First-Order Under-Approximations of Consistent Query Answers

Floris Geerts[1], Fabian Pijcke[2], and Jef Wijsen[2]

[1] Universiteit Antwerpen, Antwerpen, Belgium
[2] Université de Mons, Mons, Belgium

Abstract. Consistent Query Answering (CQA) has by now been widely adopted as a principled approach for answering queries on inconsistent databases. The consistent answer of a query q on an inconsistent database $\mathbf{db}$ is the intersection of the answers to q on all repairs, where a repair is any consistent database that is maximally close to $\mathbf{db}$. Unfortunately, computing consistent answers under primary key constraints has already exponential data complexity for very simple conjunctive queries, which is completely impracticable.
In this paper, we propose a new framework for divulging an inconsistent database to end users, which adopts two access restrictions. The first restriction complies with CQA and states that inconsistencies should never be divulged to end users. Therefore, end users should only get consistent query answers. The second restriction states that the data complexity of user queries must remain tractable (i.e., in **P** or even in **FO**). User queries with exponential data complexity will be rejected. We investigate which consistent query answers can still be obtained under such access restrictions.

1 Introduction

Inconsistent, incomplete and uncertain data is widespread in the internet and social media era. This has given rise to a new paradigm for query answering, called *Consistent Query Answering* (CQA). This paradigm starts with the notion of *repair*, which is a new consistent database that minimally differs from the original inconsistent database. In general, an inconsistent database can have many repairs. In this respect, database repairing is different from data cleaning which aims at a unique cleaned database.

In this paper, we assume that the only constraints are primary keys, one per relation. A repair of an inconsistent database $\mathbf{db}$ is a maximal subset of $\mathbf{db}$ that satisfies all primary key constraints. Primary keys will be underlined. For example, the database of Fig. 1 stores ages and cities of residence of male and female persons. For simplicity, assume that persons have unique names (attribute N). Every person has exactly one age (attribute A) and city (attribute C). However, distinct tuples may agree on the primary key N, because there can be uncertainty about ages and cities. In the database of Fig. 1, there is uncertainty

M	$\underline{N}$	A	C
	Ed	48	Mons
	Ed	48	Paris
	Dirk	29	Mons

F	$\underline{N}$	A	C
	An	37	Mons
	Iris	37	Paris

Fig. 1. Example database with primary key violations.

about the city of Ed (it can be Mons or Paris). The database can be repaired in two ways: delete either $M(\underline{\text{Ed}}, 48, \text{Mons})$ or $M(\underline{\text{Ed}}, 48, \text{Paris})$.

When database repairing results in multiple repairs, CQA shifts from standard semantics to certainty semantics. Given a query, the *certain answer* (also called *consistent answer*) is defined as the intersection of the answers on all repairs. That is, for a query q on an inconsistent database $\mathbf{db}$, CQA replaces the standard query answer $q(\mathbf{db})$ with the certain answer, defined by the following intersection:

$$\bigcap\{q(\mathbf{r}) \mid \mathbf{r} \text{ is a repair of } \mathbf{db}\}. \tag{1}$$

Thus, the certainty semantics exclusively returns answers that hold true in every repair. Given a query q, we will denote by $\lfloor q \rfloor$ the query that maps a database to the answer defined by (1).

A practical obstacle to CQA is that the shift to certainty semantics involves a significant increase of complexity. When we refer to complexity in this paper, we mean data complexity, i.e., the complexity in terms of the size of the database (for a fixed query) [1, p. 422]. It is known for long [7] that there exist conjunctive queries q that join two relations such that the data complexity of $\lfloor q \rfloor$ is already **coNP**-hard. If this happens, CQA is completely impracticable.

This paper investigates ways to circumvent the high data complexity of CQA in a realistic setting, which is based on the following assumptions:

– If a query returns an answer to a user, then every tuple in that answer should belong to the certain answer. In Libkin's terminology [16], query answers must not contain *false positives*, i.e., tuples that are not certain.
– The only queries that can be executed in practice are those with data complexity in **P** or, even better, in **FO**. **FO** is the descriptive complexity class that captures all queries expressible in relational calculus.

Therefore, if the data complexity of a query $\lfloor q \rfloor$ is not in **P**, then the best we can go for is an approximation without false positives (also called under-approximation), computable in polynomial time. The term *strategy* will be used for queries that compute such approximations.

For readability, we depict our setting by the following scenario with two persons, called *Bob* and *Alice*. The person called *Bob* owns a database that is publicly accessible only via a query interface which restricts the syntax of the queries that can be asked. Our main results concern the case where the interface is restricted to self-join-free conjunctive queries. The database schema including all primary key constraints is publicly available. However, *Bob* is aware that his database contains many mistakes which should not be divulged. Therefore, whenever some end user asks a query q, *Bob* will actually execute the query $\lfloor q \rfloor$. That is, end users will get exclusively consistent answers. But, for feasibility reasons, *Bob* will reject any query q for which the data complexity of $\lfloor q \rfloor$ is too high. In this paper, we assume that *Bob* considers that data complexity is too high when it is beyond **FO**. The person called *Alice* interrogates *Bob*'s database, and she will be happy to get exclusively consistent answers. Unfortunately, her query q will be rejected by *Bob* if the data complexity of $\lfloor q \rfloor$ is too high (i.e., not in **FO**). If this happens, *Alice* has to change strategy. Instead of asking q, she can ask a finite number of queries $q_1, q_2, \dots, q_\ell$ such that for every $i \in \{1, \dots, \ell\}$, the data complexity of $\lfloor q_i \rfloor$ is in **FO**, and hence the query q_i will be accepted by *Bob*. No restriction is imposed on the number ℓ of queries that can be asked. The best *Alice* can hope for is that she can compute herself the answer to $\lfloor q \rfloor$ (or even to q) from *Bob*'s answers to $\lfloor q_1 \rfloor, \dots, \lfloor q_\ell \rfloor$. The question addressed in this paper is: Given that *Alice* wants to answer q, what queries should she ask to *Bob*?

Here is a concrete example. Assume *Bob* owns the database of Fig. 1. Interested in stable couples[3], *Alice* submits the query q_1 which asks "Get pairs of ages of men and women living in the same city":

$$q_1 = \{y, w \mid \exists x \exists u \exists z \, (M(\underline{x}, y, z) \wedge F(\underline{u}, w, z))\}.$$

The consistent answer is $\{(48, 37), (29, 37)\}$. However, the query $\lfloor q_1 \rfloor$ that returns the certain answer is known to have **coNP**-hard data complexity [14,13]. Therefore, *Bob* will reject q_1. *Alice* changes strategy and asks the query q_2 which asks "Get pairs of ages and city of men and women living in the same city":

$$q_2 = \{y, w, z \mid \exists x \exists u \, (M(\underline{x}, y, z) \wedge F(\underline{u}, w, z))\}. \tag{2}$$

Since the data complexity of $\lfloor q_2 \rfloor$ is known to be in **FO** [14,13], *Bob* will execute $\lfloor q_2 \rfloor$. The query q_2 returns {(29,37,Mons), (48,37,Mons)} on one repair, and {(29,37,Mons), (48,37,Paris)} on the other repair, so the certain answer is $\{(29, 37, \text{Mons})\}$. This in turn allows *Alice* to derive a certain answer to the original query: since $(29, 37, \text{Mons})$ belongs to the answer to $\lfloor q_2 \rfloor$, it is correct to conclude that $(29, 37)$ belongs to the answer to $\lfloor q_1 \rfloor$. An interesting question is whether *Alice* has a better strategy that divulges even more answers to $\lfloor q_1 \rfloor$.

The technical contributions of this paper are as follows. We first show that the following problem is undecidable: Given a relational calculus query q, is $\lfloor q \rfloor$ in **FO**? In view of this undecidability result, we then limit our attention to strategies that are first-order combinations (using disjunction and existential quantification) of queries $\lfloor q \rfloor$ that are known to be in **FO**. We show how to build optimal strategies under such syntax restrictions.

This paper is organized as follows. Section 2 discusses related work. Section 3 provides some mathematical definitions. Section 4 introduces our new framework for studying consistent query answering under primary key constraints, and introduces the problem OPTSTRATEGY. Intuitively, OPTSTRATEGY asks, given

[3] According to [6], marital stability is higher when the wife is 5+ years younger than her husband.

a query q, to find a new query q' that gets the largest subset of consistent answers while still obeying the restrictions imposed by our framework. Section 5 provides ways to solve $\mathsf{OPTSTRATEGY}$ in restricted settings. Finally, Section 6 concludes the paper.

2 Related Work

Consistent query answering (CQA) was proposed in [2] as a principled approach to handle data quality problems that arise from violations of integrity constraints. See the textbooks [10] and [3] for comprehensive overviews of these domains.

Fuxman and Miller [11] were the first ones to focus on CQA under the restrictions that consistency is only with respect to primary keys and that queries are self-join-free conjunctive. See [21] for a survey on consistent query answering to conjunctive queries under primary key constraints. Some recent results not covered by this survey can be found in [14,13].

Instead of returning the query answers true in every repair, one could return the query answers true in, e.g., a majority of repairs. This leads to the counting variant of CQA, which has been studied in [17,18]. As observed in [20], the counting variant of CQA under primary key constraints is closely related to query answering in block-independent-disjoint (BID) probabilistic databases [8,9]. Alternatively, one can obtain approximations by restricting the set of repairs. This approach has been considered in [5] in the setting of ontology-based data access.

Our work can also be regarded as querying "consistent views," in the sense that *Bob* returns exclusively consistent answers. It has been observed long ago [19] that consistent views are not closed under relational calculus. In other words, the position of the $\lfloor\cdot\rfloor$ construct in a query does matter. For example, the query $\{x \mid \exists y \exists z \lfloor M(\underline{x}, y, z)\rfloor\}$ returns only Dirk, while $\lfloor\{x \mid \exists y \exists z M(\underline{x}, y, z)\}\rfloor$ returns both Ed and Dirk. Bertossi and Li [4] have used views to protect the secrecy of data in a database. In our setting, the query answers that are to be hidden from end users are those that are not true in every repair.

3 Preliminaries

We assume disjoint sets of *variables* and *constants*. If $\boldsymbol{x}$ is a sequence containing variables and constants, then $\mathsf{vars}(\boldsymbol{x})$ 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.

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 $\{1, 2, \ldots, k\}$. We say that R is *all-key* if $n = k$. 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 $\boldsymbol{y} = s_{k+1}, \ldots, s_n$. An *$R$-fact* (or simply fact) is an R-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 databases, 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.

A *database* is a finite set $\mathbf{db}$ of facts using only the relation names of the schema. We often refer to databases as "uncertain databases" to stress that such databases can violate primary key constraints.

A *block* of $\mathbf{db}$ is a maximal set of key-equal facts of $\mathbf{db}$. The term R-block refers to a block of R-facts, i.e., facts with relation name R. An uncertain database $\mathbf{db}$ is *consistent* if no two distinct facts 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 $\mathbf{db}$. We write $\mathsf{rset}(\mathbf{db})$ for the set of repairs of $\mathbf{db}$.

Queries and Consistent Query Answering. We assume that the reader is familiar with *relational calculus* [1, Chapter 5] and with the notion of *queries* [15, Definition 2.7]. By **FO**, we denote the descriptive complexity class that contains the queries expressible in relational calculus.

For every m-ary ($m \geq 0$) relational calculus query q, we define $\lfloor q \rfloor$ as the m-ary query that maps every database $\mathbf{db}$ to $\bigcap\{q(\mathbf{r}) \mid \mathbf{r} \in \mathsf{rset}(\mathbf{db})\}$. Clearly, if $\mathbf{db}$ is a consistent database, then $\lfloor q \rfloor(\mathbf{db}) = q(\mathbf{db})$.

Given two m-ary queries q_1 and q_2, we say that q_1 is *contained in* q_2, denoted by $q_1 \sqsubseteq q_2$ if for every database $\mathbf{db}$, $q_1(\mathbf{db}) \subseteq q_2(\mathbf{db})$. We write $q_1 \sqsubset q_2$, if $q_1 \sqsubseteq q_2$ and $q_2 \not\sqsubseteq q_1$. We say that q_1 and q_2 are *equivalent*, denoted by $q_1 \equiv q_2$, if $q_1 \sqsubseteq q_2$ and $q_2 \sqsubseteq q_1$.

A 0-ary query is called Boolean. If q is a Boolean query, then q maps any database to either $\{\langle\rangle\}$ or $\{\}$, corresponding to **true** and **false** respectively.

A *conjunctive query* is a relational calculus query of the form $\{\boldsymbol{z} \mid \exists \boldsymbol{y} B\}$ where B is a conjunction of atoms. The conjunction B and the query are said to be *self-join-free* if no relation name occurs more than once in B. We write $\mathsf{vars}(B)$ for the set of variables that occur in B. By a slight abuse of notation, we denote by B also the set of conjuncts that occur in B. For example, if $B_1 = R(\boldsymbol{x}) \wedge R(\boldsymbol{x}) \wedge R(\boldsymbol{y})$ and $B_2 = R(\boldsymbol{x}) \wedge R(\boldsymbol{y}) \wedge R(\boldsymbol{z})$, then we may write $B_1 \subseteq B_2$.

Significantly, the following example shows that $\lfloor q \rfloor$ may not be expressible in relational calculus, even if q is self-join-free conjunctive.

Example 1. Let $q_1 = \{\langle\rangle \mid \exists x \exists y \exists z \left(R(\underline{x}, z) \wedge S(\underline{y}, z)\right)\}$. The query q_1 is self-join-free conjunctive. It follows from [13] that $\lfloor q_1 \rfloor$ is not in **FO** (i.e., not expressible in relational calculus).

Let $q_2 = \{\langle\rangle \mid \exists x \exists y \left(R(\underline{x}, y) \land S(\underline{y}, b)\right)\}$, where b is a constant. Then, $\lfloor q_2 \rfloor$ is equivalent to the following relational calculus query:

$$\begin{aligned}&\exists x \exists y (R(\underline{x}, y) \land \\ &\quad \forall y \left(R(\underline{x}, y) \rightarrow \left(S(\underline{y}, b) \land \forall z \left(S(\underline{y}, z) \rightarrow z = b\right)\right)\right)) .\end{aligned} \qquad \square$$

4 A Framework for Divulging Inconsistent Databases

In this section, we formalize the setting that was described and illustrated in Section 1. The setting is captured by the language called CQAFO, which consists of first-order quantification and Boolean combinations of atomic formulas of the form $\lfloor q \rfloor$, where q is any relational calculus query. The atomic formulas $\lfloor q \rfloor$ capture that the database owner *Bob* only returns certain answers. Subsequently, the end user *Alice*, who interrogates *Bob*'s database, can do some post-processing on *Bob*'s outputs. In our setting, we assume that *Alice* uses first-order quantification and Boolean combinations of *Bob*'s answers.

Example 2. The scenario in Section 1 is captured by the CQAFO query

$$\{y, w \mid \exists Z \lfloor \exists x \exists u \left(M(\underline{x}, y, Z) \land F(\underline{u}, w, Z)\right) \rfloor\}.$$

The formula within $\lfloor\cdot\rfloor$ is the query (2). The quantification $\exists Z$ corresponds to *Alice* projecting away the cities column returned by *Bob*. For readability, we will often use upper case letters for variables that are quantified outside the range of $\lfloor\cdot\rfloor$. $\square$

Example 3. The following query allows *Alice* to find the names of men with more than two cities in the database:

$$\{x \mid \lfloor \exists y \exists z M(\underline{x}, y, z) \rfloor \land \neg \exists Z \lfloor \exists y M(\underline{x}, y, Z) \rfloor\}.$$

To understand this query, it may be helpful to notice that $\{x, Z \mid \lfloor \exists y M(\underline{x}, y, Z) \rfloor\}$ returns tuple (n, c) whenever c is the only city of residence encoded for the person named n. $\square$

4.1 The Language CQAFO

Syntax of CQAFO

- If q is a relational calculus query, then $\lfloor q \rfloor$ is a CQAFO formula.
- If φ_1 and φ_2 are CQAFO formulas, then $\varphi_1 \land \varphi_2$, $\varphi_1 \lor \varphi_2$, and $\neg\varphi_1$ are CQAFO formulas.
- If φ is a CQAFO formula, then $\exists x \varphi$ and $\forall x \varphi$ are CQAFO formulas.

If φ is a CQAFO formula, then $\mathsf{free}(\varphi)$ denotes the set of free variables of φ (i.e., the variables not bound by a quantifier). If $\boldsymbol{x}$ is a tuple containing all the free variables of φ, we write $\varphi(\boldsymbol{x})$.

A CQAFO query is an expression of the form $\{\boldsymbol{x} \mid \varphi\}$, where $\boldsymbol{x}$ is a sequence of variables and constants containing each variable of $\mathsf{free}(\varphi)$. If $\boldsymbol{x}$ contains no constants and no double occurrences of the same variable, then such query is also denoted $\varphi(\boldsymbol{x})$.

Semantics Let $\mathbf{db}$ be an uncertain database. Let $\varphi(\boldsymbol{x})$ be a CQAFO formula, and $\boldsymbol{a}$ be a sequence of constants (of same length as $\boldsymbol{x}$). We inductively define $\mathbf{db} \models \varphi(\boldsymbol{a})$.

- If $\varphi(\boldsymbol{x}) = \lfloor q(\boldsymbol{x}) \rfloor$ for some relational calculus query $q(\boldsymbol{x})$, then $\mathbf{db} \models \varphi(\boldsymbol{a})$ if for every repair $\mathbf{r}$ of $\mathbf{db}$, $\mathbf{r} \models q(\boldsymbol{a})$;[4]
- $\mathbf{db} \models \neg\varphi(\boldsymbol{a})$ if $\mathbf{db} \not\models \varphi(\boldsymbol{a})$;
- $\mathbf{db} \models \varphi_1 \wedge \varphi_2$ if $\mathbf{db} \models \varphi_1$ and $\mathbf{db} \models \varphi_2$;
- $\mathbf{db} \models \varphi_1 \vee \varphi_2$ if $\mathbf{db} \models \varphi_1$ or $\mathbf{db} \models \varphi_2$;
- if $\psi(\boldsymbol{x}) = \exists y \varphi(y, \boldsymbol{x})$, then $\mathbf{db} \models \psi(\boldsymbol{a})$ if $\mathbf{db} \models \varphi(a', \boldsymbol{a})$ for some a';
- if $\psi(\boldsymbol{x}) = \forall y \varphi(y, \boldsymbol{x})$, then $\mathbf{db} \models \psi(\boldsymbol{a})$ if $\mathbf{db} \models \varphi(a', \boldsymbol{a})$ for all a'.

Let $Q = \{\boldsymbol{x}' \mid \varphi(\boldsymbol{x})\}$ be a CQAFO query. The answer $Q(\mathbf{db})$ is the smallest set containing $\theta(\boldsymbol{x}')$ for every valuation θ over $\mathsf{vars}(\boldsymbol{x})$ such that for some $\boldsymbol{a}$, $\theta(\boldsymbol{x}) = \boldsymbol{a}$ and $\mathbf{db} \models \varphi(\boldsymbol{a})$. Notice that $\mathsf{vars}(\boldsymbol{x}') = \mathsf{vars}(\boldsymbol{x})$, but $\boldsymbol{x}'$, unlike $\boldsymbol{x}$, can contain constants and multiple occurrences of the same variable. If $\boldsymbol{x}'$ contains no variables, then Q is Boolean.

4.2 Restrictions on Data Complexity

The language CQAFO of Section 4.1 captures our first access restriction which states that the database owner *Bob* returns exclusively certain answers. But we do not prohibit that end user *Alice* does some post-processing on *Bob*'s answers. In this section, we will add our second access restriction which states that *Bob* rejects queries q if the data complexity of $\lfloor q \rfloor$ is not in $\mathbf{FO}$. Unfortunately, *Bob* has to face the following undecidability result.

Theorem 1. *The following problem is undecidable. Given a relational calculus query q, is $\lfloor q \rfloor$ in $\mathbf{FO}$?*

Proof. Let $q_1 = \{\langle\rangle \mid \exists x \exists y \exists z \left(R(\underline{x}, z) \wedge S(\underline{y}, z) \wedge \varphi\right)\}$ where φ is a closed relational calculus formula such that all relation names in φ are all-key. We show hereinafter that $\lfloor q \rfloor$ is in $\mathbf{FO}$ if and only if φ is unsatisfiable. The desired result then follows by [1, Theorem 6.3.1], which states that (finite) satisfiability of relational calculus queries is undecidable.

Obviously, if φ is unsatisfiable, then $\lfloor q_1 \rfloor \equiv \textbf{false}$, and hence $\lfloor q_1 \rfloor$ is in $\mathbf{FO}$.

We show next that if φ is satisfiable, then $\lfloor q_1 \rfloor$ is not in $\mathbf{FO}$. Assume that φ is satisfiable. Let $q_0 = \exists x \exists y \exists z \left(R(\underline{x}, z) \wedge S(\underline{y}, z)\right)$. Let $\mathsf{CERTAIN0}$ and $\mathsf{CERTAIN1}$ be the problems defined next.

- $\mathsf{CERTAIN0}$: Given a database $\mathbf{db}$, determine whether every repair of $\mathbf{db}$ satisfies q_0.
- $\mathsf{CERTAIN1}$: Given a database $\mathbf{db}$, determine whether every repair of $\mathbf{db}$ satisfies q_1.

[4] $\mathbf{r} \models q(\boldsymbol{a})$ is defined in the standard way.

Let $\mathbf{db}_0$ be a database that is input to $\mathsf{CERTAIN0}$. We show a polynomial-time many-one reduction from $\mathsf{CERTAIN0}$ to $\mathsf{CERTAIN1}$. Let $\mathbf{S}$ be the database schema that contains the relation names occurring in φ. An algorithm can consider systematically every finite database $\mathbf{db}'$ over $\mathbf{S}$ and test $\mathbf{db}' \models \varphi$, until a database $\mathbf{db}'$ is found such that $\mathbf{db}' \models \varphi$. The algorithm terminates because φ is satisfiable. Since the computation of $\mathbf{db}'$ does not depend on $\mathbf{db}_0$, it takes $\mathcal{O}(1)$ time. Since all relation names in $\mathbf{db}'$ are all-key, we have that $\mathbf{db}'$ is consistent. Clearly, q_0 is true in every repair of $\mathbf{db}_0$ if and only if q_1 is true in every repair of $\mathbf{db}_0 \cup \mathbf{db}'$. So we have established a polynomial-time many-one reduction from $\mathsf{CERTAIN0}$ to $\mathsf{CERTAIN1}$. Since $\mathsf{CERTAIN0}$ is **coNP**-hard [13], it follows that $\mathsf{CERTAIN1}$ is **coNP**-hard. Since $\mathbf{FO} \subsetneq \mathbf{coNP}$ [12], it follows that $\mathsf{CERTAIN1}$ is not in **FO**. □

By Theorem 1, there exists no algorithm that allows *Bob* to decide whether he has to accept or reject a relational calculus query. In general, little is known about the complexity of $\lfloor q \rfloor$ for relational calculus queries q. One of the stronger known results is the following.

Theorem 2 ([13]). *The following problem is decidable in polynomial time. Given a self-join-free conjunctive query q, is $\lfloor q \rfloor$ in* **FO**? *Moreover, if $\lfloor q \rfloor$ is in* **FO**, *then a relational calculus query equivalent to $\lfloor q \rfloor$ can be effectively constructed.*

In view of Theorems 1 and 2, the following scenario is the best we can go for with the current state of art.

1. The database owner *Bob* only accepts self-join-free conjunctive queries q such that $\lfloor q \rfloor$ is in **FO**. Thus, *Bob* rejects every query that is not self-join-free conjunctive, and rejects a self-join-free conjunctive query q if $\lfloor q \rfloor$ is not in **FO**.
2. As before, *Alice* can do some first-order post-processing on the answers obtained from *Bob*.

Under these restrictions, we focus on the following research task: given that *Alice* wants to answer a self-join-free conjunctive query q on a database owned by *Bob*, develop a *strategy* for *Alice* to get a subset (the greater, the better) of certain answers. A formal definition follows.

Our framework applies to Boolean queries: $\{\langle\rangle\}$ (**true**) and $\{\}$ (**false**) are two under-approximations of $\{\langle\rangle\}$, while $\{\}$ is the only under-approximation of itself.

4.3 Strategies

Strategies for a query q are defined next as relational calculus queries that can be expressed in CQAFO and that are contained in $\lfloor q \rfloor$.

Definition 1. *Let q be a self-join-free conjunctive query. A* strategy for *q is a* CQAFO *query φ such that $\varphi \sqsubseteq \lfloor q \rfloor$ and for every atomic formula $\lfloor q' \rfloor$ in φ, we have that q' is a self-join-free conjunctive query such that $\lfloor q' \rfloor$ is in* **FO**.

A strategy φ for q is optimal *if for every strategy ψ for q, we have $\psi \sqsubseteq \varphi$. The problem* OPTSTRATEGY *takes in a self-join-free conjunctive query q and asks to determine an optimal strategy for q.*

Some observations are in place.

– If the input to OPTSTRATEGY is a self-join-free conjunctive q such that $\lfloor q \rfloor$ is in **FO**, then the CQAFO query $\lfloor q \rfloor$ is itself an optimal strategy.
– Every strategy φ is in **FO**, because all atomic formulas $\lfloor q' \rfloor$ are required to be in **FO**. Therefore, if *Alice* wants to answer a query q such that $\lfloor q \rfloor$ is not in **FO**, then there is no strategy φ such that $\varphi \equiv \lfloor q \rfloor$.
– There is no fundamental reason why the input query to OPTSTRATEGY is required to be self-join-free conjunctive query. However, developing strategies for more expressive queries is left as an open question.

5 How to Construct Good Strategies?

Let q be a self-join-free conjunctive query. In this section, we investigate ways for constructing good (if not optimal) strategies for q of a particular syntax. In Section 5.1, we take the most simple approach: take the union of queries $\lfloor q_i \rfloor$ contained in $\lfloor q \rfloor$, where q_i is self-join-free conjunctive and $\lfloor q_i \rfloor$ is in **FO**. We then show that the strategies obtained in this way cannot be optimal. Therefore, an enhanced approach is developed in Section 5.2.

5.1 Post-Processing by Unions Only

Assume that the input to OPTSTRATEGY is a self-join-free conjunctive query $q(\boldsymbol{z})$. In this section, we look at strategies of the form

$$\bigcup_{i=1}^{\ell} \lfloor q_i \rfloor, \tag{3}$$

where each q_i is of the form $\{\boldsymbol{z_i} \mid \exists \boldsymbol{y}_i B_i\}$ in which $\boldsymbol{z}_i$ has same length as $\boldsymbol{z}$ and B_i is a self-join-free conjunction of atoms.[5]

We use union (with its standard semantics) instead of disjunction to avoid notational difficulties. For example, the union

$$\{x, a \mid \lfloor R(\underline{x}, a) \rfloor\} \cup \{x, y \mid \lfloor S(\underline{x}, y) \rfloor\},$$

where a is a constant, is semantically clear, and is equivalent to

$$\{x, y \mid \lfloor R(\underline{x}, y) \land y = a \rfloor \lor \lfloor S(\underline{x}, y) \rfloor\},$$

[5] Notice that is can be easily verified that $\lfloor \{\boldsymbol{z_i} \mid \exists \boldsymbol{y}_i B_i\} \rfloor \equiv \{\boldsymbol{z_i} \mid \lfloor \exists \boldsymbol{y}_i B_i \rfloor\}$.

in which equality is needed. It would be wrong to write $\{x, y \mid \lfloor R(\underline{x}, a)\rfloor \vee \lfloor S(\underline{x}, y)\rfloor\}$, an expression that is even not domain independent [1, p. 79].

Clearly, a formula of the form (3) is a strategy if for every $i \in \{1, \dots, \ell\}$, $\lfloor q_i \rfloor$ is in **FO** and $\lfloor q_i \rfloor \sqsubseteq \lfloor q \rfloor$. The latter condition is equivalent to $q_i \sqsubseteq q$ as shown next.

Lemma 1. *Let q and q' be self-join-free m-ary conjunctive queries. Then, $q \sqsubseteq q'$ if and only if $\lfloor q \rfloor \sqsubseteq \lfloor q' \rfloor$.*

Proof. Let $q = \{\boldsymbol{z} \mid \exists \boldsymbol{y} B\}$ and $q' = \{\boldsymbol{z}' \mid \exists \boldsymbol{y}' B'\}$, where $\boldsymbol{z}$ and $\boldsymbol{z}'$ both have the same length m.

$\boxed{\Longrightarrow}$ Straightforward. $\boxed{\Longleftarrow}$ Assume $\lfloor q \rfloor \sqsubseteq \lfloor q' \rfloor$. Let μ be an injective mapping with domain $\mathsf{vars}(B)$ that maps each variable to a fresh constant not occurring elsewhere. Since μ is injective, its inverse μ^{-1} is well defined. Let $\mathbf{db} = \mu(B)$. Clearly, $\mathbf{db}$ is consistent and $q(\mathbf{db}) = \{\mu(\boldsymbol{z})\} = \lfloor q \rfloor(\mathbf{db})$. From $\lfloor q \rfloor \sqsubseteq \lfloor q' \rfloor$, it follows $\mu(\boldsymbol{z}) \in q'(\mathbf{db}) = \lfloor q' \rfloor(\mathbf{db})$. Then, there exists a valuation θ over $\mathsf{vars}(B')$ such that $\theta(B') \subseteq \mathbf{db}$ and $\theta(\boldsymbol{z}') = \mu(\boldsymbol{z})$. Then $\mu^{-1} \circ \theta(B') \subseteq B$ and $\mu^{-1} \circ \theta(\boldsymbol{z}') = \boldsymbol{z}$. Since $\mu^{-1} \circ \theta$ is a homomorphism from q' to q, it follows $q \sqsubseteq q'$ by the Homomorphism Theorem [1, Theorem 6.2.3]. □

Lemma 1 does not hold for conjunctive queries with self-joins, as shown next.

Example 4. Let $q = \{\langle\rangle \mid R(\underline{a}, b) \wedge R(\underline{a}, c)\}$. For every uncertain database $\mathbf{db}$, $\lfloor q \rfloor(\mathbf{db}) = \{\}$. Let q' be a query such that $q \not\sqsubseteq q'$ (such query obviously exists). Then, $\lfloor q \rfloor \sqsubseteq \lfloor q' \rfloor$ and $q \not\sqsubseteq q'$. □

Lemma 1 allows us to construct strategies of the form (3), as follows. Assume that the input to $\mathsf{OPTSTRATEGY}$ is a self-join-free conjunctive query $q(\boldsymbol{z})$. For some positive integer ℓ, generate self-join-free conjunctive queries $q_1, \dots, q_\ell$ such that for each $i \in \{1, \dots, \ell\}$, $q_i \sqsubseteq q$ and $\lfloor q_i \rfloor$ is in **FO**. The condition $q_i \sqsubseteq q$ is decidable by [1, Theorem 6.2.3]; the condition that $\lfloor q_i \rfloor$ is in **FO** is decidable by Theorem 2. Then by Lemma 1, $\bigcup_{i=1}^{\ell} \lfloor q_i \rfloor$ is a strategy for q.

Unfortunately, Theorem 3 given hereinafter states that there are cases where no strategy of the form (3) is optimal. We first generalize Lemma 1 to unions.

Lemma 2. *Let q_0, q_1, $\dots q_\ell$ be self-join-free m-ary conjunctive queries. Then, $\lfloor q_0 \rfloor \sqsubseteq \bigcup_{i=1}^{\ell} \lfloor q_i \rfloor$ if and only if for some $i \in \{1, \dots, \ell\}$, $q_0 \sqsubseteq q_i$.*

Proof. $\boxed{\Longleftarrow}$ Straightforward. $\boxed{\Longrightarrow}$ Assume $\lfloor q_0 \rfloor \sqsubseteq \bigcup_{i=1}^{\ell} \lfloor q_i \rfloor$. Let $q_0 = \{\boldsymbol{z}_0 \mid \exists \boldsymbol{y}_0 B_0\}$, where B_0 is self-join-free. Let μ be an injective mapping with domain $\mathsf{vars}(B_0)$ that maps each variable to a fresh constant not occurring elsewhere. Since μ is injective, its inverse μ^{-1} is well defined. Let $\mathbf{db} = \mu(B_0)$. Clearly, $\mathbf{db}$ is consistent and $q_0(\mathbf{db}) = \{\mu(\boldsymbol{z}_0)\} = \lfloor q_0 \rfloor(\mathbf{db})$. From $\lfloor q_0 \rfloor \sqsubseteq \bigcup_{i=1}^{\ell} \lfloor q_i \rfloor$, it follows that we can assume $i \in \{1, \dots, \ell\}$ such that $\mu(\boldsymbol{z}_0) \in q_i(\mathbf{db}) = \lfloor q_i \rfloor(\mathbf{db})$. Let $q_i = \{\boldsymbol{z}_i \mid \exists \boldsymbol{y}_i B_i\}$. Then, there exists a valuation θ over $\mathsf{vars}(B_i)$ such that $\theta(B_i) \subseteq \mathbf{db}$ and $\theta(\boldsymbol{z}_i) = \mu(\boldsymbol{z}_0)$. Then $\mu^{-1} \circ \theta(B_i) \subseteq B_0$ and $\mu^{-1} \circ \theta(\boldsymbol{z}_i) = \boldsymbol{z}_0$. Since $\mu^{-1} \circ \theta$ is a homomorphism from q_i to q_0, it follows $q_0 \sqsubseteq q_i$. □

Theorem 3. *There exists a self-join-free conjunctive query q such that for every strategy φ of the form (3) for q, there exists another strategy ψ of the form (3) for q such that $\varphi \sqsubset \psi$.*

Proof. Let $q = \{\langle\rangle \mid \exists x \exists y \exists z \left(R(\underline{x}, z) \wedge S(\underline{y}, z)\right)\}$. Then $\lfloor q \rfloor$ is not in **FO** [14]. For every constant c, let q_c be the query defined by $\{\langle\rangle \mid \exists y \exists z \left(R(\underline{c}, z) \wedge S(\underline{y}, z)\right)\}$. For every constant c, we have that $\lfloor q_c \rfloor \sqsubseteq \lfloor q \rfloor$ and $\lfloor q_c \rfloor$ is in **FO**.

Let φ be a strategy for q of the form (3). Let A be the greatest set of constants such that for all $c \in A$, there exists some $i \in \{1, \dots, \ell\}$ such that $q_i \equiv q_c$. Let b be a constant such that $b \notin A$. Clearly $\varphi \sqsubseteq \varphi \cup \lfloor q_b \rfloor \sqsubseteq \lfloor q \rfloor$. It suffices to show that $\varphi \sqsubset \varphi \cup \lfloor q_b \rfloor$, meaning that φ is not optimal.

Assume towards a contradiction that $\lfloor q_b \rfloor \sqsubseteq \varphi$. By Lemma 2, there exists $i \in \{1, \dots, \ell\}$ such that $q_b \sqsubseteq q_i \sqsubseteq q$. Let q_i be the existential closure of $(R(\underline{s}, t) \wedge S(\underline{u}, v))$. From $q_i \sqsubseteq q$, it follows that $t = v$. From $q_b \sqsubseteq q_i$ and $b \notin A$, it follows that s, t, u are pairwise distinct variables. But then $q_i \equiv q$, contradicting that $\lfloor q_i \rfloor$ is in **FO**. We conclude by contradiction that $\varphi \sqsubset \varphi \cup \lfloor q_b \rfloor$. □

5.2 Post-Processing by Unions and Quantification

The proof of Theorem 3 indicates that strategies of the form (3) lack expressiveness because the number of constants in such strategies is bounded. An obvious extension is to look for strategies that replace constants with existentially quantified variables. The following example shows how such extension solves the lack of expressiveness that underlies the proof of Theorem 3.

Example 5. Let $q = \exists x \exists y \exists z \left(R(\underline{x}, z) \wedge S(\underline{y}, z)\right)$. Let φ be the CQAFO formula defined by $\varphi := \exists X \lfloor \exists y \exists z \left(R(\underline{X}, z) \wedge S(\underline{y}, z)\right) \rfloor$. It can be shown that φ is a strategy for q, i.e., $\varphi \sqsubseteq \lfloor q \rfloor$ and $\lfloor \exists y \exists z \left(R(\underline{X}, z) \wedge S(\underline{y}, z)\right) \rfloor$ is in **FO**. Recall from Example 2 that the use of upper case X is for readability. □

Assume that the input to OPTSTRATEGY is a self-join-free conjunctive query $q(\boldsymbol{z})$. In this section, we investigate strategies of the form

$$\bigcup_{i=1}^{\ell} Q_i, \tag{4}$$

where for each $i \in \{1 \dots, \ell\}$, Q_i is a CQAFO query of the form

$$\{\boldsymbol{z}_i \mid \exists \boldsymbol{X}_i \lfloor \exists \boldsymbol{y}_i B_i \rfloor\}, \tag{5}$$

in which $\boldsymbol{z}_i$ has the same length as $\boldsymbol{z}$, and B_i is a self-join-free conjunction of atoms. It is understood that $\boldsymbol{z}_i$, $\boldsymbol{X}_i$, and $\boldsymbol{y}_i$ have, pairwise, no variables in common, and that $\mathsf{vars}(\boldsymbol{z}_i \boldsymbol{X}_i \boldsymbol{y}_i) = \mathsf{vars}(B_i)$. For readability, we will use upper case Q to refer to CQAFO queries of the form (5). The main tools for constructing strategies of the form (4) are provided by Theorems 4 and 5.

Theorem 4. *The following problem is decidable in polynomial time. Given a* CQAFO *query Q of the form (5), is Q in* **FO***? Moreover, if Q is in* **FO***, then a relational calculus query equivalent to Q can be effectively constructed.*

Proof. A CQAFO query Q of the form (5) is in **FO** if and only if $\lfloor \exists \boldsymbol{y}_i B_i \rfloor$ is in **FO**. The latter condition is decidable by Theorem 2.

Theorem 5. *Given a self-join-free conjunctive query q_1 and a* CQAFO *query Q_2 of the form (5), it can be decided whether $Q_2 \sqsubseteq \lfloor q_1 \rfloor$.*

Proof. (Crux.) Let $q_1 = \{\boldsymbol{z}_1 \mid \exists \boldsymbol{y}_1 B_1\}$ and $Q_2 = \{\boldsymbol{z}_2 \mid \exists \boldsymbol{X}_2 \lfloor \exists \boldsymbol{y}_2 B_2 \rfloor\}$. It can be shown that $Q_2 \sqsubseteq \lfloor q_1 \rfloor$ if and only if there exists a valuation θ over $\mathsf{vars}(B_1)$ such that $\theta(\boldsymbol{z}_1) = \boldsymbol{z}_2$ and $\theta(B_1) \subseteq B_2$. □

We point out that Theorem 5 is interesting in its own right. It is well known [1, Corollary 6.3.2] that containment of relational calculus queries is undecidable. A large fragment for which containment is decidable is the class of unions of conjunctive queries. Notice, however, that the queries in the statement of Theorem 5 need not be monotone (and even not first-order), and that decidability of query containment for such queries is not obvious.

Example 6. Let $Q = \{x \mid \exists Y \lfloor R(\underline{x}, Y) \rfloor\}$. Let $\mathbf{db} = \{R(\underline{a}, 1)\}$ and $\mathbf{db}' = \{R(\underline{a}, 1),\ R(\underline{a}, 2)\}$. Then $\mathbf{db} \subseteq \mathbf{db}'$, but $Q(\mathbf{db}) = \{a\}$ is not contained in $Q(\mathbf{db}) = \{\}$. Hence Q is not monotone. We have that Q is equivalent to the following relational calculus query:

$$\{x \mid \exists y\, (R(\underline{x}, y) \land \forall y'\, (R(\underline{x}, y') \rightarrow y = y'))\}. \qquad \square$$

Assume that the input to OPTSTRATEGY is a self-join-free conjunctive query $q(\boldsymbol{z})$. Theorem 5 allows us to build a strategy of the form (4) for q as follows. Let A be the set of constants that occur in q. Let φ be the disjunction of all (up to variable renaming) CQAFO formulas Q_i of the form (5) that use exclusively constants from A such that $Q_i \sqsubseteq \lfloor q \rfloor$ and Q_i is in **FO**. Clearly, there are at most finitely many such formulas (up to variable renaming). Containment of Q_i in $\lfloor q \rfloor$ is decidable by Theorem 5. Finally, the condition that Q_i is in **FO** is decidable by Theorem 4. The following theorem remedies the negative result of Theorem 3.

Theorem 6. *For every self-join-free conjunctive query q, there exists a computable strategy φ of the form (4) for q, such that for every strategy ψ of the form (4) for q, $\psi \sqsubseteq \varphi$.*

Proof. Assume that the input to OPTSTRATEGY is a self-join-free conjunctive query $q(\boldsymbol{z})$. Let φ be the strategy defined in the paragraph preceding this theorem. Let $Q = \{\boldsymbol{z}_0 \mid \exists \boldsymbol{X} \lfloor \exists \boldsymbol{y} B \rfloor\}$ be a query of the form (5) where B is a self-join-free conjunction of atoms such that Q is in **FO** and $Q \sqsubseteq \lfloor q \rfloor$. If all constants that occur in B also occur in q, then Q is already contained in some disjunct of φ (by construction of φ). Assume next that B contains some constants that do not occur in q, and let these constants be $a_1, \ldots, a_m$. For $i \in \{1, \ldots, m\}$, let X_i

be a new fresh variable. Let B' be the conjunction obtained from B by replacing each occurrence of each a_i with X_i. Let $Q' = \{\boldsymbol{z}_0 \mid \exists \boldsymbol{X} \exists X_1 \cdots \exists X_m \lfloor \exists \boldsymbol{y} B' \rfloor\}$. From the proof of Theorem 2, it follows $Q' \sqsubseteq \lfloor q \rfloor$. It can be easily seen that $Q \sqsubseteq Q'$. Furthermore, from [13], it follows that Q' is in **FO**. Since all constants that occur in B' also occur in q, we have that Q' is already contained in some disjunct of φ (by construction of φ).

To conclude, whenever $Q = \{\boldsymbol{z}_0 \mid \exists \boldsymbol{X} \lfloor \exists \boldsymbol{y} B \rfloor\}$ is a query of the form (5) where B is a self-join-free conjunction of atoms such that Q is in **FO** and $Q \sqsubseteq \lfloor q \rfloor$, we have that $\varphi \cup Q \sqsubseteq \varphi$. □

So far, we have imposed no restrictions on the size of the computable strategy φ in the statement of Theorem 6. From a practical point of view, it is interesting to construct, among all optimal strategies φ of the form (4), the one with the smallest number ℓ of disjuncts. It is an open question, however, how to minimize strategies of the form (4).

6 Conclusion

We have studied a realistic setting for divulging an inconsistent database to end users. In this setting, users access the database exclusively via syntactically restricted queries, and get exclusively consistent answers computable in **FO** data complexity. If the data complexity is higher, then the query will be rejected, in which case users have to fall back on strategies that obtain a large (the larger, the better) subset of the consistent answer. Such strategies combine answers obtained from several "easier" queries.

Although our setting applies to arbitrary queries and constraints, we searched for strategies when constraints are primary keys, and the database is accessible only via self-join-free conjunctive queries for which consistent query answering is in **FO**. Under these access restrictions, we showed how to construct strategies that combine answers by means of union and quantification. It is an open question whether our strategies can still be improved, e.g., by using negation.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.
2. M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pages 68–79. ACM Press, 1999.
3. L. E. Bertossi. *Database Repairing and Consistent Query Answering.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.
4. L. E. Bertossi and L. Li. Achieving data privacy through secrecy views and null-based virtual updates. *IEEE Trans. Knowl. Data Eng.*, 25(5):987–1000, 2013.
5. M. Bienvenu and R. Rosati. Tractable approximations of consistent query answering for robust ontology-based data access. In *IJCAI*. IJCAI/AAAI, 2013.
6. N. V. Cao, E. Fragnire, J.-A. Gauthier, M. Sapin, and E. D. Widmer. Optimizing the marriage market: An application of the linear assignment model. *European Journal of Operational Research*, 202(2):547 – 553, 2010.

7. J. Chomicki and J. Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1-2):90–121, 2005.
8. N. N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: diamonds in the dirt. *Commun. ACM*, 52(7):86–94, 2009.
9. N. N. Dalvi, C. Re, and D. Suciu. Queries and materialized views on probabilistic databases. *J. Comput. Syst. Sci.*, 77(3):473–490, 2011.
10. W. Fan and F. Geerts. *Foundations of Data Quality Management.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.
11. A. Fuxman and R. J. Miller. First-order query rewriting for inconsistent databases. In *ICDT*, volume 3363 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 2005.
12. N. Immerman. *Descriptive complexity.* Graduate texts in computer science. Springer, 1999.
13. P. Koutris and J. Wijsen. The data complexity of consistent query answering for self-join-free conjunctive queries under primary key constraints. In *PODS*, pages 17–29. ACM, 2015.
14. P. Koutris and J. Wijsen. A trichotomy in the data complexity of certain query answering for conjunctive queries. *CoRR*, abs/1501.07864, 2015.
15. L. Libkin. *Elements of Finite Model Theory.* Springer, 2004.
16. L. Libkin. SQL's three-valued logic and certain answers. In *ICDT*, volume 31 of *LIPIcs*, pages 94–109. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.
17. D. Maslowski and J. Wijsen. A dichotomy in the complexity of counting database repairs. *J. Comput. Syst. Sci.*, 79(6):958–983, 2013.
18. D. Maslowski and J. Wijsen. Counting database repairs that satisfy conjunctive queries with self-joins. In *ICDT*, pages 155–164. OpenProceedings.org, 2014.
19. J. Wijsen. Making more out of an inconsistent database. In *ADBIS*, volume 3255 of *Lecture Notes in Computer Science*, pages 291–305. Springer, 2004.
20. J. Wijsen. Charting the tractability frontier of certain conjunctive query answering. In *PODS*, pages 189–200. ACM, 2013.
21. J. Wijsen. A survey of the data complexity of consistent query answering under key constraints. In *FoIKS*, volume 8367 of *Lecture Notes in Computer Science*, pages 62–78. Springer, 2014.