First-Order Rewritability in Consistent Query Answering with Respect to Multiple Keys

Paraschos Koutris
paris@cs.wisc.edu
University of Wisconsin-Madison
WI, USA

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

ABSTRACT

We study consistent query answering with respect to key dependencies. Given a (possibly inconsistent) database instance and a set of key dependencies, a repair is an inclusion-maximal subinstance that satisfies all key dependencies. Consistent query answering for a Boolean query is the following problem: given a database instance as input, is the query true in every repair? In [Koutris and Wijsen, ICDT 2019], it was shown that for every self-join-free Boolean conjunctive query and set of key dependencies containing exactly one key dependency per relation name (also called the primary key), this problem is in **FO**, **L**-complete, or **coNP**-complete, and it is decidable which of the three cases applies. In this paper, we consider the more general case where a relation name can be associated with more than one key dependency. It is shown that in this more general setting, it remains decidable whether or not the above problem is in **FO**, for self-join-free Boolean conjunctive queries. Moreover, it is possible to effectively construct a first-order query that solves the problem whenever such a query exists.

CCS CONCEPTS

• **Information systems → Relational database query languages**; • **Theory of computation → Incomplete, inconsistent, and uncertain databases**; **Logic and databases**.

KEYWORDS

conjunctive queries, consistent query answering, database repairing, first-order rewriting, keys

ACM Reference Format:
Paraschos Koutris and Jef Wijsen. 2020. First-Order Rewritability in Consistent Query Answering with Respect to Multiple Keys. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3375395.3387654

1 INTRODUCTION

Key dependencies (or "keys" for short) are the most prevalent integrity constraints in relational databases. In database theory, key dependencies are usually introduced as a special kind of functional dependencies. In practice, however, it is generally sufficient to have keys, because a relational database schema in Boyce-Codd Normal Form (BCNF) uses no functional dependencies other than keys. SQL provides the constructs PRIMARY KEY and UNIQUE for defining keys. In this paper, we deal with a dirty data scenario where keys are known, but violated by the database instance. This situation can occur, for example, in data integration [2]. A common approach to deal with the inconsistency is to repair the instance, i.e., find an inclusion-maximal consistent subinstance. Given that there can be many repairs, we are particularly interested in finding query answers that are consistent, i.e., they are query answers in every repair.

Assume a vocabulary of relation names, where every relation name R is associated with an arity $\mathsf{arity}(R)$ and a set of keys. Every key K of R is a subset of $\{1, \ldots, \mathsf{arity}(R)\}$. We say that a database instance **db** is consistent if for every relation name R and every key K of R, the following holds: if **db** contains two distinct facts $R(\vec{a}_1)$ and $R(\vec{a}_2)$, then $\vec{a}_1$ and $\vec{a}_2$ disagree on some position of K. A repair of a database instance is any inclusion-maximal consistent subinstance. Given a Boolean query q, we are interested in the (data) complexity of the following problem:

INPUT: A database instance **db**.
QUESTION: Is q true in every repair of **db**?

We denote this problem by $\mathsf{CERTAINTY}(q)$, where it is understood that the integrity constraints are the keys associated with the relation names that occur in q. For the results in the current paper, the distinction between Boolean and non-Boolean queries is not theoretically fundamental; this will be explained in Section 8.2.

In [17, 19], the authors study consistent query answering (CQA) for queries in sjfBCQ, the class of self-join-free Boolean conjunctive queries. It is shown that when every relation name is equipped with exactly one key dependency, which is called the *primary key*, every problem in $\{\mathsf{CERTAINTY}(q) \mid q \in \mathsf{sjfBCQ}\}$ is in **FO**, **L**-complete, or **coNP**-complete, and it is decidable (in polynomial time), given $q \in \mathsf{sjfBCQ}$, to which of these complexity classes $\mathsf{CERTAINTY}(q)$ belongs. Moreover, whenever $\mathsf{CERTAINTY}(q)$ is in **L**, it can be expressed in symmetric stratified Datalog with some aggregation operator.

In this paper, we investigate the more general setting where relation names can be equipped with more than one key dependency. We argue next that the two-keys case is already fundamentally different from the primary-key case. In the case of primary keys, two distinct database facts $R(\vec{a}_1)$ and $R(\vec{a}_2)$ conflict with one another if $\vec{a}_1$ and $\vec{a}_2$ have the same value for the primary key of R. The binary relation *"conflicts with"* on the set of database facts is symmetric and transitive. Thus, the reflexive closure of this binary relation is an equivalence relation, whose equivalence classes are commonly

PODS'20, June 14–19, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-7108-7/20/06…$15.00
https://doi.org/10.1145/3375395.3387654

called *blocks*. Every repair is obtained by selecting exactly one fact from each block. It follows that all repairs have the same cardinality, and that every primary-key value occurs (exactly once) in every repair. If a relation name can have two (or more) keys, a significant difference emerges: the relation *"conflicts with"* is no longer always transitive. Consider, for example, the instance shown next where $\{A\}$ and $\{B\}$ are both keys.[1]

R

A	B
c	e
c	f
d	f

.

The fact $R(c, e)$ conflicts with $R(c, f)$, and $R(c, f)$ conflicts with $R(d, f)$, but $R(c, e)$ does not conflict with $R(d, f)$. This instance has two repairs, i.e., consistent subsets that are maximal with respect to set inclusion:

A	B
c	e
d	f

and

A	B
c	f

.

The two repairs have different cardinalities, and the right-hand repair does not contain the A-value d nor the B-value e. An easy yet significant observation is that every repair corresponds to an *inclusion-maximal matching* of the undirected bipartite graph with edges the tuples in R. In fact, we will show that in the presence of two keys, there are queries q such that CERTAINTY(q) is equivalent (under logspace reductions) to the complement of Bipartite Perfect Matching (BPM), which is known to be **NL**-hard [5]. Consequently, in the presence of two (or more) keys, it is possible that CERTAINTY(q) is neither in **L** nor **coNP**-complete (under standard complexity assumptions), which cannot occur if we have only a single key per relation.

The difficulties that emerge for two keys will be captured by the novel notion of *bi-tanglement*. As it will turn out, the addition of a third key incurs some new difficulties, which will be captured by the notion of *tri-tanglement*. For the results in the current paper, it turns out that the addition of a fourth (fifth, sixth,…) key does not create more complexity, and can be fully captured in terms of bi- and tri-tanglements.

In the case of multiple key dependencies per relation name, an ambitious research goal is to exhibit a fine-grained complexity classification of all problems in the set $\{\text{CERTAINTY}(q) \mid q \in \text{sjfBCQ}\}$. This goal is currently unrealistic when one realizes that the exact complexity of BPM is already a notorious open problem (see, for example, [9, 10]). Nevertheless, in this paper we take a first important step by establishing, in the above set, the boundary between problems in **FO** and problems outside **FO**. By definition, CERTAINTY(q) is in **FO** if there exists a first-order query Q such that for every database instance **db**, the following are equivalent:

(1) q is true in every repair of **db**; and
(2) Q is true in **db**.

Such a query Q, if it exists, is called a *consistent first-order rewriting of* q. Since database systems are primarily founded on first-order queries, it is significant to ask which queries have a consistent first-order rewriting. This question is answered in the current paper by means of the following theorem.

Theorem 1.1 (main theorem). *Let q be a self-join-free Boolean conjunctive query, where each relation name is associated with one or more key dependencies. It is decidable (in polynomial time in the size of q and its associated set of key dependencies) whether or not the problem* CERTAINTY(q) *is in* **FO**. *Moreover, if* CERTAINTY(q) *is in* **FO**, *then a consistent first-order rewriting of q can be effectively constructed.*

Guiding Example. This example shows some of the intricacies of our result, as well as our technical contributions. Consider the following database instance for the Eurovision Song Contest.

Host

Year	*Country*
2017	Ukraine
2018	Portugal
2019	Israel

Results

Year	*Country*	*Place*
2018	Portugal	26th
2019	Netherlands	1st
2019	Italy	2nd
2019	Israel	23rd

The only key of *Host* is $\{Year\}$: the contest is organized in one country every year. The keys of *Results* are $\{Year, Country\}$ and $\{Year, Place\}$. The former key expresses that a country cannot occupy different places in a same year; the latter key expresses that there are no ties.

Consider now the following two queries:

$$q_0 = \exists y \exists u \left(Host(y, u) \wedge Results(y, u, \text{“1st”})\right); \quad (1)$$

$$q_1 = \exists y \exists u \exists p \left(Host(y, u) \wedge Results(y, u, p)\right). \quad (2)$$

The first query asks: "Did it happen that the hosting country was also the winner?" The second query asks: "Did it happen that the hosting country was also ranked?" (only countries reaching the "grand final" are ranked).

We will show in this paper that the problem CERTAINTY(q_0) is equivalent under first-order reductions to CERTAINTY(q_0'), where

$$q_0' = \exists y \exists u \exists p \left(Host(y, u) \wedge Results(y, u, p) \wedge P^{\mathsf{c}}(p)\right), \quad (3)$$

a query that is constant-free. Here, P^{c} is an always consistent relation (as the superscript c indicates) whose primary key is the empty set; hence every instance of P^{c} can contain at most one tuple. As we will see later, this elimination of constants turns out to be a very helpful simplification in the technical treatment.

In order to decide whether a problem CERTAINTY(q) is first-order rewritable, we show that the tool of *attack graph* [17, 22] can be extended to deal with the case of multiple keys. For our example, the attack graphs of both queries q_0' and q_1 will turn out to be acyclic. However, in contrast to the case of primary keys, acyclicity of a query's attack graph is only a necessary (but not a sufficient) condition for the existence of a consistent first-order rewriting. Informally, this is because attack graphs capture well the interaction between keys belonging to different relation names, but fall short in capturing the interplay between different keys of a same relation name. We coin the term "tanglement" to refer to this interplay between keys.

For our example, q_1 has a bi-tanglement (tanglement between two keys; details will be given in Examples 5.2–6.3). Even if the attack graph of q_1 is acyclic, our results imply that CERTAINTY(q_1) is **NL**-hard, and hence not in **FO**. In contrast, q_0' has no bi-tanglement,

[1] As is common in database theory, we assume in this example that attribute names can be used to denote positions.

and CERTAINTY(q_0) is in **FO**. In particular, a consistent first-order rewriting of q_0 is:

$$\exists y\left(\begin{array}{l}\exists u Host(y,u)\\ \wedge\ \forall u\left(Host(y,u)\rightarrow\left(\begin{array}{l}Results(y,u,\text{“1st”})\wedge\\ \neg\exists u'\exists p\left(\begin{array}{l}Results(y,u',p)\wedge\\ \left(\begin{array}{c}(u'=u\wedge p\neq\text{“1st”})\\ \vee\\ (p=\text{“1st”}\wedge u'\neq u)\end{array}\right)\end{array}\right)\end{array}\right)\right)\end{array}\right). \tag{4}$$

The consistent first-order rewritings in our study can be easily encoded in SQL or Datalog. Nonetheless, it remains an open question whether and when such queries can be executed more efficiently than generic SAT-based implementations of CERTAINTY(q) [8].

Organization. The next section discusses related work. After that, the paper step by step introduces new constructs and intermediate results that are needed in the proof of the main Theorem 1.1. Section 3 defines the basic concepts and terminology. Section 4 shows that for every self-join-free Boolean conjunctive query q, the problem CERTAINTY(q) is equivalent under first-order reductions to a problem CERTAINTY(q') such that q' has no atom containing constants or repeated variables. Section 5 introduces the construct of attacks among keys, and extends attack graphs to deal with multiple keys. Section 6 introduces the notion of bi-tanglement (tanglement of two keys), and Section 7 the notion of tri-tanglement (tanglement of three keys), At that point, we have all ingredients to tackle the proof of the main Theorem 1.1 in Section 8. While the main focus of this paper is on consistent first-order rewriting, Section 9 gives a glimpse of what happens beyond first-order rewritability. Section 10 discusses open problems for CQA with respect to keys, and shows some relationships with notorious complexity problems in theoretical computer science.

2 RELATED WORK

Consistent query answering (CQA) started with an article at PODS 1999 by Arenas, Bertossi and Chomicki [1]. Twenty years later, the significance of their contribution was acknowledged through a *Gems of PODS* session [3]. An overview of complexity results in CQA appeared recently in the *Database Principles* column of SIGMOD Record [24].

In CQA, the existence of consistent first-order rewritings, for different classes of queries and integrity constraints, is a recurrent research problem. For self-join-free conjunctive queries and primary keys, the problem has been studied in depth since 2005 [12, 13]. In this setting, the problem was first solved in [22, 23], albeit under the restriction that queries are α-acyclic; a solution without this restriction appeared later in [17]. The latter solution was extended in [18] to self-join-free conjunctive queries with clique-guarded negated atoms. Negation is clique-guarded if whenever two variables occur together in some negated atom, they also occur together in some non-negated atom. To the best of our knowledge, for primary keys, it remains an open problem to determine the **FO**-boundary in the set $\{\text{CERTAINTY}(q) \mid q \in \text{BCQ}\}$, where BCQ is the class of Boolean conjunctive queries, possibly with self-joins.

The importance of first-order rewritings is also recognized in the Description Logic (DL) community, where it was a major motivation for the *DL-Lite* family of description logics [4]. The description logic *DLR-Lite*$_{\mathcal{A},\sqcap}$ even allows for the use of n-ary relations, rather than binary roles, and key assertions. Given a *DL-Lite* TBox and a conjunctive query q in the signature of the TBox, the *certain answers* to q on an ABox can be obtained by issuing a first-order rewriting of q against the ABox. Certain answers in DL are similar to consistent answers: whenever there is more than one model, one is interested in the intersection of query answers, where the intersection is taken over all models. In [4], the study of first-order rewriting of (unions of) conjunctive queries with respect to *DLR-Lite*$_{\mathcal{A},\sqcap}$ TBoxes assumes that key assertions are not violated, and therefore is fundamentally different from our study.

The goal of the current paper is the effective classification of problems CERTAINTY(q) as either in **FO** or outside **FO**. Another important problem is that of classifying problems CERTAINTY(q) as either in **PTIME** or **coNP**-complete. A long line of work [12, 13, 15–17] has shown that for self-join-free conjunctive queries and primary keys, CERTAINTY(q) is either in **PTIME** or **coNP**-complete. In particular, this class of problems exhibits a stronger dichotomy, because CERTAINTY(q) is either in **L** or **coNP**-complete [19]. Unfortunately, not much is known beyond this case.

Beyond CQA, related to this paper is recent work [21] that studied the problem of computing a cardinality-maximal subset repair with respect to functional dependencies. For the binary relation R in Section 1 with functional dependencies $A \rightarrow B$ and $B \rightarrow A$, every cardinality-maximal subset repair is a matching of maximal cardinality (and vice versa). The proof of Lemma 6.4 reduces the problem of finding such a matching to the problem of finding a falsifying repair for a query with a bi-tanglement.

3 PRELIMINARIES

This section introduces basic concepts. For the sake of rigor, we repeat some constructs that were already introduced, somewhat informally, in Section 1.

Key Dependencies. We assume a schema with a fixed vocabulary of relation names. Each relation name R has an associated arity arity(R) and a non-empty set of key dependencies. A *key dependency (KD)* is an expression $[\![R : K]\!]$ where K is a (possibly empty) subset of $\{1, \ldots, \text{arity}(R)\}$.

An *atom* is an expression $R(s_1, \ldots, s_n)$ where $n = \text{arity}(R)$, and each s_i is a constant or a variable. If A is an atom, then vars(A) is the set of variables that occur in A. If $\text{vars}(A) = \emptyset$, then A is a *fact*. A *database instance* is a finite set of facts. A database instance **db** satisfies $[\![R : K]\!]$ if it contains no two distinct R-facts that agree on all positions in K. If R has only one key, then this key, called the *primary key*, is usually underlined.

We will assume that for any two distinct KDs $[\![R : K]\!]$ and $[\![R : L]\!]$ with the same relation name R, we have that K and L are not comparable by set inclusion. The hypothesis that every relation name is associated with at least one KD is without loss of generality. Informally, this is because every set of R-facts is duplicate-free and hence satisfies the trivial KD $[\![R : \{1, \ldots, \text{arity}(R)\}]\!]$. If $[\![R : K]\!]$ is associated with R, then we also say that $[\![R : K]\!]$ is a *KD of* R.

A database instance **db** is *consistent* if it satisfies all KDs associated with the relation names that occur in **db**. A *repair* of **db** is an inclusion-maximal consistent subset of **db**. We write rset(**db**) for

the set of repairs of **db**, and adom(**db**) for the active domain of **db** (i.e., the set of all constants that occur in **db**).

The *conflict graph* of **db** is an undirected graph whose vertex-set is **db**; two distinct vertices are adjacent if they violate some KD.

In the case that every relation name is associated with a single key, then maximality with respect to set inclusion coincides with maximality with respect to cardinality. This is no longer true if two or more keys can be associated with a same relation name, as illustrated in Section 1.

Consistent Query Answering. A *Boolean query* maps every database instance to true or false. For a Boolean query q, CERTAINTY(q) is the following problem:

INPUT: A database instance **db**.
QUESTION: Is q true in every repair of **db**?

Our results will be proved for self-join-free Boolean conjunctive queries. A Boolean conjunctive query q is a closed first-order sentence of the form $\exists\vec{x}\,(R_1(\vec{x}_1) \wedge \cdots \wedge R_m(\vec{x}_m))$. In the technical treatment, such a query is identified with its set of atoms, i.e., $q = \{R_i(\vec{x}_i)\}_{i=1}^{m}$. A conjunctive query is *self-join-free* if no relation name occurs more than once in it, i.e., if $i \neq j$ implies $R_i \neq R_j$. We denote by sjfBCQ the set of self-join-free Boolean conjunctive queries. We write vars(q) for the set of variables occurring in q.

Once a self-join-free query $q = \{R_i(\vec{x}_i)\}_{i=1}^{m}$ is fixed in some context, there is no ambiguity when we use a relation name at places where an atom is expected: when we refer to the atom R_i, we mean the atom of q with relation name R_i, that is, $R_i(\vec{x}_i)$.

To simplify our technical treatment, we associate with each relation name a *mode*, which is a value in {i, c}. Informally, i and c stand respectively for *(possibly) inconsistent* and *(necessarily) consistent*. The notation R^{c} is used to indicate that R is a relation name of mode c. For a Boolean query q, CERTAINTY(q) is then the following problem:

INPUT: A database instance **db** such that for every relation name R of mode c, the set of R-facts in **db** is consistent.
QUESTION: Is q true in every repair of **db**?

Lemma 3.1 shows that every problem CERTAINTY(q) is equivalent under first-order reductions to a problem CERTAINTY(q') such that q' contains no atoms of mode c. Therefore, in the complexity classification of CERTAINTY(q), atoms of mode c can be seen as (very convenient) syntactic sugar.

Lemma 3.1 (mode c elimination). *For every query $q \in$ sjfBCQ, it is possible to compute in linear time (in the size of q) a query $q' \in$ sjfBCQ such that*

(1) q' contains no atom of mode c*; and*
(2) CERTAINTY(q) *and* CERTAINTY(q') *are equivalent under first-order reductions.*

In the next section, we show that in the complexity classification of {CERTAINTY(q) | $q \in$ sjfBCQ}, it can be assumed without loss of generality that no atom contains constants or repeated variables.

4 ELIMINATION OF ATOMS WITH CONSTANTS OR REPEATED VARIABLES

In this section, we show that for every $q \in$ sjfBCQ, there exists some constant-free query $q' \in$ sjfBCQ such that CERTAINTY(q) and CERTAINTY(q') are equivalent under first-order reductions, and no atom of q' contains repeated variables. Moreover, q' can be effectively computed, given q. The elimination of atoms with constants or repeated variables allows for a significant syntactic simplification, as explained in Section 4.2: it allows us to use variables as attribute names.

4.1 Elimination Lemmas

In the following lemma and elsewhere, it is understood that every relation name is equipped with a set of KDs (instead of just one primary key).

The following lemma explains how to eliminate repeated variables from an atom of a query. The lemma states, for example, that an atom $R(x, x, \vec{y})$ can be replaced with two atoms $R(x, v, \vec{y})$, $S^{\mathsf{c}}(x, v)$, where S^{c} has two keys $\{1\}$ and $\{2\}$. Informally, this means that an instance of S^{c} is a one-one mapping which can be used to capture equality between positions 1 and 2 in R.

Lemma 4.1 (Elimination of repeated variables). *Let $q \in$ sjfBCQ, and let $F := R(t_1, \ldots, t_n)$ be an atom of q in which the variable x occurs more than once. Let $i, j \in \{1, \ldots, n\}$ with $i < j$ be integers such that $t_i = t_j = x$. Let v be a fresh variable, and let $F' := R(t'_1, \ldots, t'_n)$ be an R-atom such that $t'_j = v$ and $t'_\ell = t_\ell$ for $\ell \neq j$. Let $p = \{F', S^{\mathsf{c}}(x, v)\}$, where S is a fresh relation name of mode* c *whose set of KDs is the pair $\{[\![S^{\mathsf{c}} : \{1\}]\!], [\![S^{\mathsf{c}} : \{2\}]\!]\}$. Then* CERTAINTY($q$) *and* CERTAINTY($(q \setminus \{F\}) \cup p$) *are equivalent under first-order reductions.*

The following lemma explains how to eliminate constants from an atom of a query. The lemma states that an atom $R(a, \vec{y})$ can be replaced with two atoms $R(v, \vec{y})$, $C^{\mathsf{c}}(v)$, where the empty set is a key of C^{c}. Informally, this means that an instance of C^{c} can contain at most one value, which can be used to capture the occurrence of a constant at position 1 in R.

Lemma 4.2 (Elimination of constants). *Let $q \in$ sjfBCQ, and let $F := R(t_1, \ldots, t_n)$ be an atom of q in which a constant occurs. Let $j \in \{1, \ldots, n\}$ such that t_j is a constant. Let v be a fresh variable, and let $F' := R(t'_1, \ldots, t'_n)$ be an R-atom such that $t'_j = v$ and $t'_\ell = t_\ell$ for $\ell \neq j$. Let $p = \{F', C^{\mathsf{c}}(v)\}$, where C is a fresh relation name of mode* c *whose only KD is $[\![C^{\mathsf{c}} : \emptyset]\!]$. Then* CERTAINTY($q$) *and* CERTAINTY($(q \setminus \{F\}) \cup p$) *are equivalent under first-order reductions.*

Corollary 4.3. *For every query $q \in$ sjfBCQ, there exists a query $q' \in$ sjfBCQ such that*

- *no constant occurs in q';*
- *q' contains no atom in which some variable occurs more than once; and*
- CERTAINTY(q) *and* CERTAINTY(q') *are equivalent under first-order reductions.*

Moreover, the query q' can be computed in linear time in the size of q.

4.2 Syntactic Simplification

Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. Assume that q contains an atom $R(x_1, \ldots, x_n)$. We have that for all $i, j \in \{1, \ldots, n\}$, $i \neq j$ implies $x_i \neq x_j$ (the converse holds vacuously). This allows for a far-reaching syntactic

simplification: for every $i \in \{1, \ldots, n\}$, we can use the variable x_i at all places where the position i is expected. For instance, if we refer to the constant occurring *at the position of* x_i in some R-fact, we mean the constant occurring at position i. Formally, for an R-fact $F = R(a_1, \ldots, a_n)$, we define the expression $F@x_i$ to denote the constant a_i (where the R-atom of the query q is implicitly understood).

For example, if q contains $R(x, y, z)$, then we can say that the constant at the position of y in $R(a, b, c)$ is b. Formally, we write $R(a, b, c)@y = b$.

The blurring of variables and positions naturally extends to other expressions: if we say that the R-facts A and B agree on some subset X of $\{x_1, \ldots, x_n\}$, then we mean that for every $x_i \in X$, we have $A@x_i = B@x_i$.

For example, if q contains $R(x, y, z)$, then it is correct to say that $R(a, b, c)$ and $R(a, b, e)$ agree on $\{x, y\}$.

Furthermore, if R is associated with the KD $[\![R : K]\!]$, then we can also refer to this KD as $[\![R : K']\!]$ with $K' = \{x_i \mid i \in K\}$. From here on, to make the difference explicit, we will call $[\![R : K']\!]$ *a key of* q, or alternatively, *a key of* R. That is, we will use the term *key* whenever variables are used instead of positions. As a shorthand, we may also say that K' is a key of R, with the meaning that $[\![R : K']\!]$ is a key of R.

For example, if q contains $R(x, y, z)$ and $[\![R : \{1, 2\}]\!]$ is a KD of R, then it is correct to say that $[\![R : \{x, y\}]\!]$ is a key of R. For short, we can also say that $\{x, y\}$ is a key of R.

5 ATTACKS AMONG KEYS

We extend the notion of *attack graph*, initially introduced in [23] for primary keys, to the case with multiple keys per relation name. We then show that if the attack graph of a query q contains a cycle, then CERTAINTY(q) is **L**-hard, and therefore not in **FO**.

Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. A variable of vars(q) is called a *join variable* if it occurs more than once in q. Since no atom contains repeated variables, a join variable must necessarily occur in at least two distinct atoms.

A *functional dependency on* q is an expression $X \to Y$ with $X \cup Y \subseteq \mathrm{vars}(q)$. We define $\mathcal{K}(q)$ as the set of functional dependencies on q containing $K \to \mathrm{vars}(R)$ whenever $[\![R : K]\!]$ is a key of q. For every atom R of q, we define:

$$\mathcal{K}(q \ominus R) := \begin{cases} \mathcal{K}(q \setminus \{R\}) & \text{if } R \text{ has mode i} \\ \mathcal{K}(q) & \text{if } R \text{ has mode c} \end{cases}$$

Example 5.1. Let $q = \{R^{\mathsf{c}}(\underline{x}, y), S(\underline{y}, x)\}$ with keys $[\![R : \{x\}]\!]$ and $[\![S : \{y\}]\!]$. Then, $\mathcal{K}(q \ominus R) \equiv \{x \to y,\ y \to x\}$ and $\mathcal{K}(q \ominus S) \equiv \{x \to y\}$.

For every set $V \subseteq \mathrm{vars}(R)$, we define $[R : V]^{\ominus, q}$ as the following set of variables:

$$[R : V]^{\ominus, q} := \{x \in \mathrm{vars}(q) \mid \mathcal{K}(q \ominus R) \models V \to x\}.$$

Let $[\![R : K]\!]$ be a key of q, and v a variable of vars(q). We say that $[\![R : K]\!]$ *attacks* v, denoted $[\![R : K]\!] \overset{q}{\rightsquigarrow} v$, if for some $\ell \geq 0$, there exists a sequence $v_0, v_1, \ldots, v_\ell$ of variables, all belonging to $\mathrm{vars}(q) \setminus [R : K]^{\ominus, q}$, such that $v_0 \in \mathrm{vars}(R)$, $v_\ell = v$, and every two adjacent variables occur together in some atom of q.

We write $[\![R : K]\!] \overset{q}{\rightsquigarrow} [\![S : L]\!]$ if $[\![R : K]\!]$ attacks some variable of L, where it is understood that $[\![S : L]\!]$ is a key of q. If $[\![R : K]\!] \overset{q}{\rightsquigarrow} [\![S : L]\!]$, we also say that $[\![R : K]\!]$ attacks $[\![S : L]\!]$.

Informally, an attack $[\![R : K]\!] \overset{q}{\rightsquigarrow} v$ means that there exists a "yes"-instance **db** of CERTAINTY(q) containing distinct facts $R(\vec{a}_1)$ and $R(\vec{a}_2)$ that agree on K, but join, through the variable sequence $v_0, v_1, \ldots, v_\ell = v$, with different values of v.

Example 5.2. For q_1 in Equation (2), the keys are $[\![\mathit{Host} : \{y\}]\!]$, $[\![\mathit{Results} : \{y, u\}]\!]$, and $[\![\mathit{Results} : \{y, p\}]\!]$. We have

$$\begin{aligned} \mathcal{K}(q_1 \ominus \mathit{Host}) &\equiv \{\{y, u\} \to \{p\}, \{y, p\} \to \{u\}\}; \\ \mathcal{K}(q_1 \ominus \mathit{Results}) &\equiv \{\{y\} \to \{u\}\}. \end{aligned}$$

Since u belongs to vars(Host) but not to $[\mathit{Host} : \{y\}]^{\ominus, q_1} = \{y\}$, the singleton sequence u shows that $[\![\mathit{Host} : \{y\}]\!] \overset{q_1}{\rightsquigarrow} u$. The sequence u, p shows that $[\![\mathit{Host} : \{y\}]\!] \overset{q_1}{\rightsquigarrow} p$. Consequently, the key of *Host* attacks each key of *Results*.

Since p is in vars($\mathit{Results}$) but not in $[\mathit{Results} : \{y, u\}]^{\ominus, q_1} = \{y, u\}$, the singleton sequence p shows that $[\![\mathit{Results} : \{y, u\}]\!] \overset{q_1}{\rightsquigarrow} p$. From $[\mathit{Results} : \{y, p\}]^{\ominus, q_1} = \mathrm{vars}(\mathit{Results})$, it follows that no variable is attacked by the key $\{y, p\}$ of *Results*.

Example 5.3. For q_0' in Equation (3), the keys are $[\![\mathit{Host} : \{y\}]\!]$, $[\![\mathit{Results} : \{y, u\}]\!]$, $[\![\mathit{Results} : \{y, p\}]\!]$, and $[\![P^{\mathsf{c}} : \emptyset]\!]$. We have

$$\begin{aligned} \mathcal{K}(q_0' \ominus \mathit{Host}) &\equiv \{\{y, u\} \to \{p\}, \{y, p\} \to \{u\}, \emptyset \to \{p\}\}; \\ \mathcal{K}(q_0' \ominus \mathit{Results}) &\equiv \{\{y\} \to \{u\}, \emptyset \to \{p\}\}. \end{aligned}$$

Since $[\mathit{Host} : \{y\}]^{\ominus, q_0'} = \mathrm{vars}(\mathit{Host})$, it follows that the key of *Host* attacks no variable. Since $[\mathit{Results} : \{y, u\}]^{\ominus, q_0'} = [\mathit{Results} : \{y, p\}]^{\ominus, q_0'} = \mathrm{vars}(\mathit{Results})$, no variable is attacked by one of the keys of *Results*.

Lemma 5.4. *Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. Let $[\![R : K]\!]$ be a key of q. Let S be an atom of q such that $S \neq R$. If $[\![R : K]\!]$ attacks some variable of* vars(S)*, then $[\![R : K]\!]$ attacks every key of S.*

Proof. Assume $[\![R : K]\!] \overset{q}{\rightsquigarrow} v$ with $v \in \mathrm{vars}(S)$. Let $[\![S : L]\!]$ be a key of S.

The desired result trivially holds if $v \in L$. Assume $v \notin L$ from here on. We can assume a sequence

$$v_0, v_1, \ldots, v_\ell$$

such that $v_0 \in \mathrm{vars}(R)$, $v_\ell = v$, no v_i belongs to $[R : K]^{\ominus, q}$, and every two adjacent variables occur together in some atom of q. It must be the case that $L \not\subseteq [R : K]^{\ominus, q}$, or else $v_\ell \in \mathrm{vars}(S) \subseteq [R : K]^{\ominus, q}$, a contradiction. Therefore, there is a variable $x \in L$ such that $x \notin [R : K]^{\ominus, q}$. The sequence

$$v_0, v_1, \ldots, v_\ell, x$$

shows that $[\![R : K]\!] \overset{q}{\rightsquigarrow} x$, and therefore $[\![R : K]\!] \overset{q}{\rightsquigarrow} [\![S : L]\!]$. □

We will prove that CERTAINTY(q) is **L**-hard if two keys of different atoms attack one another. Now if two keys of different atoms attack one another, then, by Lemma 5.4, every key of either atom attacks every key of the other atom. Therefore, to detect mutual attacks, we can use a granularity of atoms rather than individual keys, which motivates the following definition.

The *attack graph* of a query q is a directed graph whose vertices are the atoms of q. There is a directed edge from an atom R to an atom S if $R \neq S$ and some key of R attacks some (and therefore every) key of S.

From Example 5.2, it follows that the attack graph of the query q_1 in Equation (2) contains a single edge from *Host* to *Results*. Appendix D contains a proof of the following result.

Lemma 5.5. *Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. If the attack graph of q has a cycle, then* CERTAINTY(q) *is* **L***-hard.*

6 BI-TANGLEMENT

Before giving the definition of bi-tanglement, we recall that Bipartite Perfect Matching (BPM) is the following problem:

INPUT: An undirected bipartite graph G with the same number of vertices in each partition.

QUESTION: Does G have a perfect matching?

Since the complexity of BPM is still unsettled as of today, we introduce the following terminology. A decision problem P is said to be **coBPM**-hard under first-order reductions if there exists a first-order reduction from the complement of BPM to P. Since BPM is **NL**-hard [5], and since **NL** is closed under complement, every **coBPM**-hard problem is **NL**-hard.

We now define what it means that a query q has a *bi-tanglement*; in the next section we will introduce the related concept of *tri-tanglement*. These notions are used to show some complexity lower bounds: we will prove shortly that if q has a bi-tanglement, then CERTAINTY(q) is **coBPM**-hard. In Section 7, we will show that if q has a tri-tanglement, then CERTAINTY(q) is **L**-hard.

Definition 6.1. (**bi-tanglement**) Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. We say that q *has a bi-tanglement* if q has an atom R of mode i with two distinct keys $[\![R : K]\!]$, $[\![R : L]\!]$ such that $\mathrm{vars}(R) \setminus K$ contains a join variable and $K \not\subseteq [R : K \cap L]^{\ominus,q}$.

Example 6.2. The query q_1 of Equation (2) has a bi-tanglement between the keys $[\![\mathit{Results} : \{y, u\}]\!]$ and $[\![\mathit{Results} : \{y, p\}]\!]$. The intersection of these keys is $\{y\}$. The variable u is a join variable. Since the key $\{y, p\}$ does not contain the variable u and is not included in $[\mathit{Results} : \{y\}]^{\ominus,q_1} = \{y, u\}$, it is correct to conclude that q_1 has a bi-tanglement.

Example 6.3. The query q_0' of Equation (3) has no bi-tanglement between the keys $[\![\mathit{Results} : \{y, u\}]\!]$ and $[\![\mathit{Results} : \{y, p\}]\!]$, because $[\mathit{Results} : \{y\}]^{\ominus,q_0'} = \mathrm{vars}(\mathit{Results})$ includes both keys.

Lemma 6.4 (bi-tanglement). *Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. If q has a bi-tanglement, then* CERTAINTY(q) *is* **coBPM***-hard (under first-order reductions).*

Proof. Assume that q has a bi-tanglement. Therefore, we can assume that q contains an R-atom of mode i with two distinct keys $[\![R : K]\!]$, $[\![R : L]\!]$ such that the following two conditions are satisfied:

(A) some variable of $\mathrm{vars}(R) \setminus K$ is a join variable (i.e., occurs also in $q \setminus \{R\}$); and

(B) $\mathcal{K}(q \ominus R) \not\models K \cap L \to K$ (or equivalently, $K \not\subseteq [R : K \cap L]^{\ominus,q}$).

The proof is a first-order reduction from BPM to the complement of CERTAINTY(q). Let (P_1, P_2, E) with $|P_1| = |P_2|$ and $E \subseteq P_1 \times P_2$ be an instance of BPM. We construct a database instance **db** as follows. Let $V = [R : K \cap L]^{\ominus,q}$. Thus $K \not\subseteq V$.

For every bipartite edge (a, b) in E, **db** contains an R-fact $\Gamma_{a,b}$ such that for every $x \in \mathrm{vars}(R)$,

$$\Gamma_{a,b}@x = \begin{cases} a_x & \text{if } x \in K \setminus V \\ \bot_x & \text{if } x \in K \cap V \\ b_x & \text{if } x \in L \setminus K \\ \langle a, b\rangle_x & \text{otherwise} \end{cases}$$

The subscripting with x is not essential, but has been added for clarity to indicate that values can be typed according to their position.

Moreover, for every vertex a in P_1, **db** contains $\theta_a(q)$ where θ_a is the valuation over $\mathrm{vars}(q)$ such that for every $x \in \mathrm{vars}(q)$,

$$\theta_a(x) = \begin{cases} \bot_x & \text{if } x \in V \\ a_x & \text{otherwise} \end{cases}$$

Let $\mathbf{db} := \mathbf{db}_0 \cup \mathbf{db}_1$ where

$$\begin{aligned} \mathbf{db}_0 &:= \{\Gamma_{a,b} \mid (a, b) \in E\}; \\ \mathbf{db}_1 &:= \{\theta_a(q) \mid a \in P_1\}. \end{aligned}$$

For edges (a, b), (a', b') in E, consider the set $C := \{\Gamma_{a,b}, \Gamma_{a',b'}\}$. Note that $K \setminus V \neq \emptyset$, and since $L \not\subseteq K$, also $L \setminus K \neq \emptyset$. These two observations imply that $\Gamma_{a,b} = \Gamma_{a',b'}$ if and only if $a = a'$ and $b = b'$. Also, the following hold:

(a) If $a \neq a'$ and $b \neq b'$, then C is consistent. Indeed, assume for the sake of contradiction that the two facts of C agree on some key $[\![R : N]\!]$. By our construction, it must be that $N \subseteq K \cap V$. Since no two distinct keys are comparable by set inclusion, it follows $N = K$, hence $K \subseteq K \cap V$, contradicting $K \not\subseteq V$.

(b) If $a = a'$ and $b \neq b'$, then C is inconsistent. Indeed, in this case, the two facts of C agree on K (by construction), but are not equal. To see why the two facts of C are not equal, note that there exists some variable $x \in \mathrm{vars}(R) \setminus K$ such that $\Gamma_{a,b}@x \in \{b_x, \langle a, b\rangle_x\}$ and $\Gamma_{a',b'}@x \in \{b'_x, \langle a, b'\rangle_x\}$. From $b \neq b'$, it follows that the two facts of C disagree at the position of x.

(c) If $a \neq a'$ and $b = b'$, then C is inconsistent. Indeed, in this case, the two facts of C agree on L (by construction, using that $K \cap L \subseteq K \cap V$), but are not equal. To see why the two facts of C are not equal, note that there exists some variable $x \in K \setminus V$ such that $\Gamma_{a,b}@x = a_x$ and $\Gamma_{a',b'}@x = a'_x$. From $a \neq a'$, it follows that the two facts of C disagree at the position of x.

From the above, it is correct to conclude that any repair of $\mathbf{db}_0$ corresponds to a maximal matching of the bipartite graph.

Consider next the database instance $\mathbf{db}_1$. We first show that for every relation name S distinct from R, the set of S-facts in $\mathbf{db}_1$ is consistent. For this purpose, take a key $[\![S : M]\!]$ and assume that θ_a and θ_b with $a \neq b$ agree on M. Then, $\mathcal{K}(q \ominus R) \models K \cap L \to M$, and therefore, since $\mathcal{K}(q \ominus R)$ contains $M \to \mathrm{vars}(S)$, we have $\mathcal{K}(q \ominus R) \models K \cap L \to \mathrm{vars}(S)$. By our construction, θ_a and θ_b agree on $\mathrm{vars}(S)$.

For edge (a', b) in E and vertex $a \in P_1$, consider the set $D := \{\theta_a(R), \Gamma_{a',b}\}$. The following hold:

(a) If $a = a'$, then D is inconsistent. Indeed, by our construction the two facts of D agree on K, but they will disagree on every variable outside of K.

(b) If $a \neq a'$, then D is consistent. Indeed, assume $N \subseteq \mathrm{vars}(R)$ such that $[\![R : N]\!]$ is a key of R that is violated by D. By our construction, it must be the case that $N \subseteq K \cap V$. Since no two distinct keys are comparable by set inclusion, it follows $N = K$, and consequently, $K \subseteq V$, contradicting condition (B).

We now consider the two possibilities for (P_1, P_2, E).

Case that (P_1, P_2, E) is a "yes"-instance of BPM. Let $\mathbf{r}_0$ be a repair of $\mathbf{db}_0$ that encodes a perfect matching. Then, we can extend $\mathbf{r}_0$ to a repair of $\mathbf{db}$ that contains no R-fact of $\mathbf{db}_1$, since any extension with an R-fact from $\mathbf{db}_1$ results in inconsistency. By condition (A), such a repair will falsify q. Indeed, by condition (A), we can assume a variable $z \in \mathrm{vars}(R) \setminus K$ that also occurs in $q \setminus \{R\}$. All z-values in $\mathbf{db}_0$ belong to $\{\langle a, b\rangle_z \mid (a, b) \in E\} \cup \{b_z \mid b \in P_2\}$, and none of these values occurs in $\mathbf{db}_1$.

Case that (P_1, P_2, E) is a "no"-instance of BPM. Then, for every repair $\mathbf{r}_0$ of $\mathbf{db}_0$, we can assume the existence of a vertex $a \in P_1$ such that for every $x \in K \setminus V$, we have $a_x \notin \mathrm{adom}(\mathbf{r}_0)$. Informally, a is an unmatched P_1-vertex in the bipartite matching encoded by $\mathbf{r}_0$. From our above observations, $\mathbf{r}_0 \cup \theta_a(q)$ is consistent, and hence every repair $\mathbf{r}$ of $\mathbf{db}$ contains one or more R-facts of $\mathbf{db}_1$. It is now obvious that every repair $\mathbf{r}$ of $\mathbf{db}$ satisfies q. □

Lemma 6.5. *Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. If q has no bi-tanglement, then for every atom R of q, one of the following holds true:*

- *there is no key of R that attacks another key of R; or*
- *every join variable of $\mathrm{vars}(R)$ belongs to the intersection of all keys of R.*

Proof. Assume that q has no bi-tanglement. Assume that there is a join variable in $\mathrm{vars}(R)$ that does not belong to the intersection of all keys of R. We need to show that there is no key of R that attacks another key of R. We can assume a join variable $x \in \mathrm{vars}(R)$ and a key $[\![R : K]\!]$ of R such that $x \notin K$.

Let $[\![R : L]\!]$, $[\![R : M]\!]$ be two distinct keys of R. To conclude the proof, it suffices to show $\mathcal{K}(q \ominus R) \models M \to L$, because this implies $[\![R : M]\!] \overset{q}{\rightsquigarrow} [\![K : L]\!]$. We distinguish two cases:

Case that $K = L$. Since q has no bi-tanglement, it follows that $\mathcal{K}(q \ominus R) \models K \cap M \to K$, which implies $\mathcal{K}(q \ominus R) \models M \to L$.

Case that $K \neq L$. Since q has no bi-tanglement, it follows that $\mathcal{K}(q \ominus R) \models K \cap L \to K$. But then, there must exist a join variable $z \in K \setminus L$. Since q has no bi-tanglement, this implies $\mathcal{K}(q \ominus R) \models L \cap M \to L$, hence $\mathcal{K}(q \ominus R) \models M \to L$. □

7 TRI-TANGLEMENT

In this section, we introduce the notion of tri-tanglement, and show that CERTAINTY(q) is **L**-hard if q has a tri-tanglement.

Definition 7.1. (**tri-tanglement**) Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. We say that q *has a tri-tanglement* if q has an atom R of mode i and three distinct keys $[\![R : K]\!]$, $[\![R : L]\!]$, $[\![R : M]\!]$ such that:

(a) for every $N \subseteq \mathrm{vars}(R)$ such that $[\![R : N]\!]$ is a key, we have $K \subseteq [R : K \cap N]^{\ominus, q}$ and $M \subseteq [R : M \cap N]^{\ominus, q}$; and

(b) $[R : K \cap L \cap M]^{\ominus, q}$ includes no set among $K \cap L$, $K \cap M$, $L \cap M$.

Note that the definition is symmetric in K and M (but not in L).

Example 7.2. Let $q_3 = \{R(x, y, z), S^{\mathsf{c}}(x, y, z)\}$ where the keys are $[\![R : \{x, y\}]\!]$, $[\![R : \{y, z\}]\!]$, $[\![R : \{x, z\}]\!]$, $[\![S^{\mathsf{c}} : \{x\}]\!]$, $[\![S^{\mathsf{c}} : \{y\}]\!]$, and $[\![S^{\mathsf{c}} : \{z\}]\!]$. Since S has mode c, it cannot be the cause of a bi- or tri-tanglement. Informally, S^{c} encodes a set of vertex-disjoint tripartite triangles. Every repair of R encodes a set of edge-disjoint tripartite triangles: if a repair contains $R(a, b, c)$, encoding triangle abc, then it cannot contain another triangle with an edge among ab, bc, ac. The query q_3 has no bi-tanglement, because every two distinct keys of R have a nonempty intersection, and $[R : \{x\}]^{\ominus, q_3} = [R : \{y\}]^{\ominus, q_3} = [R : \{z\}]^{\ominus, q_3} = \{x, y, z\}$ includes every key of R.

On the other hand, q_3 has a tri-tanglement. Indeed, satisfaction of condition (a) in Definition 7.1 follows from the previous paragraph; for condition (b), note that the three keys of R have an empty intersection, and $[R : \emptyset]^{\ominus, q_3} = \emptyset$ does not include the intersection of any two keys of R.

Definition 7.3. Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. Let R be an atom of q, and $N \subseteq \mathrm{vars}(R)$. Let $x, y \in \mathrm{vars}(R)$. We write $R : N \mid x \overset{q}{\leftrightsquigarrow} y$ if there exists a sequence of variables $x_0, x_1, \ldots, x_n \notin [R : N]^{\ominus, q}$ such that $x_0 = x$, $x_n = y$, and every two adjacent variables occur together in some atom of $q \setminus \{R\}$. In particular, we have $x, y \notin [R : N]^{\ominus, q}$.

For example, for q_3 in Example 7.2, we have $R : \emptyset \mid x \overset{q_3}{\leftrightsquigarrow} y$.

Lemma 7.4. *Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. If q has a tri-tanglement, then* CERTAINTY(q) *is* **L***-hard (under first-order reductions).*

Proof. Assume that q has a tri-tanglement with keys $[\![R : K]\!]$, $[\![R : L]\!]$, and $[\![R : M]\!]$ as in Definition 7.1. The proof is a first-order reduction from the **L**-complete problem Undirected Forest Accessibility (UFA) [6]:

INPUT: An acyclic undirected graph $G = (V, E)$ consisting of exactly two connected components; two vertices s, t.

QUESTION: Are s and t connected in G?

Let (V, E, s, t) be an instance of UFA. We will assume that the undirected edges of E are encoded in the input as ordered pairs, where each undirected edge $\{a, b\}$ of E is ordered as either (a, b) or (b, a) (but not both). Furthermore, we will assume that $\{s, t\} \notin E$, and that every undirected edge of E that contains s or t is ordered such that the second position is s or t. The problem UFA obviously remains **L**-hard under the restriction that there is no edge $\{s, t\}$. We construct a database instance $\mathbf{db}$ as follows.

For every vertex $a \in V$, $\mathbf{db}$ contains $\theta_a(q)$ where θ_a is the valuation over $\mathrm{vars}(q)$ such that for every $v \in \mathrm{vars}(q)$,

$$\theta_a(v) = \begin{cases} \bot_v & \text{if } v \in [R : K \cap L \cap M]^{\ominus, q} \\ a_v & \text{otherwise} \end{cases}$$

Every fact $\theta_a(R)$ will be called a *vertex-fact.*

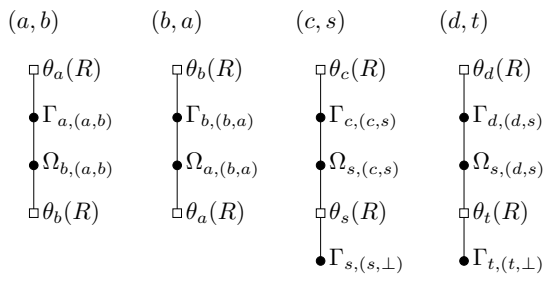


Figure 1: Conflict graph for the R-facts entailed by an undirected edge $\{a, b\}$, which is encoded in the input as the ordered pair (a, b) or (b, a). The edge $\{c, s\}$ is ordered as (c, s). The edge $\{d, t\}$ is ordered as (d, t). For readability, vertex-facts are depicted as □, and edge-facts as •.

For every relation name S with $S \neq R$ that occurs in q, we have that the set of S-facts of **db** is consistent. Indeed, let $a \neq b$, and let $[\![S : N]\!]$ be a key of q with $S \neq R$. If $N \subseteq [R : K \cap L \cap M]^{\ominus,q}$, then $\text{vars}(S) \subseteq [R : K \cap L \cap M]^{\ominus,q}$ and therefore $\theta_a(S) = \theta_b(S)$. If $N \not\subseteq [R : K \cap L \cap M]^{\ominus,q}$, then $\theta_a(S), \theta_b(S)$ do not agree on N.

Then, for every ordered pair (a, b) in the input, **db** contains R-facts $\Gamma_{a,(a,b)}$ and $\Omega_{b,(a,b)}$ defined as follows: for every $x \in \text{vars}(R)$,

$$\Gamma_{a,(a,b)}@x = \begin{cases} \bot_x & \text{if } x \in [R : K \cap L \cap M]^{\ominus,q} \\ a_x & \text{if } x \in K \setminus [R : K \cap L \cap M]^{\ominus,q} \\ b_x & \text{if } x \in (M \cap L) \setminus [R : K \cap L \cap M]^{\ominus,q} \\ \langle a, b\rangle_x & \text{otherwise} \end{cases}$$

$$\Omega_{b,(a,b)}@x = \begin{cases} \bot_x & \text{if } x \in [R : K \cap L \cap M]^{\ominus,q} \\ b_x & \text{if } x \in M \setminus [R : K \cap L \cap M]^{\ominus,q} \\ a_x & \text{if } x \in (K \cap L) \setminus [R : K \cap L \cap M]^{\ominus,q} \\ \langle a, b\rangle_x & \text{otherwise} \end{cases}$$

We will call these *edge-facts*. Finally, **db** contains two special edge-facts $\Gamma_{s,(s,\bot)}$ and $\Gamma_{t,(t,\bot)}$.

Note that edge-facts $\Gamma_{a,(a,b)}$ are well-defined, since the intersection of $K \setminus [R : K \cap L \cap M]^{\ominus,q}$ and $(M \cap L) \setminus [R : K \cap L \cap M]^{\ominus,q}$ is empty, because $K \cap L \cap M \subseteq [R : K \cap L \cap M]^{\ominus,q}$. Likewise, edge-facts $\Omega_{b,(a,b)}$ are well defined.

Claim 7.1. *No key of R is included in $[R : K \cap L \cap M]^{\ominus,q} \cup (K \cap L)$ or $[R : K \cap L \cap M]^{\ominus,q} \cup (M \cap L)$.*

Proof of Claim 7.1. Assume, for the sake of contradiction, a key $[\![R : N]\!]$ such that for some $K' \in \{K, M\}$, we have $N \subseteq [R : K \cap L \cap M]^{\ominus,q} \cup (K' \cap L)$. We treat the case $K' = K$ (the case $K' = M$ is symmetrical). Since $N \subseteq [R : K \cap L \cap M]^{\ominus,q} \cup (K \cap L)$, we have, by intersecting with the set M and distributing intersection over union:

$$N \cap M \subseteq \left(M \cap [R : K \cap L \cap M]^{\ominus,q}\right) \cup (K \cap L \cap M)\,.$$

Since $M \cap [R : K \cap L \cap M]^{\ominus,q} \subseteq [R : K \cap L \cap M]^{\ominus,q}$ and $K \cap L \cap M \subseteq [R : K \cap L \cap M]^{\ominus,q}$, we have $N \cap M \subseteq [R : K \cap L \cap M]^{\ominus,q}$. Thus, $\mathcal{K}(q \ominus R) \models K \cap L \cap M \rightarrow N \cap M$.

From condition (a) in Definition 7.1 of tri-tanglements, we have $\mathcal{K}(q \ominus R) \models M \cap N \rightarrow M$. Therefore, $\mathcal{K}(q \ominus R) \models K \cap L \cap M \rightarrow M$, that is, $M \subseteq [R : K \cap L \cap M]^{\ominus,q}$. But then condition ($b$) in Definition 7.1 is obviously violated, a contradiction. This concludes the proof of Claim 7.1. ∎

Here are some important observations that hold for every ordered edge (a, b):

(a) $\Gamma_{a,(a,b)}$ and $\Omega_{b,(a,b)}$ agree on L.
Rationale: Let $x \in L$. The desired result is obvious if x also belongs to $[R : K \cap L \cap M]^{\ominus,q}$. Assume next that $x \notin [R : K \cap L \cap M]^{\ominus,q}$. Then, if $x \in K \cap L$, we have $\Gamma_{a,(a,b)}@x = a_x = \Omega_{b,(a,b)}@x$; and if $x \in M \cap L$, we have $\Gamma_{a,(a,b)}@x = b_x = \Omega_{b,(a,b)}@x$. In all other cases, $\Gamma_{a,(a,b)}@x = \langle a, b\rangle_x = \Omega_{b,(a,b)}@x$.

(b) $\Gamma_{a,(a,b)}$ and $\theta_a(R)$ agree on K.
Rationale: Let $x \in K$. The desired result is obvious if x also belongs to $[R : K \cap L \cap M]^{\ominus,q}$. Otherwise, $\Gamma_{a,(a,b)}@x = a_x = \theta_a(R)@x$.

(c) $\Omega_{b,(a,b)}$ and $\theta_b(R)$ agree on M.
Rationale: Symmetric to the previous item.

(d) $\{\theta_a(R), \Omega_{b,(a,b)}\}$ is consistent.
Rationale: Consider any key $[\![R : N]\!]$. From Claim 7.1, it follows that N contains a variable x such that $x \notin K \cap L$ and $x \notin [R : K \cap L \cap M]^{\ominus,q}$. We have $\theta_a(R)@x = a_x$ and $\Omega_{b,(a,b)}@x \in \{b_x, \langle a, b\rangle_x\}$, and therefore $\theta_a(R)$ and $\Omega_{b,(a,b)}$ disagree on some variable of N.

(e) $\{\theta_b(R), \Gamma_{a,(a,b)}\}$ is consistent.
Rationale: Symmetric to the previous item.

(f) $\{\theta_a(R), \theta_b(R)\}$ is consistent.

(g) $\theta_a(R)$, $\theta_b(R)$, $\Gamma_{a,(a,b)}$, and $\Omega_{b,(a,b)}$ are pairwise distinct.

Hence, the conflict graph of the set of R-facts resulting from edge $\{a, b\}$ is as in Figure 1.

Note that if a vertex a is adjacent with two distinct vertices, say b and c, then the edges $\{a, b\}$ and $\{a, c\}$ can result in conflicting edge-facts as follows. The orderings (b, a) and (c, a) result in $\Omega_{a,(b,a)}$ and $\Omega_{a,(c,a)}$, two distinct facts that agree on M. Symmetrically, the orderings (a, b) and (a, c) result in $\Gamma_{a,(a,b)}$ and $\Gamma_{a,(a,c)}$, two distinct facts that agree on K. Note incidentally that for the orderings (c, a) and (a, b), the sets $\{\Gamma_{c,(c,a)}, \Gamma_{a,(a,b)}\}$ and $\{\Omega_{a,(c,a)}, \Omega_{b,(a,b)}\}$ are both consistent.

We now show that (V, E, s, t) is a "no"-instance of UFA (i.e., s and t are not connected) if and only if **db** has a falsifying repair.

The construction of a falsifying repair proceeds as follows, without loss of generality. Repeatedly pick a vertex a that is a leaf of the UFA instance under consideration; we can assume a unique vertex b that is adjacent to a. When picking vertices, we avoid picking s, t, or any vertex adjacent to s or t, whenever possible. The corresponding R-fact $\theta_a(R)$ will be a leaf of the conflict graph, conflicting with (and only with) $\Gamma_{a,(a,b)}$ or $\Omega_{a,(b,a)}$ (depending on whether $\{a, b\}$ was ordered as (a, b) or as (b, a); only one of these facts is in **db**). Since a falsifying repair cannot contain $\theta_a(R)$, it must contain $\Gamma_{a,(a,b)}$ or $\Omega_{a,(b,a)}$ (depending on whichever is in **db**). If a falsifying repair contains $\Gamma_{a,(a,b)}$, it cannot contain $\Omega_{b,(a,b)}$; and if it contains $\Omega_{a,(b,a)}$, it cannot contain $\Gamma_{b,(b,a)}$. By construction, since a is a leaf, $\Gamma_{a,(a,b)}$ conflicts with no facts except for $\theta_a(R)$ and $\Omega_{b,(a,b)}$; and $\Omega_{a,(b,a)}$ conflicts with no facts except for $\theta_a(R)$ and $\Gamma_{b,(b,a)}$. We now remove the vertex a and the edge $\{a, b\}$ from the UFA instance, and remove the corresponding vertex-fact and

edge-facts from **db**. Clearly, the removal of a and $\{a, b\}$ does not change the connectedness (or non-connectedness) of s and t; and the removal of the corresponding vertex-fact and edge-facts does not change the existence (or non-existence) of a falsifying repair.

Assume that s and t are connected in the UFA instance. For the connected component that does not contain s, t, the repeated removal ends with some edge $\{a, b\}$. It can be seen that every repair must contain a fact of the form $\theta_c(R)$. A repair containing $\theta_c(R)$ satisfies $\theta_c(q)$, and therefore satisfies q.

Assume that s and t are not connected in the UFA instance. The repeated removal ends with edges $\{c, s\}$ and $\{d, t\}$, ordered as (c, s) and (d, t). By picking $\Gamma_{c,(c,s)}$, $\Gamma_{d,(d,s)}$, $\Gamma_{s,(s,\perp)}$, $\Gamma_{t,(t,\perp)}$, we claim that we obtain a falsifying repair. To prove the latter claim, it suffices to show that for every valuation $\mu(q)$ over $\mathrm{vars}(q)$ such that $\mu(q) \subseteq \mathbf{db}$, we have that $\mu(q)$ contains no edge-facts (i.e., no facts of the form $\Gamma_{a,(a,b)}$ or $\Omega_{b,(a,b)}$). Informally, edge-facts cannot make the query true.

Claim 7.2. *There exist join variables $y, z \in \mathrm{vars}(R)$ such that $y \in (K \cap L) \setminus M$, $z \in (M \cap L) \setminus K$, and $R : K \cap L \cap M \mid y \overset{q}{\rightsquigarrow} z$.*

Proof of Claim 7.2. By condition (a) in Definition 7.1 of tri-tanglements, we have $\mathcal{K}(q \ominus R) \models K \cap L \to K$ and $\mathcal{K}(q \ominus R) \models K \cap M \to M$. It follows that $\mathcal{K}(q \ominus R) \models K \cap L \to M \cap L$. Since $M \cap L \notin [R : K \cap L \cap M]^{\ominus, q}$ by condition (b) in Definition 7.1, we can assume a variable $z \in M \cap L$ such that $z \notin [R : K \cap L \cap M]^{\ominus, q}$. Since $[R : K \cap L \cap M]^{\ominus, q}$ includes $K \cap L \cap M$, we have $z \notin K \cap L \cap M$, and therefore $z \notin K$. Since $\mathcal{K}(q \ominus R) \models K \cap L \to z$, there exists a sequence of keys of q

$$[\![T_1 : M_1]\!], [\![T_2 : M_2]\!], \ldots, [\![T_n : M_n]\!] \tag{5}$$

with $n \geq 1$ such that $z \in \mathrm{vars}(T_n) \setminus M_n$ and for every $i \in \{1, \ldots, n\}$,

- $T_i \neq R$; and
- $M_i \subseteq (K \cap L) \cup \left(\bigcup_{j=1}^{i-1} \mathrm{vars}(T_j) \right)$.

Let $M_0 := K \cap L$. Therefore, $M_0 \notin [R : K \cap L \cap M]^{\ominus, q}$. We claim:

$$\begin{array}{l} \text{for every } i \geq 1, \text{ if } M_i \notin [R : K \cap L \cap M]^{\ominus, q}, \text{ then} \\ \text{there exists } j < i \text{ and variables } x_i \in M_i, x_j \in M_j \\ \text{such that } R : K \cap L \cap M \mid x_j \overset{q}{\rightsquigarrow} x_i. \end{array} \tag{6}$$

To prove the latter claim, assume $M_i \notin [R : K \cap L \cap M]^{\ominus, q}$ with $i \geq 1$. There exists $x_i \in M_i$ such that $x_i \notin [R : K \cap L \cap M]^{\ominus, q}$. If $x_i \in M_0$, we have $R : K \cap L \cap M \mid x_i \overset{q}{\rightsquigarrow} x_i$, the desired result. Assume next $x_i \notin M_0$. By the construction of (5), there exists $j < i$ such that $x_i \in \mathrm{vars}(T_j)$. Since $x_i \notin [R : K \cap L \cap M]^{\ominus, q}$, we have $M_j \notin [R : K \cap L \cap M]^{\ominus, q}$. Therefore, there is some $x_j \in M_j$ such that $x_j \notin [R : K \cap L \cap M]^{\ominus, q}$. The atom T_j shows $R : K \cap L \cap M \mid x_j \overset{q}{\rightsquigarrow} x_i$. The following transitivity property is straightforward, for every $V \subseteq \mathrm{vars}(R)$:

$$\begin{array}{l} \text{if } R : V \mid x \overset{q}{\rightsquigarrow} y \text{ and } R : V \mid y \overset{q}{\rightsquigarrow} z, \text{ then} \\ R : V \mid x \overset{q}{\rightsquigarrow} z. \end{array} \tag{7}$$

Since $z \in \mathrm{vars}(T_n)$ and $z \notin [R : K \cap L \cap M]^{\ominus, q}$, there exists $x_n \in M_n$ such that $x_n \notin [R : K \cap L \cap M]^{\ominus, q}$, and therefore $R : K \cap L \cap M \mid x_n \overset{q}{\rightsquigarrow} z$. By (6) and (7), there exists $y \in K \cap L$ such that $R : K \cap L \cap M \mid y \overset{q}{\rightsquigarrow} z$. From $y \notin [R : K \cap L \cap M]^{\ominus, q}$, it follows $y \notin K \cap L \cap M$, and therefore $y \notin M$.

The proof of Claim 7.2 is concluded by noting that y and z are join variables because they occur in R and in the sequence (5). ∎

Let $\mu(q)$ be a valuation over $\mathrm{vars}(q)$ such that $\mu(q) \subseteq \mathbf{db}$. We need to show that $\mu(R) \in \{\theta_a(R) \mid a \in V\}$. By Claim 7.2, there exist join variables $y, z \in \mathrm{vars}(R)$ such that $y \in (K \cap L) \setminus M$, $z \in (M \cap L) \setminus K$, and $R : K \cap L \cap M \mid y \overset{q}{\rightsquigarrow} z$. Therefore, there exists a sequence of variables $y_0, y_1, \ldots, y_n \notin [R : K \cap L \cap M]^{\ominus, q}$ such that $y_0 = y$, $y_n = z$, and every two adjacent variables in the sequence occur together in some atom of $q \setminus \{R\}$.

We can assume an atom S such that $S \neq R$ and $y_0, y_1 \in \mathrm{vars}(S)$. We can also assume a vertex $a \in V$ such that $\mu(S) = \theta_a(S)$. Consequently, $\mu(y_0) = a_{y_0}$ and $\mu(y_1) = a_{y_1}$. By induction on increasing i, it can be easily seen that for every $i \in \{1, \ldots, n\}$, $\mu(y_i) = a_{y_i}$, and therefore $\mu(y_n) = a_{y_n}$. Thus, $\mu(R)@y = \mu(y) = a_y$ and $\mu(R)@z = \mu(z) = a_z$. It must be the case that $\mu(R) = \theta_a(R)$. Indeed, there is no vertex-fact F such that both $F@y = a_y$ and $F@z = a_z$, because for every edge (c, d), with $c \neq d$, we have

- $\Gamma_{c,(c,d)}@y = c_y$ and $\Gamma_{c,(c,d)}@z = d_z$; and
- $\Omega_{d,(c,d)}@y = c_y$ and $\Omega_{d,(c,d)}@z = d_z$. □

8 FINALE

In the preceding sections, we established necessary conditions for a query q to have a consistent first-order rewriting: acyclicity of the attack graph, and the absence of bi- and tri-tanglements. In this section, we show that these necessary conditions are also sufficient for the existence of a consistent first-order rewriting. An important step in the construction of such a rewriting is the determination of what we call *reifiable variables*; informally, these are variables that can be existentially quantified outside the scope of any universal quantifier, like the variable y in the first-order rewriting (4) of Section 1. This section contains three subsections. In the first subsection, it is shown that if a query has no bi- or tri-tanglement, then every unattacked variable is reifiable. After that, we elaborate on the treatment of free variables, which will emerge during the construction of a first-order rewriting. In the last subsection, we provide the proof of the main Theorem 1.1.

8.1 Reification

Definition 8.1. (**reifiable variables**) Let q be a Boolean conjunctive query, and let $u \in \mathrm{vars}(q)$. If c is a constant, then $q_{[u \mapsto c]}$ denotes the query obtained from q by replacing every occurrence of u by c. We say that u is *reifiable* if for every database instance **db** that is a "yes"-instance of CERTAINTY(q), there exists a constant c (which depends on **db**) such that **db** is a "yes"-instance of CERTAINTY($q_{[u \mapsto c]}$). For every database instance **db**, we define:

$$\mathrm{Reify}(q, u, \mathbf{db}) := \{c \mid c \text{ is a constant and } \mathbf{db} \models q_{[u \mapsto c]}\}.$$

Although Reify is defined relative to any database instance, it will only be used for database instances that are repairs. A fact $A \in \mathbf{db}$ is said to be *relevant for q in* **db** if there exists a valuation θ over $\mathrm{vars}(q)$ such that $A \in \theta(q) \subseteq \mathbf{db}$.

Lemma 8.2. *Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. Assume that q has no bi-tanglement*

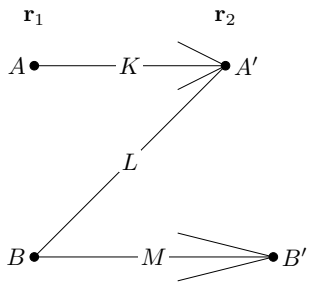


Figure 2: Helping mnemonic for the proof of Lemma 8.2. We have $A, B \in \mathbf{r}_1$, $A' \in \mathbf{r}_2$, and $B' \in \mathbf{db} \setminus \mathbf{r}_1$. It is possible that $B' \notin \mathbf{r}_2$.

and no tri-tanglement. Let $u \in \mathsf{vars}(q)$ be an unattacked variable, that is, $[\![S : L]\!] \overset{q}{\nrightarrow} u$ for every key $[\![S : L]\!]$ of q. Let $\mathbf{r}_1$ and $\mathbf{r}_2$ be repairs of a same database instance $\mathbf{db}$. There exists a repair $\mathbf{s}$ of $\mathbf{db}$ such that $\mathsf{Reify}(q, u, \mathbf{s}) \subseteq \mathsf{Reify}(q, u, \mathbf{r}_1) \cap \mathsf{Reify}(q, u, \mathbf{r}_2)$.

Proof. In the first part of the proof, we show the following:

$$\text{for every } A \in \mathbf{r}_1 \setminus \mathbf{r}_2 \text{ such that } A \text{ is relevant for } q \text{ in } \mathbf{r}_1, \text{ it is possible to construct a repair } \mathbf{r}_1' \text{ of } \mathbf{db} \text{ such that } A \notin \mathbf{r}_1' \text{ and } \mathsf{Reify}(q, u, \mathbf{r}_1') \subseteq \mathsf{Reify}(q, u, \mathbf{r}_1). \tag{8}$$

Assume a fact $A \in \mathbf{r}_1 \setminus \mathbf{r}_2$ such that A is relevant for q in $\mathbf{r}_1$. Let R be the relation name of A. We can assume a valuation α over $\mathsf{vars}(q)$ such that $A \in \alpha(q) \subseteq \mathbf{r}_1$.

Since $A \notin \mathbf{r}_2$, we can assume an R-fact $A' \in \mathbf{r}_2 \setminus \mathbf{r}_1$ and a key K of R such that A and A' agree on K, with $A \neq A'$. Let $\mathbf{r}_1'$ be a repair of $\mathbf{db}$ constructed from $\mathbf{r}_1$ by means of the following three steps, starting with $\mathbf{r}_1' = \mathbf{r}_1$:

(1) *Shrinking step:* delete from $\mathbf{r}_1'$ all R-facts B for which $\{B, A'\}$ is inconsistent (in particular, A will be deleted);
(2) *First growing step:* insert the R-fact A' into $\mathbf{r}_1'$; and
(3) *Second growing step:* insert R-facts from $\mathbf{db}$ into $\mathbf{r}_1'$ as long as consistency is preserved. This step is non-deterministic and guarantees that eventually $\mathbf{r}_1'$ is an inclusion-maximal consistent subset of $\mathbf{db}$.

Obviously, all facts in $\mathbf{r}_1' \setminus \mathbf{r}_1$ are R-facts. Let $f : \mathbf{r}_1' \setminus \mathbf{r}_1 \to \mathcal{P}(\mathbf{r}_1 \setminus \mathbf{r}_1')$ be the total function such that for every fact C' in the domain of f, we have $f(C') := \{C \in \mathbf{r}_1 \mid \{C', C\} \text{ is inconsistent}\}$. For every C' in the domain of f, the set $f(C')$ is a nonempty set of R-facts. Indeed, if there is some $C' \in \mathbf{r}_1' \setminus \mathbf{r}_1$ such that $f(C') = \emptyset$, then $\mathbf{r}_1 \cup \{C'\}$ would be consistent, contradicting that $\mathbf{r}_1$ is an inclusion-maximal consistent subset of $\mathbf{db}$. Note also that for every $C' \in \mathbf{r}_1' \setminus \mathbf{r}_1$, we have $f(C') \subseteq f(A') = \mathbf{r}_1 \setminus \mathbf{r}_1'$.

Claim 8.1. *For every fact $B' \in \mathbf{r}_1' \setminus \mathbf{r}_1$, if B' is relevant for q in $\mathbf{r}_1'$, then some fact of $f(B')$ is relevant for q in $\mathbf{r}_1$.*

Proof of Claim 8.1. Let $B' \in \mathbf{r}_1' \setminus \mathbf{r}_1$ such that B' is relevant for q in $\mathbf{r}_1'$. We can assume a valuation β' over $\mathsf{vars}(q)$ such that $B' \in \beta'(q) \subseteq \mathbf{r}_1'$.

For the sake of contradiction, assume that no fact of $f(B')$ is relevant for q in $\mathbf{r}_1$. Since A is relevant for q in $\mathbf{r}_1$, we have $A \notin f(B')$. Since $A \in f(A')$, we have $A' \neq B'$. Since $f(B') \neq \emptyset$, we can assume the existence of a fact $B \in f(B')$. From $A \notin f(B')$, it follows $A \neq B$. Since $B \in f(B') \subseteq f(A')$, we can assume a key L of R on which B and A' agree. We can also assume a key M of R on which B and B' agree. The situation is depicted in Figure 2. We have $K \neq L$, or else $\{A, B\}$ is an inconsistent subset of $\mathbf{r}_1$, contradicting that $\mathbf{r}_1$ is a repair. We also have $L \neq M$, or else $\{A', B'\}$ is an inconsistent subset of $\mathbf{r}_1'$, contradicting that $\mathbf{r}_1'$ is a repair. Indeed, the *Second growing step* never inserts a fact that agrees with A' on some key.

Since B is not relevant for q in $\mathbf{r}_1$, but B' is relevant for q in $\mathbf{r}_1'$, it must be the case that there exists a join variable $v \in \mathsf{vars}(R) \setminus M$ such that $B@v \neq B'@v$. Since q has no bi-tanglement, if follows that:

$$\text{for every key } [\![R : N]\!], \text{ we have } \mathcal{K}(q \ominus R) \models M \cap N \to M. \tag{9}$$

We now show that $K \neq M$. Assume for the sake of contradiction that $K = M$. Since the valuations α and β' obviously agree on $K \cap L \cap M$, they agree on $K \cap L$. Since $\mathcal{K}(q \ominus R) \models K \cap L \to K$ by (9) with $N = L$, α and β' must agree on K. Therefore, A and B' agree on K. Then A' and B' also agree on K, a contradiction. We conclude by contradiction $K \neq M$. Therefore, $[\![R : K]\!]$, $[\![R : L]\!]$, and $[\![R : M]\!]$ are three distinct keys.

Next, we show that A' is not relevant for q in $\mathbf{r}_1'$. For the sake of contradiction, assume that A' is relevant for q in $\mathbf{r}_1'$. Then there exists a valuation α' over $\mathsf{vars}(q)$ such that $A' \in \alpha'(q) \subseteq \mathbf{r}_1'$. Since $\mathcal{K}(q \ominus R) \models M \cap L \to M$ by (9), and α', β' agree on $M \cap L$, it follows that A' and B' agree on M, contradicting that $\{A', B'\}$ is consistent.

Since A is relevant for q in $\mathbf{r}_1$, but A' is not relevant for q in $\mathbf{r}_1'$, it must be the case that there exists a join variable $v \in \mathsf{vars}(R) \setminus K$ such that $A@v \neq A'@v$. Since q has no bi-tanglement, if follows that:

$$\text{for every key } [\![R : N]\!], \text{ we have } \mathcal{K}(q \ominus R) \models K \cap N \to K. \tag{10}$$

It can be easily seen that all facts among A, A', B, B' agree on $K \cap L \cap M$. It follows that α and β' agree on $K \cap L \cap M$. Since $\mathbf{r}_1$ and $\mathbf{r}_1'$ contain the same set of S-facts for every $S \neq R$, it follows that α and β' agree on $[R : K \cap L \cap M]^{\ominus, q}$.

We next show that $\mathcal{K}(q \ominus R) \not\models K \cap L \cap M \to K$. For the sake of contradiction, assume $\mathcal{K}(q \ominus R) \models K \cap L \cap M \to K$. Then it follows that α and β' agree on K. Consequently, for every $x \in K$, we have $A@x = B'@x$. Since *(i)* A and B' agree on K, and *(ii)* A and A' agree on K, it follows that A' and B' agree on K. Then $\{A', B'\}$ is inconsistent, a contradiction. Using symmetrical reasoning, we can show that $\mathcal{K}(q \ominus R) \not\models K \cap L \cap M \to M$.

It is now easy to see that none of $K \cap L$, $K \cap M$, or $L \cap M$ is included in $[R : K \cap L \cap M]^{\ominus, q}$, which together with (9) and (10) implies that q has a tri-tanglement, which contradicts the hypothesis of the lemma. This concludes the proof of Claim 8.1. ∎

To finish the proof of (8), we show:

$$\mathsf{Reify}(q, u, \mathbf{r}_1') \subseteq \mathsf{Reify}(q, u, \mathbf{r}_1). \tag{11}$$

Recall that $\mathbf{r}_1' \setminus \mathbf{r}_1$ is a set of R-facts. Let $B' \in \mathbf{r}_1' \setminus \mathbf{r}_1$, and let μ' be a valuation over $\mathsf{vars}(q)$ such that $B' \in \mu'(q) \subseteq \mathbf{r}_1'$. Let $c := \mu'(u)$. Therefore, $c \in \mathsf{Reify}(q, u, \mathbf{r}_1')$. It suffices to show $c \in \mathsf{Reify}(q, u, \mathbf{r}_1)$. The reason why this suffices is that for every R-fact $C \in \mathbf{r}_1' \cap \mathbf{r}_1$, for every valuation γ over $\mathsf{vars}(q)$, if $C \in \gamma(q) \subseteq \mathbf{r}_1'$ then $C \in \gamma(q) \subseteq \mathbf{r}_1$.

By Claim 8.1, there is some fact $B \in f(B')$ that is relevant for q in $\mathbf{r}_1$. We can assume a key M of R such that B' and B agree on M. Notice that if $B' = A'$, then B can be taken to be A, and M can be taken to be K. We can assume a valuation μ over $\mathrm{vars}(q)$ such that $B \in \mu(q) \subseteq \mathbf{r}_1$. Clearly, μ and μ' agree on M.

Let κ be the valuation such that for every variable $v \in \mathrm{vars}(q)$,

$$\kappa(v) = \begin{cases} \mu(v) & \text{if } [\![R : M]\!] \overset{q}{\rightsquigarrow} v \\ \mu'(v) & \text{if } [\![R : M]\!] \overset{q}{\not\rightsquigarrow} v \end{cases}$$

Clearly, $\kappa(u) = c$. To conclude the proof of (8), it suffices now to show $\kappa(q) \subseteq \mathbf{r}_1$, which is done in the next two paragraphs.

Let S be an atom of q such that $S \neq R$. Assume there are two variables $v_a, v_u \in \mathrm{vars}(S)$ such that $[\![R : M]\!] \overset{q}{\rightsquigarrow} v_a$ and $[\![R : M]\!] \overset{q}{\not\rightsquigarrow} v_u$ (informally, the subscripts a and u stand for "attacked" and "unattacked" respectively). Then it must be the case that $\mathcal{K}(q \ominus R) \models M \to v_u$. Since μ and μ' agree on M, we have $\mu(v_u) = \mu'(v_u)$, and therefore $\kappa(S) = \mu(S) \in \mathbf{r}_1$. Furthermore, it is clear that if all variables of S are attacked, then $\kappa(S) = \mu(S) \in \mathbf{r}_1 \cap \mathbf{r}'_1$. Symmetrically, if no variable of S is attacked, then $\kappa(S) = \mu'(S) \in \mathbf{r}_1 \cap \mathbf{r}'_1$.

Finally, consider the atom R itself. We need to show $\kappa(R) = \mu(R)$. Assume that $v_u \in \mathrm{vars}(R)$ such that $[\![R : M]\!] \overset{q}{\not\rightsquigarrow} v_u$. Here also, it must be the case that $\mathcal{K}(q \ominus R) \models M \to v_u$. Since μ and μ' agree on M, we have $\mu(v_u) = \mu'(v_u)$, and therefore $\kappa(v_u) = \mu(v_u)$. This concludes the proof of (8).

To conclude the proof of Lemma 8.2, we show that there exists a repair $\mathbf{s}$ of $\mathbf{db}$ such that $\mathsf{Reify}(q, u, \mathbf{s}) \subseteq \mathsf{Reify}(q, u, \mathbf{r}_1) \cap \mathsf{Reify}(q, u, \mathbf{r}_2)$. For this purpose, construct a maximal sequence of pairs of repairs of $\mathbf{db}$:

$$(\mathbf{r}_1^{(0)}, \mathbf{r}_2^{(0)}); (\mathbf{r}_1^{(1)}, \mathbf{r}_2^{(1)}); \ldots; (\mathbf{r}_1^{(n)}, \mathbf{r}_2^{(n)}), \tag{12}$$

where $(\mathbf{r}_1^{(0)}, \mathbf{r}_2^{(0)}) = (\mathbf{r}_1, \mathbf{r}_2)$ and for every $i \geq 1$, $(\mathbf{r}_1^{(i+1)}, \mathbf{r}_2^{(i+1)})$ is constructed as follows (define $\bar{1} := 2$ and $\bar{2} := 1$):

(1) select $\ell \in \{1, 2\}$ and a fact $A \in \mathbf{r}_\ell^{(i)} \setminus \mathbf{r}_{\bar{\ell}}^{(i)}$ such that A is relevant for q in $\mathbf{r}_\ell^{(i)}$. Let R be the relation name of A;
(2) select $A' \in \mathbf{r}_{\bar{\ell}}^{(i)}$ such that $\{A, A'\}$ is inconsistent. Such A' must exist, or else $\mathbf{r}_{\bar{\ell}}^{(i)} \cup \{A\}$ would be consistent, contradicting that $\mathbf{r}_{\bar{\ell}}^{(i)}$ is an inclusion-maximal consistent subset of $\mathbf{db}$;
(3) let $\mathbf{r}_\ell^{(i+1)}$ be the repair obtained by executing the *Shrinking and growing steps* specified at the beginning of the current proof. In particular, in moving from i to $i+1$, the fact A of $\mathbf{r}_\ell^{(i)}$ is replaced with A' (among other replacements). We let $\mathbf{r}_{\bar{\ell}}^{(i+1)} = \mathbf{r}_{\bar{\ell}}^{(i)}$, i.e., the other repair does not change.

We argue that for $i \geq 0$, we have $\mathbf{r}_1^{(i)} \cap \mathbf{r}_2^{(i)} \subsetneq \mathbf{r}_1^{(i+1)} \cap \mathbf{r}_2^{(i+1)}$. This holds true because *(i)* in moving from i to $i+1$, the fact A' is added to the intersection, and *(ii)* once a fact belongs to the intersection of a pair of repairs in (12), it will not be deleted later on. Therefore, the sequence (12) is finite.

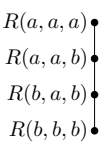
By repeated application of (11), $\mathsf{Reify}(q, u, \mathbf{r}_1^{(n)}) \subseteq \mathsf{Reify}(q, u, \mathbf{r}_1)$ and $\mathsf{Reify}(q, u, \mathbf{r}_2^{(n)}) \subseteq \mathsf{Reify}(q, u, \mathbf{r}_2)$. We also have that for every $\ell \in \{1, 2\}$, if C is a fact of $\mathbf{r}_\ell^{(n)}$ that is relevant for q, then $C \in \mathbf{r}_{\bar{\ell}}^{(n)}$. It follows that the set of facts of $\mathbf{r}_1^{(n)}$ that are relevant for q is equal to the set of facts of $\mathbf{r}_2^{(n)}$ that are relevant for q. Consequently, $\mathsf{Reify}(q, u, \mathbf{r}_1^{(n)}) = \mathsf{Reify}(q, u, \mathbf{r}_2^{(n)})$. It follows that both $\mathsf{Reify}(q, u, \mathbf{r}_1^{(n)}) \subseteq \mathsf{Reify}(q, u, \mathbf{r}_1) \cap \mathsf{Reify}(q, u, \mathbf{r}_2)$ and $\mathsf{Reify}(q, u, \mathbf{r}_2^{(n)}) \subseteq \mathsf{Reify}(q, u, \mathbf{r}_1) \cap \mathsf{Reify}(q, u, \mathbf{r}_2)$. Therefore, both $\mathbf{r}_1^{(n)}$ and $\mathbf{r}_2^{(n)}$ are repairs with the desired property. This concludes the proof of Lemma 8.2. □

Corollary 8.3 (reification). *Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. Assume that q has no bi-tanglement and no tri-tanglement. If u is a variable of $\mathrm{vars}(q)$ that is not attacked by any key of q, then u is reifiable.*

Proof. Let u be a variable of $\mathrm{vars}(q)$ that is not attacked by any key of q. Let $\mathbf{db}$ be a database instance that is a "yes"-instance of CERTAINTY(q). From Lemma 8.2, it follows that there exists a repair $\mathbf{s}$ of $\mathbf{db}$ such that $\emptyset \neq \mathsf{Reify}(q, u, \mathbf{s}) = \bigcap\{\mathsf{Reify}(q, u, \mathbf{r}) \mid \mathbf{r} \in \mathrm{rset}(\mathbf{db})\}$. Clearly, for every $c \in \mathsf{Reify}(q, u, \mathbf{s})$, for every repair $\mathbf{r}$ of $\mathbf{db}$, we have $\mathbf{r} \models q_{[u \mapsto c]}$. □

Example 8.4 shows the existence of unattacked variables that are not reifiable. Corollary 8.3 does not apply to the query of the example, because it has a tri-tanglement.

Example 8.4. Take again the query q_3 of Example 7.2, which has a tri-tanglement. It can be verified that no variable is attacked, and we claim that no variable is reifiable. Indeed, every repair of the following database instance satisfies q_3, by either satisfying $q_{3[xyz \mapsto aaa]}$ or $q_{3[xyz \mapsto bbb]}$ (but not both).



Conflict graph

R	x	y	z
	a	a	a
	a	a	b
	b	a	b
	b	b	b

S^{c}	x	y	z
	a	a	a
	b	b	b

8.2 Treatment of Free Variables

Let q be a self-join-free conjunctive query in which no atom contains constants or repeated variables. Assume now that some variable x of q is free, which is denoted by writing $q(x)$. Our theory so far has been developed for Boolean queries. Nevertheless, we will show how it can be adapted to handle free variables.

We say that a constant b is a *consistent answer* to $q(x)$ on a database instance $\mathbf{db}$ if every repair of $\mathbf{db}$ satisfies $q(b)$ (using the standard notion of satisfaction). Let $q_{[x \mapsto b]}$ be the query obtained from q by replacing every occurrence of x with b. Clearly, every repair of $\mathbf{db}$ satisfies $q(b)$ if and only every repair of $\mathbf{db}$ satisfies $q_{[x \mapsto b]}$. This is an application of [20, Lemma 2.3] when constants are interpreted by themselves. Obviously, $q_{[x \mapsto b]}$ is a Boolean query, and the consistent answers to $q(x)$ on $\mathbf{db}$ are the constants b of $\mathrm{adom}(\mathbf{db})$ for which $q_{[x \mapsto b]}$ holds true in every repair.

A technical difficulty is that $q_{[x \mapsto b]}$ contains the constant b, while most of our results are stated for constant-free queries. We can however apply Lemma 4.2: if v is a fresh variable not occurring elsewhere, then the problems CERTAINTY($q_{[x \mapsto b]}$) and

CERTAINTY$(q_{[x\mapsto v]} \cup \{C_b^{\mathsf{c}}(v)\})$ are equivalent under first-order reductions, where C_b is a fresh relation name of mode c with key dependency $[\![C_b : \emptyset]\!]$, in which the subscript b indicates the constant for which the new atom was introduced. Note here that we treat $q_{[x\mapsto v]} \cup \{C_b^{\mathsf{c}}(v)\}$ as a Boolean query, and therefore v is not free in this query. We could have reused x instead of v, but this would lead to some name clashes in the treatment later on. In short, for every database instance **db**, we have that $q_{[x\mapsto b]}$ is true in every repair of **db** if and only if $q_{[x\mapsto v]} \cup \{C_b^{\mathsf{c}}(v)\}$ is true in every repair of $\mathbf{db} \cup \{C_b^{\mathsf{c}}(b)\}$.

Now let φ be a consistent first-order rewriting of the Boolean query $q_{[x\mapsto v]} \cup \{C_b^{\mathsf{c}}(v)\}$. It is possible that φ, which is obviously Boolean, contains some atom $C_b^{\mathsf{c}}(y)$. In any repair of $\mathbf{db} \cup \{C_b^{\mathsf{c}}(b)\}$, this atom is true if and only if $y = b$. A consistent first-order rewriting of CERTAINTY$(q_{[x\mapsto b]})$ is therefore obtained by replacing any atom $C_b^{\mathsf{c}}(y)$ in φ with the equality $y = b$.

Finally, since query evaluation is closed under renaming of constants, if a is another constant, then a consistent first-order rewriting of $q_{[x\mapsto v]} \cup \{C_a^{\mathsf{c}}(v)\}$ can be obtained from a consistent first-order rewriting of $q_{[x\mapsto v]} \cup \{C_b^{\mathsf{c}}(v)\}$ by renaming b into a. This means that we could treat the free variable x as a generic constant (instead of using b), and consider the Boolean query $q_{[x\mapsto v]} \cup \{C_x^{\mathsf{c}}(v)\}$ (instead of $q_{[x\mapsto v]} \cup \{C_b^{\mathsf{c}}(v)\}$).

Example 8.5. Let $q(x) = \exists u R(\underline{u}, x)$, where x is a free variable. We have that b is a consistent answer to $q(x)$ on some database instance **db** if $\exists u R(\underline{u}, b)$ is true in every repair. Elimination of b yields the query $\exists u \exists v \left(R(\underline{u}, v) \wedge C_b^{\mathsf{c}}(v)\right)$, which has the following consistent first-order rewriting:

$$\exists u \left(\left(\exists v R(\underline{u}, v)\right) \wedge \forall v \left(R(\underline{u}, v) \rightarrow C_b^{\mathsf{c}}(v)\right)\right).$$

We obtain a consistent first-order rewriting of $\exists u R(\underline{u}, b)$ by replacing $C_b^{\mathsf{c}}(v)$ with $v = b$:

$$\exists u \left(\left(\exists v R(\underline{u}, v)\right) \wedge \forall v \left(R(\underline{u}, v) \rightarrow v = b\right)\right).$$

The detour via b can be avoided by eliminating from $q(x)$ the free variable x as if it were a constant, resulting in the query $\exists u \exists v \left(R(\underline{u}, v) \wedge C_x^{\mathsf{c}}(v)\right)$, which has the following consistent first-order rewriting:

$$\exists u \left(\left(\exists v R(\underline{u}, v)\right) \wedge \forall v \left(R(\underline{u}, v) \rightarrow C_x^{\mathsf{c}}(v)\right)\right).$$

We obtain a consistent first-order rewriting of $\exists u R(\underline{u}, x)$ by replacing $C_x^{\mathsf{c}}(v)$ with $v = x$:

$$\exists u \left(\left(\exists v R(\underline{u}, v)\right) \wedge \forall v \left(R(\underline{u}, v) \rightarrow v = x\right)\right).$$

The following helping lemma shows that Corollary 8.3, which was stated for one unattacked variable u, extends to any set of unattacked variables.

Lemma 8.6. *Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. Assume that q has no bi-tanglement and no tri-tanglement. Let $x \in \mathsf{vars}(q)$ such that x is not attacked by any key of q. Let $q' := q \cup \{C^{\mathsf{c}}(x)\}$, where C is a fresh relation name of mode* c *with key $[\![C^{\mathsf{c}} : \emptyset]\!]$. Then,*

(1) q' has no bi-tanglement and no tri-tanglement; and
(2) if there is a key $[\![R : K]\!]$ of q' and a variable $v \in \mathsf{vars}(q')$ such that $[\![R : K]\!] \overset{q'}{\rightsquigarrow} v$, then $R \neq C$ and $[\![R : K]\!] \overset{q}{\rightsquigarrow} v$.

Proof. Note that x is a join variable in q'. The proof of the second item is straightforward. For the first item, it suffices to show the following:

(a) if q' has a bi-tanglement, then q has a bi-tanglement; and
(b) if q' has a tri-tanglement and q has no bi-tanglement, then q has a tri-tanglement.

(a) Assume q' has a bi-tanglement. Then q' contains an atom R of mode i with two distinct keys $[\![R : K]\!], [\![R : L]\!]$ such that $K \not\subseteq [R : K \cap L]^{\ominus, q'}$ and there is $y \in \mathsf{vars}(R) \setminus K$ such that y is a join variable in q'. Obviously, $R \neq C$. It is easily verified that $[R : K \cap L]^{\ominus, q} \subseteq [R : K \cap L]^{\ominus, q'}$, and therefore $K \not\subseteq [R : K \cap L]^{\ominus, q}$. If y is a join variable in q, then q obviously has a bi-tanglement. Assume, for the sake of contradiction, that y is not a join variable in q. Then, it is necessarily the case that $y = x$ and x occurs only once in q. Since $y \in \mathsf{vars}(R) \setminus K$ and $[\![R : K]\!] \overset{q}{\not\rightsquigarrow} y$ by the lemma's hypothesis, it follows $\mathcal{K}(q \setminus \{R\}) \models K \rightarrow y$, which implies that y must be a join variable in q, a contradiction.

(b) Assume q' has a tri-tanglement and q has no bi-tanglement. By (a), q' has no bi-tanglement. Then q' contains an atom R of mode i with three distinct keys $[\![R : K]\!], [\![R : L]\!], [\![R : M]\!]$ such that

(A) for every key $[\![R : N]\!]$, we have $K \subseteq [R : K \cap N]^{\ominus, q'}$ and $M \subseteq [R : M \cap N]^{\ominus, q'}$; and
(B) $[R : K \cap L \cap M]^{\ominus, q'}$ includes no set among $K \cap L$, $K \cap M$, $L \cap M$.

Since $[R : K \cap L \cap M]^{\ominus, q} \subseteq [R : K \cap L \cap M]^{\ominus, q'}$, it follows that $[R : K \cap L \cap M]^{\ominus, q}$ includes no set among $K \cap L$, $K \cap M$, $L \cap M$. It remains to be shown that for every key $[\![R : N]\!]$, we have $K \subseteq [R : K \cap N]^{\ominus, q}$ and $M \subseteq [R : M \cap N]^{\ominus, q}$. To this end, assume a key $[\![R : N]\!]$. We will show $K \subseteq [R : K \cap N]^{\ominus, q}$ (the proof of $M \subseteq [R : M \cap N]^{\ominus, q}$ is symmetrical). Assume for the sake of contradiction that $K \not\subseteq [R : K \cap N]^{\ominus, q}$. Then there is $w \in K$ such that $w \notin [R : K \cap N]^{\ominus, q}$, and therefore $w \in K \setminus N$. Since $\mathcal{K}(q' \setminus \{R\}) \models K \cap N \rightarrow w$ by item (A), it follows $\mathcal{K}(q \setminus \{R\}) \models (K \cap N) \cup \{x\} \rightarrow w$. Consequently, if $w \neq x$, then w is necessarily a join variable in q, and therefore w is also a join variable in q'. If $w = x$, then w is obviously a join variable in q'. Therefore, for the query q', we have that w is a join variable in $\mathsf{vars}(R) \setminus N$. Since q' has no bi-tanglement, $N \subseteq [R : N \cap K]^{\ominus, q'}$. Since N and K are not comparable by set inclusion, we can assume a variable $v \in N \setminus K$. Since $\mathcal{K}(q' \setminus \{R\}) \models N \cap K \rightarrow v$, it follows that v is a join variable in q'. If $v \neq x$, then v is obviously also a join variable in q. If $v = x$, then since $v \in N \setminus K$ and $[\![R : K]\!] \overset{q}{\not\rightsquigarrow} v$ by the lemma's hypothesis, it must be the case that $\mathcal{K}(q \setminus \{R\}) \models K \rightarrow v$, and therefore v is a join variable in q. So we conclude that v is a join variable in q such that $v \in \mathsf{vars}(R) \setminus K$. From our hypothesis that $K \not\subseteq [R : K \cap N]^{\ominus, q}$, it follows that q has a bi-tanglement, a contradiction. □

8.3 Proof of the Main Theorem

We can now give the proof of the main Theorem 1.1

Proof of Theorem 1.1. By Corollary 4.3, we can compute a query p such that

- no atom of p contains constants or repeated variables; and

- CERTAINTY(q) and CERTAINTY(p) are equivalent under first-order reductions.

The following statements hold true:

- if the attack graph of p is cyclic, then CERTAINTY(q) is **L**-hard (and therefore not in **FO**) by Lemma 5.5;
- if p has a bi-tanglement, then CERTAINTY(p) is **coBPM**-hard (and therefore not in **FO**) by Lemma 6.4; and
- if p has a tri-tanglement, then CERTAINTY(p) is **L**-hard (and therefore not in **FO**) by Lemma 7.4.

From here on, assume that the attack graph of p is acyclic, that p has no bi-tanglement, and that p has no tri-tanglement. We show that under these assumptions, CERTAINTY(p) is in **FO**.

The proof is by induction on the number of atoms of p that are of mode i. The desired result is trivial if this number is zero, i.e., if all atoms of p are of mode c. Assume next that at least one atom of p is of mode i. Note that atoms of mode c have zero outdegree in the attack graph of p.

Since the attack graph of p is acyclic, there is an atom R^{i} of p such that R has mode i and has zero indegree in the attack graph of q. Let U be the set of unattacked variables in vars(R). Note here that even though R has zero indegree in the attack graph of p, it is still possible that some key of R attacks some variable of vars(R). Let J be the set of join variables of vars(R). By Lemma 6.5, either every key of R is included in U, or J is included in every key of R (or both). Consider these two cases, where we assume that the R-atom is $R(x_1, \ldots, x_n)$.

Case that every key of R is included in U. Let the keys of R be $[\![R : K_1]\!]$, …, $[\![R : K_k]\!]$, and let $U = \{x_1, \ldots, x_m\}$, which includes every key of R. By Corollary 8.3 and Lemma 8.6, the following are equivalent for every database instance **db**:

(1) p is true in every repair of **db**;
(2) there exist constants $c_1, \ldots, c_m$ (which depend on **db**) such that $q_{[x_1,\ldots,x_m \mapsto c_1,\ldots,c_m]}$ is true in every repair of **db**.

In the following consistent first-order rewriting, the negated existential subformula expresses that there must be no fact that both agrees with $R(x_1, \ldots, x_n)$ on some key, and disagrees with $R(x_1, \ldots, x_n)$ on some of the unattacked variables:

$$\exists x_1 \cdots \exists x_m \left(\begin{array}{l} (\exists x_{m+1} \cdots \exists x_n R(x_1, \ldots, x_n)) \\ \\ \wedge \neg \exists u_1 \cdots \exists u_n \left(\begin{array}{l} R(u_1, \ldots, u_n) \\ \wedge \left(\bigvee_{1 \le \ell \le m} x_\ell \neq u_\ell \right) \wedge \\ \wedge \left(\bigvee_{1 \le j \le k} \left(\bigwedge_{\{i \mid x_i \in K_j\}} x_i = u_i \right) \right) \end{array} \right) \\ \\ \wedge \forall x_{m+1} \cdots \forall x_n \left(R(x_1, \ldots, x_n) \rightarrow \varphi(x_1, \ldots, x_n) \right) \end{array} \right), \tag{13}$$

where φ is a consistent first-order rewriting of $p \setminus \{R\}$ in which the variables $x_1, \ldots, x_n$ are free.

Case that J is included in every key of R. Let $J = \{x_1, \ldots, x_m\}$. Clearly, no variable in J is attacked. By Corollary 8.3 and Lemma 8.6, the following are equivalent for every database instance **db**:

(1) p is true in every repair of **db**;
(2) there exist constants $c_1, \ldots, c_m$ (which depend on **db**) such that $q_{[x_1,\ldots,x_k \mapsto c_1,\ldots,c_m]}$ is true in every repair of **db**.

The consistent first-order rewriting is:

$$\exists x_1 \cdots \exists x_n \left(R(x_1, \ldots, x_n) \wedge \varphi(x_1, \ldots, x_n) \right), \tag{14}$$

where φ is a consistent first-order rewriting of $p \setminus \{R\}$ in which the variables $x_1, \ldots, x_n$ are free.

To conclude the proof, we note that by Lemma 8.6, the induction hypothesis holds for the query obtained from p by treating the variables of vars(R) as free variables. Therefore, the formulas $\varphi(x_1, \ldots, x_n)$ in (13) and (14) exist. □

We point out that the length of the consistent first-order rewriting in the previous proof is at most quadratic in the length of the input, when we consider that the input consists of a query $q \in$ sjfBCQ and a set of key dependencies. This length arises because formula (13) contains a disjunction ranging over all keys of R, in which every disjunct is a conjunction ranging over all key positions. Note incidentally that every relation name R can have a number of keys that is exponential in the arity of R.

9 BEYOND FIRST-ORDER REWRITABILITY

In this section, we briefly look at queries that do not have a consistent first-order rewriting. As we mentioned in the introduction, for the case of primary keys, there exists a clean classification: if CERTAINTY(q) is not in **FO**, then it is either **L**-complete or **coNP**-complete. However, for the case of multiple keys, it is unlikely that we can achieve such a classification. As the next proposition shows, there exists a query q for which CERTAINTY(q) is equivalent (under logspace reductions) to the complement of BPM, a problem whose complexity is still unsettled.

Proposition 9.1. *Let $q_0 = \{R(x, y), S^{\mathsf{c}}(y)\}$, where the keys are $[\![R : \{x\}]\!]$, $[\![R : \{y\}]\!]$, and $[\![S^{\mathsf{c}} : \emptyset]\!]$. Then* CERTAINTY($q_0$) *and the complement of BPM are equivalent under logspace reductions.*

Proof. For the one direction, note that q_0 has a bi-tanglement. Hence by Lemma 6.4, there exists a first-order reduction from the complement of BPM to CERTAINTY(q_0).

We next show a reduction from the complement of the problem CERTAINTY(q_0) to BPM. Let **db** be an instance of CERTAINTY(q_0). Since S^{c} is a relation name of mode c and its key is the empty set, **db** contains at most one S^{c}-fact. If **db** contains no S^{c}-fact (which can be tested in **FO**), then it is obviously a "no"-instance of CERTAINTY(q_0) and the desired result obtains. Assume next that **db** contains $S(b_0)$. We now construct a bipartite graph (P_1, P_2, E) with $|P_1| = |P_2|$ as follows. Let $A = \{a \mid R(a, b_0) \in \mathbf{db}\}$ and $B = \{b \mid \exists a \in A : R(a, b) \in \mathbf{db}\} \setminus \{b_0\}$. It is easy to see that if $|A| > |B|$ (which can be tested in **L**), then **db** is a "yes"-instance of CERTAINTY(q_0) and the desired result obtains.

Assume $|A| \le |B|$ from here on, and let $k = |B| - |A|$. Furthermore,

$$P_1 := A \cup \{a_1, \ldots, a_k\} \text{ and } P_2 := B$$
$$E := \{(a, b) \mid R(a, b) \in \mathbf{db}, a \in A, b \in B\} \cup \{a_1, \ldots, a_k\} \times B$$

where $a_1, \ldots, a_k$ are distinct fresh constants. We now claim that **db** is a "no"-instance of CERTAINTY(q_0) if and only if (P_1, P_2, E) has a perfect matching.

Indeed, consider a perfect matching M. Let $\mathbf{r} = \{R(a, b) \mid (a, b) \in M, a \in A\}$. It is easy to see that **r** is a consistent subset of **db**. Observe

that any repair $\mathbf{r}' \supseteq \mathbf{r}$ does not contain facts of the form $R(a, b_0)$, and hence falsifies q_0.

For the opposite direction, let $\mathbf{r}$ be a repair of $\mathbf{db}$ that falsifies q_0. Since $\mathbf{r}$ contains no fact of the form $R(a, b_0)$, for every $a \in A$, there exists $b \in B$ such that $R(a, b) \in \mathbf{r}$. We can now construct a perfect matching by taking these edges, and matching the remaining k unmatched vertices in B with $\{a_1, \ldots, a_k\}$. □

10 CONCLUSION

For consistent query answering with respect to key dependencies, we solved the following problem: given a self-join-free Boolean conjunctive query q, decide whether q has a consistent first-order rewriting, and construct such a rewriting if it exists.

We terminate this paper with discussing some open problems in consistent query answering with respect to key dependencies. We denote by BCQ the set of Boolean conjunctive queries, possibly with self-joins. By UBCQ, we denote the set of queries of the form $q_1 \vee \cdots \vee q_n$ where each q_i belongs to BCQ.

An open problem is to find an algorithm for the following problem: given $q \in$ BCQ, decide whether CERTAINTY(q) is in **FO**, and if it is, construct a consistent first-order rewriting of q. Another open problem is to prove that for every query $q \in$ BCQ, CERTAINTY(q) is either in **PTIME** or **coNP**-complete. Both problems are open even in the case where no relation name has more than one key

These problems could also be studied for larger query classes, starting with UBCQ. A caveat here is that a **PTIME-coNP**-complete dichotomy in $\{\text{CERTAINTY}(q) \mid q \in \text{UBCQ}\}$ is probably very hard to prove, as it is known that such a dichotomy implies Bulatov's complexity dichotomy for conservative CSPs [11]. In the light of this relationship, it may be rewarding to study whether dichotomies in CSPs can be used to prove dichotomies in CQA. Beyond UBCQ, it is shown in [14, Theorem 1] that the following problem is undecidable, even for primary keys: given a first-order query q, does q have a consistent first-order rewriting?

It follows from our previous work that in the case where no relation name has more than one key, the set $\{\text{CERTAINTY}(q) \mid q \in \text{sjfBCQ}\}$ exhibits a dichotomy between **L** and **coNP**-complete. From the current paper, it follows that if relation names can have more than one key, the set $\{\text{CERTAINTY}(q) \mid q \in \text{sjfBCQ}\}$ contains problems that are neither in **L** nor **coNP**-complete (under standard complexity assumptions). It is an open problem to obtain a more fine-grained complexity classification of problems in this set, which, as we have seen, contains problems equivalent to the complement of BPM under logspace reductions. The complexity of BPM is a notorious open problem as of today.

A final open problem is to move from key dependencies to more general functional dependencies. It is currently not clear to us whether and how the techniques in the current paper can be extended to account for functional dependencies.

ACKNOWLEDGMENTS

Paraschos Koutris was supported by NSF grants III-1910014 and CRII-1850348.

REFERENCES

[1] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. 1999. Consistent Query Answers in Inconsistent Databases. In *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania, USA*, Victor Vianu and Christos H. Papadimitriou (Eds.). ACM Press, 68–79. https://doi.org/10.1145/303976.303983

[2] Leopoldo E. Bertossi. 2011. *Database Repairing and Consistent Query Answering*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00379ED1V01Y201108DTM020

[3] Leopoldo E. Bertossi. 2019. Database Repairs and Consistent Query Answering: Origins and Further Developments. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, Dan Suciu, Sebastian Skritek, and Christoph Koch (Eds.). ACM, 48–58. https://doi.org/10.1145/3294052.3322190

[4] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. 2007. Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family. *J. Autom. Reasoning* 39, 3 (2007), 385–429. https://doi.org/10.1007/s10817-007-9078-x

[5] Ashok K. Chandra, Larry J. Stockmeyer, and Uzi Vishkin. 1984. Constant Depth Reducibility. *SIAM J. Comput.* 13, 2 (1984), 423–439. https://doi.org/10.1137/0213028

[6] Stephen A. Cook and Pierre McKenzie. 1987. Problems Complete for Deterministic Logarithmic Space. *J. Algorithms* 8, 3 (1987), 385–394. https://doi.org/10.1016/0196-6774(87)90018-6

[7] Jan Van den Bussche and Marcelo Arenas (Eds.). 2018. *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*. ACM. https://doi.org/10.1145/3196959

[8] Akhil A. Dixit and Phokion G. Kolaitis. 2019. A SAT-Based System for Consistent Query Answering. In *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings (Lecture Notes in Computer Science)*, Mikolás Janota and Inês Lynce (Eds.), Vol. 11628. Springer, 117–135. https://doi.org/10.1007/978-3-030-24258-9_8

[9] Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. 2016. Bipartite perfect matching is in quasi-NC. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, Daniel Wichs and Yishay Mansour (Eds.). ACM, 754–763. https://doi.org/10.1145/2897518.2897564

[10] Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. 2017. Guest Column: Parallel Algorithms for Perfect Matching. *SIGACT News* 48, 1 (2017), 102–109. https://doi.org/10.1145/3061640.3061655

[11] Gaëlle Fontaine. 2015. Why Is It Hard to Obtain a Dichotomy for Consistent Query Answering? *ACM Trans. Comput. Log.* 16, 1 (2015), 7:1–7:24. https://doi.org/10.1145/2699912

[12] Ariel Fuxman and Renée J. Miller. 2005. First-Order Query Rewriting for Inconsistent Databases. In *Database Theory - ICDT 2005, 10th International Conference, Edinburgh, UK, January 5-7, 2005, Proceedings (Lecture Notes in Computer Science)*, Thomas Eiter and Leonid Libkin (Eds.), Vol. 3363. Springer, 337–351. https://doi.org/10.1007/978-3-540-30570-5_23

[13] Ariel Fuxman and Renée J. Miller. 2007. First-order query rewriting for inconsistent databases. *J. Comput. Syst. Sci.* 73, 4 (2007), 610–635. https://doi.org/10.1016/j.jcss.2006.10.013

[14] Floris Geerts, Fabian Pijcke, and Jef Wijsen. 2017. First-order under-approximations of consistent query answers. *Int. J. Approx. Reasoning* 83 (2017), 337–355. https://doi.org/10.1016/j.ijar.2016.10.005

[15] Phokion G. Kolaitis and Enela Pema. 2012. A dichotomy in the complexity of consistent query answering for queries with two atoms. *Inf. Process. Lett.* 112, 3 (2012), 77–85. https://doi.org/10.1016/j.ipl.2011.10.018

[16] Paraschos Koutris and Dan Suciu. 2014. A Dichotomy on the Complexity of Consistent Query Answering for Atoms with Simple Keys. In *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014*. 165–176. https://doi.org/10.5441/002/icdt.2014.19

[17] Paraschos Koutris and Jef Wijsen. 2017. Consistent Query Answering for Self-Join-Free Conjunctive Queries Under Primary Key Constraints. *ACM Trans. Database Syst.* 42, 2 (2017), 9:1–9:45. https://doi.org/10.1145/3068334

[18] Paraschos Koutris and Jef Wijsen. 2018. Consistent Query Answering for Primary Keys and Conjunctive Queries with Negated Atoms, See [7], 209–224. https://doi.org/10.1145/3196959.3196982

[19] Paraschos Koutris and Jef Wijsen. 2019. Consistent Query Answering for Primary Keys in Logspace. In *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal (LIPIcs)*, Pablo Barceló and Marco Calautti (Eds.), Vol. 127. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 23:1–23:19. https://doi.org/10.4230/LIPIcs.ICDT.2019.23

[20] Leonid Libkin. 2004. *Elements of Finite Model Theory*. Springer. https://doi.org/10.1007/978-3-662-07003-1

[21] Ester Livshits, Benny Kimelfeld, and Sudeepa Roy. 2018. Computing Optimal Repairs for Functional Dependencies, See [7], 225–237. https://doi.org/10.1145/3196959.3196980

[22] Jef Wijsen. 2010. On the first-order expressibility of computing certain answers to conjunctive queries over uncertain databases. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, Jan Paredaens and Dirk Van

Gucht (Eds.). ACM, 179–190. https://doi.org/10.1145/1807085.1807111
[23] Jef Wijsen. 2012. Certain conjunctive query answering in first-order logic. *ACM Trans. Database Syst.* 37, 2 (2012), 9:1–9:35. https://doi.org/10.1145/2188349.2188351
[24] Jef Wijsen. 2019. Foundations of Query Answering on Inconsistent Databases. *SIGMOD Rec.* 48, 3 (2019), 6–16. https://doi.org/10.1145/3377391.3377393

A PROOF OF LEMMA 3.1

Proof of Lemma 3.1. Let $R^{\mathsf{c}}(t_1, \ldots, t_n)$ be an atom of q such that R is a relation name of mode c. For every KD $[\![R : K]\!]$ of R, Let $R_{K,1}(t_1, \ldots, t_n)$ and $R_{K,2}(t_1, \ldots, t_n)$ be two atoms, where for $i \in \{1, 2\}$, we have that $R_{K,i}$ is a fresh relation name of mode i whose set of KDs is the singleton $\{[\![R_{K,i} : K]\!]\}$. Let

$$q' = \quad (q \setminus \{R^{\mathsf{c}}(t_1, \ldots, t_n)\}) \\ \cup \{R_{K,i}(t_1, \ldots, t_n) \mid 1 \leq i \leq 2 \text{ and } [\![R : K]\!] \text{ is a KD of } R\}.$$

We show that CERTAINTY(q) and CERTAINTY(q') are equivalent under first-order reductions.

Reduction from CERTAINTY(q) *to* CERTAINTY(q'). Let **db** an instance of CERTAINTY(q). By definition, the set of R-facts of **db** is consistent. Construct an instance **db′** of CERTAINTY(q') as follows: for every R-fact $R^{\mathsf{c}}(\vec{a}) \in \mathbf{db}$ and KD $[\![R : K]\!]$, **db′** contains $R_{K,1}(\vec{a})$ and $R_{K,2}(\vec{a})$. Clearly, for $i \in \{1, 2\}$, the set of $R_{K,i}$-facts of **db′** is consistent. It is now easy to see that **db** is a "yes"-instance of q if and only of **db′** is a "yes"-instance of CERTAINTY(q').

Reduction from CERTAINTY(q') *to* CERTAINTY(q). Let **db′** be an instance of CERTAINTY(q'). For every KD $[\![R : K]\!]$ of R, let Q_K be the following first-order query:

$$Q_K(x_1, \ldots, x_n) \coloneqq \\ R_{K,1}(x_1, \ldots, x_n) \wedge R_{K,2}(x_1, \ldots, x_n) \\ \wedge \forall y_1 \cdots \forall y_n \left(\begin{array}{r} \big(R_{K,1}(y_1, \ldots, y_n) \wedge (\bigwedge_{i \in K} x_i = y_i)\big) \\ \rightarrow \big(\bigwedge_{1 \leq i \leq n} x_i = y_i\big) \end{array} \right) \\ \wedge \forall y_1 \cdots \forall y_n \left(\begin{array}{r} \big(R_{K,2}(y_1, \ldots, y_n) \wedge (\bigwedge_{i \in K} x_i = y_i)\big) \\ \rightarrow \big(\bigwedge_{1 \leq i \leq n} x_i = y_i\big) \end{array} \right).$$

Let Q be the following first-order query:

$$Q(x_1, \ldots, x_n) \coloneqq \bigwedge \{Q_K(x_1, \ldots, x_n) \mid [\![R : K]\!] \text{ is a KD of } R\}.$$

Let $\mathbf{db} = \mathbf{db'} \cup \{R^{\mathsf{c}}(a_1, \ldots, a_n) \mid \mathbf{db'} \models Q(a_1, \ldots, a_n)\}$. By our construction, the set of R-facts of **db** is consistent, and therefore **db** is a legal instance of CERTAINTY(q). Indeed, for assume that **db** contains two R-facts $R^{\mathsf{c}}(\vec{a})$ and $R^{\mathsf{c}}(\vec{b})$ that agree on K for some KD $[\![R : K]\!]$ of R. Let $i \in \{1, 2\}$. Since $\mathbf{db} \models Q(\vec{a})$ and $\mathbf{db} \models Q(\vec{b})$, we have that $\mathbf{db'} \models Q_K(\vec{a})$ and $\mathbf{db'} \models Q_K(\vec{b})$. Consequently, we have $R_{K,i}(\vec{a}), R_{K,i}(\vec{b}) \in \mathbf{db'}$, and the universally quantified subformula of Q_K that contains $R_{K,i}(y_1, \ldots, y_n)$ states that $\vec{a} = \vec{b}$. We show next that **db** is a "yes"-instance of CERTAINTY(q) if and only if **db′** is a "yes"-instance of CERTAINTY(q').

By our construction of Q, the following are equivalent for every $\vec{a}$ of length arity(R):

(1) $R^{\mathsf{c}}(\vec{a}) \in \mathbf{db}$;
(2) for every $i \in \{1, 2\}$ and every KD $[\![R : K]\!]$ of R, we have that $R_{K,1}(\vec{a})$ and $R_{K,2}(\vec{a})$ belong to every repair of **db′**.

It is now obvious that if **db** is a "yes"-instance of CERTAINTY(q), then **db′** is a "yes"-instance of CERTAINTY(q'). For the opposite direction, assume that **db** is a "no"-instance of CERTAINTY(q). Let **r** be a repair of **db** that falsifies q. Let $\mathbf{r}_{\bar{R}}$ be the set of all facts of **r** that are not R-facts.

We need to show that $\mathbf{r}_{\bar{R}}$ can be extended into a repair of **db′** that falsifies q'. Assume that for some KD $[\![R : L]\!]$ of R and $i \in \{1, 2\}$, we have $R_{L,i}(\vec{a}) \in \mathbf{db'}$ but $R^{\mathsf{c}}(\vec{a}) \notin \mathbf{db}$. Then, for some KD $[\![R : K]\!]$ of R, there exist $\ell, m \in \{1, 2\}$ with $\ell \neq m$ such that $R_{K,\ell}(\vec{a}) \in \mathbf{db'}$ and one of the following cases holds:

(1) $R_{K,m}(\vec{a}) \notin \mathbf{db'}$; or
(2) there exists $\vec{b}$ such that $R_{K,m}(\vec{a})$ and $R_{K,m}(\vec{b})$ are distinct facts of **db′** that agree on K.

Then, we extend $\mathbf{r}_{\bar{R}}$ with $R_{K,\ell}(\vec{a})$, and in the second case, also with $R_{K,m}(\vec{b})$. It is clear that in this way, $\mathbf{r}_{\bar{R}}$ can be extended into a repair of **db′** that falsifies q'. □

B PROOF OF LEMMA 4.1

Proof of Lemma 4.1. The reduction from CERTAINTY(q) to the problem CERTAINTY($(q \setminus \{F\}) \cup p$) is straightforward. Let **db** be an instance of CERTAINTY(q). Let $\mathbf{db'} = \mathbf{db} \cup \{S^{\mathsf{c}}(a, a) \mid a \in \text{adom}(\mathbf{db})\}$. Then, **db** is a "yes"-instance of CERTAINTY(q) if and only if **db′** is a "yes"-instance of CERTAINTY($(q \setminus \{F\}) \cup p$). Informally, we use the S-facts to encode equality.

For the reduction from CERTAINTY($(q \setminus \{F\}) \cup p$) to the problem CERTAINTY(q), we establish a first-order definable function f that maps instances of the problem CERTAINTY($(q \setminus \{F\}) \cup p$) to instances of CERTAINTY(q), such that for every instance **db** of CERTAINTY($(q \setminus \{F\}) \cup p$), the following are equivalent:

- **db** is a "yes"-instance of CERTAINTY($(q \setminus \{F\}) \cup p$); and
- $f(\mathbf{db})$ is a "yes"-instance of CERTAINTY(q).

Let **db** be an instance of CERTAINTY($(q \setminus \{F\}) \cup p$). The variable v occurs exactly twice in $(q \setminus \{F\}) \cup p$: once in the atom $F' = R(t'_1, \ldots, t'_n)$ with $t'_i = x$ and $t'_j = v$, and once in the atom $S^{\mathsf{c}}(x, v)$.

As argued in [17, page 9:7], we can assume that **db** is *typed*, which in particular means that for all (not necessarily distinct) facts $R(a_1, \ldots, a_n)$, $R(b_1, \ldots, b_n)$, $S^{\mathsf{c}}(c, d)$ in **db**, we have $d \neq a_i \neq b_j$ and $c \neq a_j \neq b_i$. Informally, columns that correspond to distinct variables have no constants in common. Let f map every R-fact $R(a_1, \ldots, a_n)$ in **db** to

$$R(a_1, \ldots, a_{j-1}, t, a_{j+1}, \ldots, a_n),$$

where t is as follows:

- if for some constant c, the database instance **db** contains $S^{\mathsf{c}}(c, a_j)$, then $t = c$. Since S has mode c, this constant c, if it exists, is unique;
- otherwise $t = a_j$.

The following claim has a straightforward proof.

Claim B.1. *Let A, B be two R-facts in* **db**. *For every $K \subseteq \{1, \ldots, n\}$, the following are equivalent:*

(1) A and B agree on every position in K; and
(2) $f(A)$ and $f(B)$ agree on every position in K.

From Claim B.1, it follows in particular that f is injective, i.e., f maps distinct R-facts to distinct R-facts. Furthermore, let f be the identity on facts whose relation name is not R. Finally, we define $f(\mathbf{db}) \coloneqq \{f(A) \mid A \in \mathbf{db}\}$. By Claim B.1,

$$\text{rset}(f(\mathbf{db})) = \{f(\mathbf{r}) \mid \mathbf{r} \in \text{rset}(\mathbf{db})\}. \tag{15}$$

Clearly, the function f is first-order computable. The following equivalence holds by construction, for every typed database instance $\mathbf{s}$ that is a legal input to CERTAINTY$((q \setminus \{F\}) \cup p)$:

$$\begin{array}{c} f(\mathbf{s}) \text{ contains } R(a_1, \ldots, a_n) \text{ with } a_i = a_j \\ \Updownarrow \\ \text{there exists } c \text{ such that } \mathbf{s} \text{ contains both} \\ R(a_1, \ldots, a_{j-1}, c, a_j, \ldots, a_n) \text{ and } S^{\mathsf{c}}(a_i, c). \end{array} \tag{16}$$

It is now straightforward to show, using (15) and (16), that $\mathbf{db}$ is a "yes"-instance of the problem CERTAINTY$((q \setminus \{F\}) \cup p)$ if and only if $f(\mathbf{db})$ is a "yes"-instance of CERTAINTY(q). □

C PROOF OF LEMMA 4.2

Proof of Lemma 4.2. There is an obvious first-order reduction from CERTAINTY(q) to CERTAINTY$((q \setminus \{F\}) \cup p)$: $\mathbf{db}$ is a "yes"-instance of the problem CERTAINTY(q) if and only if $\mathbf{db} \cup \{C^{\mathsf{c}}(t_j)\}$ is a "yes"-instance of the problem CERTAINTY$((q \setminus \{F\}) \cup p)$. In the remainder of the proof, we establish a first-order reduction from CERTAINTY$((q \setminus \{F\}) \cup p)$ to CERTAINTY(q).

For every constant c, let q_c be the query obtained from q by replacing the atom $R(t_1, \ldots, t_n)$ with $R(t_1, \ldots, t_{i-1}, c, t_{i+1}, \ldots, t_n)$. In particular, $q = q_{t_i}$.

Let $\mathbf{db}$ be a legal instance (either a "yes"-instance or a "no"-instance) of CERTAINTY$((q \setminus \{F\}) \cup p)$. Obviously, $\mathbf{db}$ is a "yes"-instance if and only if the following two conditions are satisfied:

(A) for some constant c, we have that $C^{\mathsf{c}}(c)$ is a (necessarily unique) C-fact of $\mathbf{db}$; and
(B) $\mathbf{db}$ is a "yes"-instance of CERTAINTY(q_c).

Our first-order reduction first checks whether $\mathbf{db}$ satisfies (A), i.e., whether $\mathbf{db} \models \exists x\,(C^{\mathsf{c}}(x))$. If $\mathbf{db} \not\models \exists x\,(C^{\mathsf{c}}(x))$, then $\mathbf{db}$ is a "no"-instance. In the remainder, assume $\mathbf{db} \models \exists x\,(C^{\mathsf{c}}(x))$. We can assume a constant c such that $\mathbf{db}$ contains $C^{\mathsf{c}}(c)$. The reduction consists in a renaming h of constants that occur at the jth position of some R-fact. This renaming is needed because we have to reduce to CERTAINTY(q_{t_i}), rather than to CERTAINTY(q_c). We can use the renaming $h = \{(t_j, c), (c, t_j)\}$ for this purpose: for every fact $R(a_1, \ldots, a_n) \in \mathbf{db}$ such that $a_j \in \{c, t_j\}$, replace the fact $R(a_1, \ldots, a_n)$ with the fact $R(a_1, \ldots, a_{j-1}, h(a_j), a_{j+1}, \ldots, a_n)$. Call the result $\mathbf{db}'$. Clearly, $\mathbf{db}'$ can be computed from $\mathbf{db}$ in **FO**. It is obvious that $\mathbf{db}$ is a "yes"-instance of CERTAINTY$((q \setminus \{F\}) \cup p)$ if and only if $\mathbf{db}'$ is a "yes"-instance of CERTAINTY(q). □

D PROOF OF LEMMA 5.5

We first show that every cyclic attack graph contains a cycle of size two.

Lemma D.1. *Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. Let $[\![R : K]\!]$ and $[\![S : L]\!]$ be keys of q such that $R \neq S$. Let $v \in \mathsf{vars}(q)$. If $[\![R : K]\!] \overset{q}{\rightsquigarrow} [\![S : L]\!]$ and $[\![S : L]\!] \overset{q}{\rightsquigarrow} v$, then either $[\![R : K]\!] \overset{q}{\rightsquigarrow} v$ or $[\![S : L]\!] \overset{q}{\rightsquigarrow} [\![R : K]\!]$ (or both).*

Proof. Assume $[\![R : K]\!] \overset{q}{\rightsquigarrow} [\![S : L]\!] \overset{q}{\rightsquigarrow} v$. There is a sequence

$$x_1, x_2, \ldots, x_{\ell-1}, x_\ell, y_1, y_2, \ldots, y_m$$

such that

- $x_1 \in \mathsf{vars}(R)$, $x_\ell \in L$, $y_1 \in \mathsf{vars}(S)$, $y_m = v$;
- no x_i belongs to $[R : K]^{\ominus, q}$;
- no y_i belongs to $[S : L]^{\ominus, q}$; and
- every two adjacent variables occur together in some atom of q. We have, in particular, $\{x_\ell, y_1\} \subseteq \mathsf{vars}(S)$.

The desired result holds obviously true if no y_i belongs to $[R : K]^{\ominus, q}$. In the remainder, we assume that some y_i belongs to $[R : K]^{\ominus, q}$. Let k be the smallest index such that $y_k \in [R : K]^{\ominus, q}$. The desired result is obvious if $y_k \in K$ (because $[\![S : L]\!] \overset{q}{\rightsquigarrow} y_k$). Assume $y_k \notin K$ from here on. There exists a sequence of keys of q

$$[\![T_1 : M_1]\!], [\![T_2 : M_2]\!], \ldots, [\![T_n : M_n]\!] \tag{17}$$

such that $y_k \in \mathsf{vars}(T_n) \setminus M_n$ and for every $i \in \{1, \ldots, n\}$,

- $T_i \neq R$; and
- $M_i \subseteq K \cup \left(\bigcup_{j=1}^{i-1} \mathsf{vars}(T_j)\right)$.

For every variable $u \in \bigcup_{i=1}^{n} \mathsf{vars}(T_i)$, we have $u \in [R : K]^{\ominus, q}$ and therefore $[\![R : K]\!] \overset{q}{\not\rightsquigarrow} u$. Since $[\![R : K]\!] \overset{q}{\rightsquigarrow} x_\ell$ with $x_\ell \in \mathsf{vars}(S)$, it follows that for every $i \in \{1, \ldots, n\}$, $T_i \neq S$.

Define $[\![T_0 : M_0]\!] := [\![R : K]\!]$. We claim:

$$\begin{array}{l} \text{for every } i \geq 1, \text{ if } [\![S : L]\!] \text{ attacks some variable} \\ \text{of } M_i, \text{ then there exists } j < i \text{ such that } [\![S : L]\!] \\ \text{attacks some variable of } M_j. \end{array} \tag{18}$$

To prove this claim, assume that $[\![S : L]\!] \overset{q}{\rightsquigarrow} x$ with $x \in M_i$ and $i \geq 1$. The desired result is obvious if $x \in M_0 = K$. We now treat the case that $x \notin M_0$. By the construction of (17), there exists $j < i$ such that $x \in \mathsf{vars}(T_j) \setminus M_j$. By Lemma 5.4, $[\![S : L]\!]$ attacks some variable in M_j.

From $[\![S : L]\!] \overset{q}{\rightsquigarrow} y_k$ and Lemma 5.4, it follows that $[\![S : L]\!]$ attacks some variable in M_n. The desired result follows by repeated application of (18). □

Corollary D.2. *Let $q \in$ sjfBCQ be a query in which no atom contains constants or repeated variables. If the attack graph of q has a cycle, then it has a cycle of length two.*

Proof. Assume that the attack graph of q has a cycle. We can assume that $[\![R_0 : K_0]\!], \ldots, [\![R_{\ell-1} : K_{\ell-1}]\!]$ is a sequence of keys of q such that for all $i, j \in \{0, \ldots, \ell - 1\}$,

- $i \neq j$ implies $R_i \neq R_j$, and
- $[\![R_i : K_i]\!] \overset{q}{\rightsquigarrow} [\![R_{i+1 \mod \ell} : K_{i+1 \mod \ell}]\!]$.

The proof runs by induction on ℓ, the length of the cycle. The desired result is obvious if $\ell = 2$.

Assume $\ell \geq 3$. We have $[\![R_0 : K_0]\!] \overset{q}{\rightsquigarrow} [\![R_1 : K_1]\!] \overset{q}{\rightsquigarrow} [\![R_2 : K_2]\!]$. The desired result obtains vacuously if $[\![R_1 : K_1]\!] \overset{q}{\rightsquigarrow} [\![R_0 : K_0]\!]$. Assume $[\![R_1 : K_1]\!] \overset{q}{\not\rightsquigarrow} [\![R_0 : K_0]\!]$ from here on. Since $[\![R_1 : K_1]\!] \overset{q}{\rightsquigarrow} [\![R_2 : K_2]\!]$, there exists a variable $v \in K_2$ such that $[\![R_1 : K_1]\!] \overset{q}{\rightsquigarrow} v$. By Lemma D.1, we have $[\![R_0 : K_0]\!] \overset{q}{\rightsquigarrow} v$, and therefore $[\![R_0 : K_0]\!] \overset{q}{\rightsquigarrow} [\![R_2 : K_2]\!]$. Then the attack graph has a cycle of length $\ell - 1$, and the desired result follows from the induction hypothesis. □

Proof of Lemma 5.5. Assume that the attack graph of q has a cycle. By Corollary D.2, the attack graph of q has a cycle of length two. We can assume that $[\![R : K]\!]$ and $[\![S : L]\!]$ with $R \neq S$ are keys of

q that mutually attack one another. The proof is a first-order reduction from CERTAINTY(q_0) with $q_0 = \{R_0(\underline{x}, y), S_0(\underline{y}, x)\}$, which is an **L**-hard problem [17].

For every pair (a, b) of constants, define Θ^a_b as the valuation over $\mathrm{vars}(q)$ such that for every $x \in \mathrm{vars}(q)$,

$$\Theta^a_b(x) = \begin{cases} a & \text{if } x \in [R:K]^{\ominus,q} \setminus [S:L]^{\ominus,q} \\ b & \text{if } x \in [S:L]^{\ominus,q} \setminus [R:K]^{\ominus,q} \\ \bot & \text{if } x \in [R:K]^{\ominus,q} \cap [S:L]^{\ominus,q} \\ \langle a, b\rangle & \text{otherwise} \end{cases}$$

Claim D.1. *$\{\Theta^a_b(R), \Theta^{a'}_{b'}(R)\}$ is inconsistent if and only if $a = a'$ and $b \neq b'$.*

Proof of Claim D.1. $\boxed{\Longrightarrow}$ By contraposition, it suffices to show the following:

(A) if $a \neq a'$, then $\{\Theta^a_b(R), \Theta^{a'}_{b'}(R)\}$ is consistent; and

(B) if $b = b'$, then $\{\Theta^a_b(R), \Theta^{a'}_{b'}(R)\}$ is consistent.

For (A), assume $a \neq a'$. We have that $K \neq \emptyset$, or else $[\![R:K]\!]$ would have no incoming attack, a contradiction. Since $K \subseteq [R:K]^{\ominus,q}$, it is obvious that $\{\Theta^a_b(R), \Theta^{a'}_{b'}(R)\} \models [\![R:K]\!]$.

Let $[\![R:M]\!]$ be another key of q. By our definition of Θ^a_b, it suffices to show $M \notin [S:L]^{\ominus,q}$. Assume, toward a contradiction, $M \subseteq [S:L]^{\ominus,q}$, that is, $\mathcal{K}(q \ominus S) \models L \to M$. Since $\mathcal{K}(q \ominus S) \models M \to K$ is obvious, we have $\mathcal{K}(q \ominus S) \models L \to K$. But then $[\![S:L]\!]$ cannot attack $[\![R:K]\!]$, a contradiction.

For (B), assume $b = b'$. If $a \neq a'$, then the desired result follows from (A). If $a = a'$, then $\Theta^a_b(R) = \Theta^{a'}_{b'}(R)$, and the desired result follows vacuously (because a database instance that is a singleton satisfies every KD).

$\boxed{\Longleftarrow}$ Assume $a = a'$ and $b \neq b'$. From $K \subseteq [R:K]^{\ominus,q}$, it follows that Θ^a_b and $\Theta^{a'}_{b'}$ agree on K. Since $[\![R:K]\!]$ has an outgoing attack, $\mathrm{vars}(R) \notin [R:K]^{\ominus,q}$, and the desired result obtains because Θ^a_b and $\Theta^{a'}_{b'}$ disagree on variables in $\mathrm{vars}(R) \setminus [R:K]^{\ominus,q}$. ∎

Claim D.2. *If $\Theta^a_b(R) = \Theta^{a'}_{b'}(R)$, then $a = a'$ and $b = b'$.*

Proof of Claim D.2. Assume $\Theta^a_b(R) = \Theta^{a'}_{b'}(R)$. As argued in the proof of Claim D.1, $\emptyset \neq K \subseteq [R:K]^{\ominus,q}$. It obviously follows $a = a'$.

Since $\{\Theta^a_b(R), \Theta^{a'}_{b'}(R)\}$ is consistent, Claim D.1 implies $a \neq a'$ or $b = b'$. Since $a = a'$, if follows $b = b'$. ∎

Claim D.3. *For every atom $T \in q \setminus \{R, S\}$, for all constants a, a', b, b', $\{\Theta^a_b(T), \Theta^{a'}_{b'}(T)\}$ is consistent.*

Proof of Claim D.3. Let $[\![T:M]\!]$ be a key of q. Assume that Θ^a_b and $\Theta^{a'}_{b'}$ agree on M. We need to show $\Theta^a_b(T) = \Theta^{a'}_{b'}(T)$. We have that both $\mathcal{K}(q \ominus R)$ and $\mathcal{K}(q \ominus S)$ contain $M \to \mathrm{vars}(T)$. The desired result is obvious if $a = a'$ and $b = b'$. Three other cases can occur:

- Case $a = a'$ and $b \neq b'$. Then $M \subseteq [R:K]^{\ominus,q}$, hence $\mathcal{K}(q \ominus R) \models K \to M$. It follows $\mathcal{K}(q \ominus R) \models K \to \mathrm{vars}(T)$, and therefore $\mathrm{vars}(T) \subseteq [R:K]^{\ominus,q}$. The desired result follows from the definition of Θ^a_b.
- Case $a \neq a'$ and $b = b'$. This case is symmetrical to the previous item.
- Case $a \neq a'$ and $b \neq b'$. Then $M \subseteq [R:K]^{\ominus,q} \cap [S:L]^{\ominus,q}$, hence $\mathcal{K}(q \ominus R) \models K \to M$ and $\mathcal{K}(q \ominus S) \models L \to M$. It follows $\mathcal{K}(q \ominus R) \models K \to \mathrm{vars}(T)$ and $\mathcal{K}(q \ominus S) \models L \to \mathrm{vars}(T)$, and therefore $\mathrm{vars}(T) \subseteq [R:K]^{\ominus,q} \cap [S:L]^{\ominus,q}$. The desired result follows from the definition of Θ^a_b.

This concludes the proof of Claim D.3. ∎

For every instance **db** of CERTAINTY(q_0), we define $f(\mathbf{db})$ as the following database instance:

- for every $R_0(\underline{a}, b) \in \mathbf{db}$ such that $S_0(\underline{b}, a) \notin \mathbf{db}$, $f(\mathbf{db})$ includes $\Theta^a_b(q \setminus \{S\})$;
- for every $R_0(\underline{a}, b) \in \mathbf{db}$ such that $S_0(\underline{b}, a) \in \mathbf{db}$, $f(\mathbf{db})$ includes $\Theta^a_b(q)$; and
- for every $S_0(\underline{b}, a) \in \mathbf{db}$ such that $R_0(\underline{a}, b) \notin \mathbf{db}$, $f(\mathbf{db})$ includes $\Theta^a_b(q \setminus \{R\})$.

Let $F(\mathbf{db})$ be the subset of $f(\mathbf{db})$ that contains all facts that are neither R-facts nor S-facts. By Claims D.1, D.2, and D.3 (and their symmetric versions),

$$\mathrm{rset}(f(\mathbf{db})) = \{f(\mathbf{r}) \cup F(\mathbf{db}) \mid \mathbf{r} \in \mathrm{rset}(\mathbf{db})\}.$$

Let **db** be an instance of CERTAINTY(q_0). It suffices to show that for every repair **r** of **db**,

$$\mathbf{r} \models q_0 \text{ if and only if } f(\mathbf{r}) \cup F(\mathbf{db}) \models q.$$

To show the latter equivalence, let **r** be an arbitrary repair of **db**.

$\boxed{\Longrightarrow}$ Assume $\mathbf{r} \models q_0$. There exist constants a, b such that $R_0(\underline{a}, b), S_0(\underline{b}, a) \in \mathbf{r}$. It follows that $f(\mathbf{r})$ includes $\Theta^a_b(q)$, hence $f(\mathbf{r}) \cup F(\mathbf{db}) \models q$.

$\boxed{\Longleftarrow}$ Assume $f(\mathbf{r}) \cup F(\mathbf{db}) \models q$. There is a valuation μ such that $\mu(q) \subseteq f(\mathbf{r}) \cup F(\mathbf{db})$. Then, there are $R_0(\underline{a}, b), S_0(\underline{b'}, a') \in \mathbf{r}$ such that $\mu(R) = \Theta^a_b(R)$ and $\mu(S) = \Theta^{a'}_{b'}(S)$. It suffices to show $a = a'$ and $b = b'$. We show $a = a'$; the proof of $b = b'$ is symmetrical. Since $[\![R:K]\!] \overset{q}{\rightsquigarrow} [\![S:L]\!]$, there is a sequence

$$F_0, v_1, F_1, v_2, F_2 \ldots, F_i, v_{i+1}, F_{i+1}, \ldots, v_\ell, F_\ell$$

where $\{F_i\}_{i=0}^{\ell} \subseteq q$, $F_0 = R$, $F_\ell = S$, and for every $i \in \{1, \ldots, \ell\}$, $v_i \in \mathrm{vars}(q) \setminus [R:K]^{\ominus,q}$ such that v_i occurs in both F_{i-1} and F_i. For every $i \in \{0, \ldots, \ell\}$, there are constants a_i, b_i such that $\mu(F_i) = \Theta^{a_i}_{b_i}(F_i)$. We show by induction on increasing i that for every $i \in \{0, \ldots, \ell\}$, $b_i = b$.

Basis $i = 0$. Since $\Theta^a_b(R) = \mu(R) = \Theta^{a_0}_{b_0}(R)$, it follows by Claim D.2 that $b_0 = b$.

Step $i \to i+1$. The induction hypothesis is $b_i = b$. We have that μ, $\Theta^{a_i}_{b_i}$, and $\Theta^{a_{i+1}}_{b_{i+1}}$ agree on v_{i+1}. Then, from $\Theta^{a_i}_{b_i}(v_{i+1}) \in \{b, \langle a_i, b\rangle\}$ and $\Theta^{a_{i+1}}_{b_{i+1}}(v_{i+1}) \in \{b_{i+1}, \langle a_{i+1}, b_{i+1}\rangle\}$, it follows $b = b_{i+1}$.

Consequently, $b_\ell = b$. Since $\Theta^{a'}_{b'}(S) = \mu(S) = \Theta^{a_\ell}_{b}(S)$, it follows from (the symmetrical of) Claim D.2 that $b = b'$. This concludes the proof of Lemma 5.5. □