINTY}(q(\vec{u}))$, 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.

```
SELECT R1.SAL FROM R AS R1
WHERE NOT EXISTS (
      SELECT * FROM R AS R2
      WHERE R2.FIRST = R1.FIRST
      AND   R2.LAST = R1.LAST
      AND  (R2.SAL <> R1.SAL
      OR    NOT EXISTS (
            SELECT * FROM S AS S1
            WHERE S1.CITY = R2.CITY
            AND   S1.COUNTRY = R2.COUNTRY
            AND   S1.STARS = '***'
            AND   NOT EXISTS (
                  SELECT * FROM S AS S2
                  WHERE S2.CITY = S1.CITY
                  AND  (S2.COUNTRY <> S1.COUNTRY
                  OR    S2.STARS <> '***' )))))
```

Figure 9: Consistent first-order SQL rewriting.

No benchmark experiments exist for comparing the performance of different systems. Devising a representative benchmark is arduous because of the many factors that can impact the performance. It is not even clear how to adequately characterize the degree of inconsistency of an uncertain database. Call a *block* a maximal set of key-equal facts. Is it important to take into account the distribution of the cardinalities of blocks, or is it sufficient to know the percentage of tuples involved in some conflict?

Nevertheless, extensive experiments [18, 24] show that if $\mathsf{CERTAINTY}(q(\vec{u}))$ is first-order expressible, 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 $\mathbf{AC}^0$. A main conclusion of [24] is that *consistent first-order rewriting should be used whenever possible*. In this respect, Theorem 1 is of much practical importance, because it reveals when exactly consistent first-order rewriting is possible, i.e., when the problem can be solved in SQL. Unlike ConQuer, our approach takes into account knowledge about satisfied constraints.

To further illustrate the practicality of our approach, we ran an experiment for the following non-Boolean query q_4, which asks for salaries of employees who live in three-star cities; its SQL encoding is shown in Figure 8. This non-Boolean variant of the Boolean query q_1 was chosen in order to get more output than a single Boolean value.

$$q_4(x) = \exists u \exists v \exists w \exists y \exists z \big(R(\underline{u, v}, w, x, y, z) \wedge S(\underline{y}, z, \text{`}***\text{'}) \big)$$

It is **coNP**-complete to decide, given an uncertain database **db** and a salary c on input, whether c is a certain answer to q_4 on **db** [17]. Therefore, any known algorithm (be it EQUIP or another) which computes all certain answers will be exponential-time in the size of the input database. An

obvious (but impractical) approach is to execute the query of Figure 8 on every repair and take the intersection of all answers.

Let us now assume that the input databases are known to satisfy the functional dependency $R : \mathbf{City} \rightarrow \mathbf{Country}$ (call it σ_1). Using the results in this paper, it can be verified that $\mathsf{CERTAINTY}(q_4(x), \{\sigma_1\})$ is first-order expressible, and is solved by executing the SQL query of Figure 9.

In our experiment, we have assumed a fixed relation S of 1000 tuples such that in the **City**-column, 100 cities appear twice and 800 cities appear exactly once. These figures are not unrealistic if we consider that touristic appreciations are volatile. Obviously, the relation S has an astronomical number (2^{100}) of repairs. We have used different databases where the cardinality of the relation R varies from 500,000 to 1,500,000, and contains one duplicate primary key for every 1000 tuples. The relation R always satisfies $R : \mathbf{City} \rightarrow \mathbf{Country}$.

Table 1 shows query execution times in this experiment. All experiments were conducted using PostgreSQL version 9.1.6 on a machine with Intel core i5 2.4GHz CPU and 2GB RAM, running Gentoo Linux. The second column lists the execution time (denoted t_{rew}) of the rewritten query (see Figure 9) on the original inconsistent database. The third column shows an optimistic estimate of the execution time (denoted t_{o}) of the original query (see Figure 8) on any repair. Since it was verified that execution times do not significantly differ from one repair to another, we have measured the execution time on 100 arbitrarily picked repairs and report the minimum. Obviously, executing the original query (of Figure 8) on each repair is infeasible.

The rightmost column in Table 1 indicates that the ratio $t_{\mathsf{rew}}/t_{\mathsf{o}}$ is constantly less than 8. That is, in this experiment, executing the consistent first-order SQL rewriting on the original inconsistent database is less than eight times slower than executing the original query *on a single repair*. Thus, in the extreme case where the original uncertain database had only one single repair (i.e., if it were consistent), our proposed technique would only result in a factor 8 slow-down. Note, however, that under the (not unrealistic) assumption of a fixed percentage of duplicates (1‰ in our experiment), the number of repairs grows *exponentially* in the size of the database, whereas our technique is not affected by the presence of such duplicates.

The ease and efficiency of consistent first-order SQL rewriting do not incite to further "manual" optimization in addition to the automated optimization already performed by the DBMS optimizer. Such manual optimization could possibly come from a separate treatment of the consistent part of the database, or even from some normalization that decomposes the database schema into consistent and inconsistent relations. Such optimizations have not yet been investigated in theory and risk to incur significant overhead in practice.

To conclude, no previously known technique is capable of recognizing that the problem in our experiment can be solved by a single SQL query. ConQuer cannot handle the problem. EQUIP uses an exponential-time algorithm and incurs significant data preprocessing [18]. Our consistent first-order SQL rewriting is in the low complexity class $\mathbf{AC}^0$ and is insensitive to database updates.

$\lvert R\rvert$	t_{rew}	t_{o}	$t_{\mathsf{rew}}/t_{\mathsf{o}}$
500,000	953	133	7.12
600,000	1,156	160	7.21
700,000	1,356	183	7.38
800,000	1,606	212	7.56
900,000	1,767	238	7.40
1,000,000	1,965	265	7.40
1,100,000	2,219	295	7.50
1,200,000	2,341	319	7.32
1,300,000	2,532	346	7.31
1,400,000	2,699	367	7.33
1,500,000	2,980	396	7.51

Table 1: Execution times (in milliseconds) on uncertain databases of growing size. The relation S is fixed. t_{rew} is the execution time of the consistent first-order SQL rewriting on the uncertain database. t_{o} is the minimum of the execution times of the original query on 100 arbitrarily picked repairs.

8. CONCLUSION

The problem of consistent query answering under primary keys, also known as $\mathsf{CERTAINTY}(q)$, has attracted much research attention in recent years. This problem takes as its input an uncertain database **db** and asks whether the Boolean query q evaluates to true on every repair of **db**. In practical situations, however, one may know that input databases satisfy some set Σ of constraints, i.e., that the input databases are partially consistent. The problem $\mathsf{CERTAINTY}(q, \Sigma)$ takes as its input an uncertain database **db** *that satisfies* Σ and asks whether the query q evaluates to true on every repair of **db**. The knowledge that some constraints be satisfied brings a new flavor of practical interest to consistent query answering.

We studied the problem $\mathsf{CERTAINTY}(q, \Sigma)$ in case q is an acyclic Boolean SJFCQ query and Σ is a set of FDs and KJDs, containing at most one KJD per relation name. The main result is that it is decidable whether $\mathsf{CERTAINTY}(q, \Sigma)$ is first-order expressible. If $\mathsf{CERTAINTY}(q, \Sigma)$ is first-order expressible, then it can be solved by a single SQL query.

9. REFERENCES

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

[2] F. N. Afrati and P. G. Kolaitis. Repair checking in inconsistent databases: algorithms and complexity. In R. Fagin, editor, *ICDT*, volume 361 of *ACM International Conference Proceeding Series*, pages 31–41. ACM, 2009.

[3] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pages 68–79. ACM Press, 1999.

[4] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30(3):479–513, 1983.

[5] L. E. Bertossi. *Database Repairing and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[6] L. E. Bertossi, L. Bravo, E. Franconi, and A. Lopatenko. The complexity and approximation of fixing numerical attributes in databases under integrity constraints. *Inf. Syst.*, 33(4-5):407–434, 2008.

[7] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In Özcan [23], pages 143–154.

[8] L. Bravo and L. E. Bertossi. Consistent query answering under inclusion dependencies. In H. Lutfiyya, J. Singer, and D. A. Stewart, editors, *CASCON*, pages 202–216. IBM, 2004.

[9] J. Chomicki and J. Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1-2):90–121, 2005.

[10] N. N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: diamonds in the dirt. *Commun. ACM*, 52(7):86–94, 2009.

[11] N. N. Dalvi, C. Re, and D. Suciu. Queries and materialized views on probabilistic databases. *J. Comput. Syst. Sci.*, 77(3):473–490, 2011.

[12] A. Decan, F. Pijcke, and J. Wijsen. Certain conjunctive query answering in SQL. In E. Hüllermeier, S. Link, T. Fober, and B. Seeger, editors, *SUM*, volume 7520 of *Lecture Notes in Computer Science*, pages 154–167. Springer, 2012.

[13] S. Flesca, F. Furfaro, and F. Parisi. Querying and repairing inconsistent numerical databases. *ACM Trans. Database Syst.*, 35(2), 2010.

[14] A. Fuxman, E. Fazli, and R. J. Miller. ConQuer: Efficient management of inconsistent databases. In Özcan [23], pages 155–166.

[15] A. Fuxman and R. J. Miller. First-order query rewriting for inconsistent databases. In T. Eiter and L. Libkin, editors, *ICDT*, volume 3363 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 2005.

[16] A. Fuxman and R. J. Miller. First-order query rewriting for inconsistent databases. *J. Comput. Syst. Sci.*, 73(4):610–635, 2007.

[17] P. G. Kolaitis and E. Pema. A dichotomy in the complexity of consistent query answering for queries with two atoms. *Inf. Process. Lett.*, 112(3):77–85, 2012.

[18] P. G. Kolaitis, E. Pema, and W.-C. Tan. Efficient querying of inconsistent databases with binary integer programming. *PVLDB*, 6(6):397–408, 2013.

[19] P. Koutris and D. Suciu. A dichotomy on the complexity of consistent query answering for atoms with simple keys. *CoRR*, abs/1212.6636, 2012.

[20] A. Lopatenko and L. E. Bertossi. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In T. Schwentick and D. Suciu, editors, *ICDT*, volume 4353 of *Lecture Notes in Computer Science*, pages 179–193. Springer, 2007.

[21] D. Maslowski and J. Wijsen. A dichotomy in the complexity of counting database repairs. *J. Comput. Syst. Sci.*, 79(6):958–983, 2013.

[22] C. Molinaro and S. Greco. Polynomial time queries over inconsistent databases with functional dependencies and foreign keys. *Data Knowl. Eng.*, 69(7):709–722, 2010.

[23] F. Özcan, editor. *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*. ACM, 2005.

[24] E. Pema. *Consistent Query Answering of Conjunctive Queries under Primary Key Constraints*. PhD thesis, University of California Santa Cruz, 2013.

[25] B. ten Cate, G. Fontaine, and P. G. Kolaitis. On the data complexity of consistent query answering. In A. Deutsch, editor, *ICDT*, pages 22–33. ACM, 2012.

[26] J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I*. Computer Science Press, 1988.

[27] J. Wijsen. Database repairing using updates. *ACM Trans. Database Syst.*, 30(3):722–768, 2005.

[28] J. Wijsen. Project-join-repair: An approach to consistent query answering under functional dependencies. In H. L. Larsen, G. Pasi, D. O. Arroyo, T. Andreasen, and H. Christiansen, editors, *FQAS*, volume 4027 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[29] J. Wijsen. On the consistent rewriting of conjunctive queries under primary key constraints. *Inf. Syst.*, 34(7):578–601, 2009.

[30] J. Wijsen. On the first-order expressibility of computing certain answers to conjunctive queries over uncertain databases. In J. Paredaens and D. V. Gucht, editors, *PODS*, pages 179–190. ACM, 2010.

[31] J. Wijsen. A remark on the complexity of consistent conjunctive query answering under primary key violations. *Inf. Process. Lett.*, 110(21):950–955, 2010.

[32] J. Wijsen. Certain conjunctive query answering in first-order logic. *ACM Trans. Database Syst.*, 37(2):9, 2012.

[33] J. Wijsen. Charting the tractability frontier of certain conjunctive query answering. In R. Hull and W. Fan, editors, *PODS*, pages 189–200. ACM, 2013.