A Dichotomy in Consistent Query Answering for Primary Keys and Unary Foreign Keys

Miika Hannula
miika.hannula@helsinki.fi
University of Helsinki
Helsinki, Finland

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

ABSTRACT

Since 2005, significant progress has been made in the problem of Consistent Query Answering (CQA) with respect to primary keys. In this problem, the input is a database instance that may violate one or more primary key constraints. A repair is defined as a maximal subinstance that satisfies all primary keys. Given a Boolean query q, the question then is whether q holds true in every repair.

So far, theoretical research in this field has not addressed the combination of primary key and foreign key constraints, despite the importance of referential integrity in database systems. This paper addresses the problem of CQA with respect to both primary keys and foreign keys. In this setting, it is natural to adopt the notion of symmetric-difference repairs, because foreign keys can be repaired by inserting new tuples.

We consider the case where foreign keys are unary, and queries are conjunctive queries without self-joins. In this setting, we characterize the boundary between those CQA problems that admit a consistent first-order rewriting, and those that do not.

CCS CONCEPTS

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

KEYWORDS

consistent query answering; primary key; foreign key; conjunctive query

ACM Reference Format:
Miika Hannula and Jef Wijsen. 2022. A Dichotomy in Consistent Query Answering for Primary Keys and Unary Foreign Keys. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3517804.3524157

1 INTRODUCTION

Consistent query answering (CQA) was introduced in [1] as a principled semantics for answering queries on inconsistent databases. A *symmetric-difference repair* (or ⊕-repair) of a database **db** is defined as a consistent database **r** that ⊆-minimizes the symmetric difference with **db**. Informally, a ⊕-repair **r** becomes inconsistent as soon as we insert into it more tuples of **db**, or delete from it tuples not in **db**. Then, given a query $q(\vec{x})$, an answer $\vec{a}$ is called *consistent* if $q(\vec{a})$ holds true in every repair. The problem is often studied for Boolean queries q, where the question is to determine whether q holds true on every repair of a given input database.

CQA has been studied in depth in case that the only constraints are primary keys, one per relation. In [30], this problem was coined as CERTAINTY(q), in which notation it is understood that every relation name in q has a predefined primary key. More than a decade of research has eventually resulted in the following complexity classification [25]: for every self-join-free Boolean conjunctive query q, the problem CERTAINTY(q) is either in FO, L-complete, or coNP-complete.

Now that this classification has been settled, it is natural to ask what happens if we add foreign key constraints. Indeed, every relational database textbook is likely to introduce very soon the notion of referential integrity, i.e., foreign keys referencing primary keys. In view thereof, one may even wonder why referential integrity in CQA has so far received little theoretical research attention. One plausible explanation is that ⊕-repairs with respect to primary keys are easy to characterize: every repair has to delete, in every *block*, all tuples but one, where a block is a maximal set of tuples of the same relation that agree on their primary key. In contrast, ⊕-repairs with respect to foreign keys can introduce new tuples, as illustrated next. It will become apparent in later sections that having, as repair primitives, both tuple insertions and tuple deletions considerably complicates the theoretical treatment of CQA.

Consider the database of Fig. 1, in which primary keys are underlined. A tuple (d, o) in the relation R means that the document with DOI d was written by the author with ORCiD o. The set of foreign keys is $\mathcal{FK}_0 := \{\mathsf{R}[1] \to \mathsf{DOCS}, \mathsf{R}[2] \to \mathsf{AUTHORS}\}$. In this paper, we assume that every foreign key is unary (i.e., consists of a single attribute) and that the referenced primary key is the leftmost attribute in the referenced table.

R

doi	orcid
d1	o1
d1	o2
d1	o3

AUTHORS

orcid	first	last
o1	*Jeff*	*Ullman*
o1	*Jeffrey*	*Ullman*
o2	*Jonathan*	*Ullman*

DOCS

doi	title	year
d1	*Some pairs problems*	2016

Figure 1: Inconsistent database.

PODS '22, June 12–17, 2022, Philadelphia, PA, USA.

ACM ISBN 978-1-4503-9260-0/22/06…$15.00
https://doi.org/10.1145/3517804.3524157

There is one foreign-key violation: the fact R(d1,o3) is *dangling*, because o3 is not an existing ORCiD in the table AUTHORS. There is also one primary-key violation, because there are two distinct tuples with ORCiD o1 in the table AUTHORS. This database has an infinite number of $\oplus$-repairs. To repair the primary-key violation, we must delete either tuple with ORCiD o1 in the table AUTHORS. To repair the foreign-key violation, we can either delete the fact R(d1,o3), or insert a new fact AUTHORS(o3, *fi*, *la*), where *fi* and *la* can be chosen arbitrarily. Consider now the Boolean query:

Does some paper of *2016* have an author with first name *Jeff*?

There is a repair in which the answer to this query is "no," in which case we also say that "no" is the consistent answer. In our setting, this Boolean query will be denoted by the following set of atoms:

$$q_0 = \{\mathsf{DOCS}(\underline{x}, t, \text{'2016'}), \mathsf{R}(\underline{x, y}), \mathsf{AUTHORS}(\underline{y}, \text{'Jeff'}, z)\}.$$

We note that q_0 satisfies the foreign keys in $\mathcal{FK}_0$ (when distinct variables are treated as distinct constants) and every relation name that occurs in $\mathcal{FK}_0$ also occurs in q_0, in which case we say that $\mathcal{FK}_0$ *is about* q_0.

Data cleaning [14, 16] differs from CQA in that it tries to single out the single best repair before asking any query. We view CQA as complementary to data cleaning. In the preceding example, it may take some time (and manual effort) to find out what is the correct first name of the author with ORCiD o1, and how the dangling fact in R has to be cleaned. An advantage of CQA is that we can immediately obtain some consistent query answers, which will hold true no matter of which repair will be chosen during the data cleaning process.

For every self-join-free Boolean conjunctive query q, for every set of foreign keys that is about q, we define CERTAINTY$(q, \mathcal{FK})$ as the following problem:

> Problem CERTAINTY$(q, \mathcal{FK})$.
> **Input:** A database instance **db**.
> **Question:** Is q true in every $\oplus$-repair w.r.t. foreign keys in $\mathcal{FK}$ and primary keys?

Obviously, if $\mathcal{FK} = \emptyset$, then CERTAINTY$(q, \mathcal{FK})$ becomes the well studied problem CERTAINTY(q).

Of special interest is the case where CERTAINTY$(q, \mathcal{FK})$ is in the complexity class FO, which is the class of problems that take a relational database instance as input and can be solved by a relational calculus query (a.k.a. *consistent first-order rewriting* in the context of CQA). A major contribution of this paper can now be stated.

Theorem 1.1. *For every self-join-free Boolean conjunctive query q, for every set of unary foreign keys $\mathcal{FK}$ that is about q, it can be decided whether or not* CERTAINTY$(q, \mathcal{FK})$ *is in* FO. *Furthermore, if* CERTAINTY$(q, \mathcal{FK})$ *is in* FO*, its consistent first-order rewriting can be effectively constructed.*

We briefly discuss the remaining restrictions, leaving a more detailed discussion to Section 9. The requirement that foreign keys be unary (i.e., consist of a single attribute) is met in our example, and is likely to be met in many real life situations where entities are identified by unique identifiers (like DOI, ORCiD…). Note that we allow composite primary keys, as in the relation R in our example, but such composite primary keys cannot be referenced by a foreign key. Nevertheless, some results in this paper are already proved for foreign keys that need not be unary.

The restriction that the set of foreign keys must be *about the query* needs some care during query writing. For example, the question whether the author with ORCiD o1 has published some paper in 2016, should be formulated as follows:

$$q_1 = \{\mathsf{DOCS}(\underline{x}, t, \text{'2016'}), \mathsf{R}(\underline{x, \text{'o1'}}), \mathsf{AUTHORS}(\underline{\text{'o1'}}, u, z)\}.$$

The third atom may look redundant in the latter query. However, $\mathcal{FK}_0$ is not about the shorter query $\{\mathsf{DOCS}(\underline{x}, t, \text{'2016'}), \mathsf{R}(\underline{x, \text{'o1'}})\}$, in which $\mathsf{R}(\underline{x, \text{'o1'}})$ is dangling with respect to $R[2] \to \mathsf{AUTHORS}$.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 introduces preliminary notions and results from the literature that are used in our work. In Section 4, we define a novel notion, called *block-interference*, which plays a central role in a main theorem, given in Section 5, which implies Theorem 1.1. Sections 6 and 7 show that CERTAINTY$(q, \mathcal{FK})$ is L-hard or NL-hard (and thus not in FO) under some conditions. Section 8 shows that if these conditions are not satisfied, then CERTAINTY$(q, \mathcal{FK})$ is in FO. In this way, our main theorem will be proved. A side result in Section 7 is the existence of NL-complete and P-complete cases of CERTAINTY$(q, \mathcal{FK})$, which complexity classes did not pop up in earlier studies that were restricted to primary keys. We conclude this paper with a discussion in Section 9. Some proofs and several helping lemmas are given in the appendix. More details can be found in the full version of this paper [15].

2 RELATED WORK

Consistent query answering (CQA) was initiated in a seminal paper by Arenas, Bertossi, and Chomicki [1], in which the notions of $\oplus$-repairs and consistent query answers were introduced. Recent overviews of two decades of research in CQA are [2, 31]. From the latter overview, it is clear that different classes of constraints have been studied independently in CQA. The current study is different in that it combines constraints from two classes: primary keys belong to the larger class of equality-generating dependencies (egd), and foreign keys belong to the larger class of tuple-generating dependencies (tgd). CQA has also been studied in the context of ontologies formulated in description logics; see [3] for a recent overview.

The term CERTAINTY(q) was coined in 2010 [30] to refer to CQA for Boolean queries q on databases that violate primary keys, one per relation, which are fixed by q's schema. A systematic study of its complexity for self-join-free conjunctive queries had started already in 2005 [13], and was eventually solved in two journal articles by Koutris and Wijsen [22, 25], as follows: for every self-join-free Boolean conjunctive query, CERTAINTY(q) is either in FO, L-complete, or coNP-complete, and it is decidable, given q, which case applies.

A few extensions beyond this trichotomy result are known. The complexity of CERTAINTY(q) for self-join-free conjunctive queries with negated atoms was studied in [23]. For self-join-free conjunctive queries with respect to multiple keys, it remains decidable whether or not CERTAINTY(q) is in FO [24]. The complexity landscape of CERTAINTY(q) for path queries, a subclass of (not necessarily self-join-free) conjunctive queries, was settled in [21]. For

unions of conjunctive queries q, Fontaine [10] established interesting relationships between CERTAINTY(q) and Bulatov's dichotomy theorem for conservative CSP [4].

The counting variant of CERTAINTY(q), denoted ♯CERTAINTY(q), asks to count the number of repairs that satisfy some Boolean query q. For self-join-free conjunctive queries, ♯CERTAINTY(q) exhibits a dichotomy between FP and ♯P-complete under polynomial-time Turing reductions [27]. This dichotomy also holds for queries with self-joins if primary keys are singletons [28]. Calautti, Console, and Pieris present in [5] a complexity analysis of these counting problems under weaker reductions, in particular, under many-one logspace reductions. The same authors have conducted an experimental evaluation of randomized approximation schemes for approximating the percentage of repairs that satisfy a given query [6]. Other approaches to making CQA more meaningful and/or tractable include operational repairs [7] and preferred repairs [19, 29].

It is worthwhile to note that theoretical research in CERTAINTY(q) has stimulated implementations and experiments in prototype systems [8, 11, 12, 18, 20].

3 PRELIMINARIES

For a positive integer n, we write $[n]$ for the set $\{1, \ldots, n\}$. We assume denumerable sets of *variables* and *constants*. A *term* is a variable or a constant. Every *relation name* is associated with a signature, which is a pair $[n, k]$ of positive integers, where n is the *arity* and $k \in [n]$; the set $[k]$ is the *primary key* of R, and each $i \in [k]$ is called a *primary-key position*.

From here on, we assume a fixed *database schema* (i.e., a finite set of relation names with their associated signatures).

3.1 CQA for Primary Keys

We summarize notations and results used in CQA for primary keys. The following definitions are borrowed and adapted from [22].

If R is a relation name with signature $[n, k]$, and $t_1, \ldots, t_n$ are terms, then $R(\underline{t_1, \ldots, t_k}, t_{k+1}, \ldots, t_n)$ is an *R-atom* (or simply *atom*). If F is an atom, then $\mathsf{vars}(F)$ denotes the set of variables that occur in F, and $\mathsf{key}(F)$ denotes the set of variables that occur in F at some primary-key position. An atom without variables is called a *fact*. Two facts A, B are said to be *key-equal*, denoted $A \sim B$, if they use the same relation name and agree on all primary-key positions.

A *database (instance)* is a finite set **db** of facts. From here on, **db** stands for a database instance. A *Boolean conjunctive query* is a finite set q of atoms. We write $\mathsf{vars}(q)$ for the set of variables that occur in q, and $\mathsf{const}(q)$ for the set of constants that occur in q. If $x \in \mathsf{vars}(q)$ and c is a constant, then $q_{[x \to c]}$ is the query obtained from q by replacing each occurrence of x with c; this notation naturally extends to sequences with more than one variable and constant. A Boolean conjunctive query is *self-join-free* if it does not contain two atoms with the same relation name. We write sjfBCQ for the class of all self-join-free Boolean conjunctive queries.

In contexts where a query q in sjfBCQ is understood, whenever we use a relation name R where an atom is expected, we mean the (unique) R-atom of q.

A *valuation* over a set V of variables is a total mapping θ from V to the set of constants. A valuation is extended to map every constant to itself. A valuation naturally extends to atoms and sets of atoms. A Boolean conjunctive query q is satisfied by **db**, denoted $\mathbf{db} \models q$, if there is a valuation over $\mathsf{vars}(q)$ such that $\theta(q) \subseteq \mathbf{db}$.

A *block* of **db** is a maximal subset of key-equal facts. If A is a fact in **db**, then $\mathsf{block}(A, \mathbf{db})$ denotes the block of **db** that contains A. If $A = R(\vec{a}, \vec{b})$, then $\mathsf{block}(A, \mathbf{db})$ is also denoted by $R(\vec{a}, *)$, and a fact in this block is said to be *of the form* $R(\vec{a}, \_)$.

A *repair of* **db** *with respect to primary keys* is a maximal subset of **db** containing no two distinct key-equal facts. If q is a Boolean conjunctive query, then CERTAINTY(q) is the problem that, given an input database instance **db**, asks whether q is satisfied by every repair of **db** with respect to primary keys.

Instead of saying that a repair must not contain two distinct key-equal facts, we can say that, for every relation name R of signature $[n, k]$, a repair must satisfy the following *primary-key constraint*:

$$\forall x_1 \cdots \forall x_n \forall y_{k+1} \cdots \forall y_n \left(\left(\begin{array}{l} R(\underline{x_1, \ldots, x_k}, x_{k+1}, \ldots, x_n) \wedge \\ R(\underline{x_1, \ldots, x_k}, y_{k+1}, \ldots, y_n) \end{array} \right) \rightarrow \left(\bigwedge_{i=k+1}^{n} x_i = y_i \right) \right). \quad (1)$$

In the technical treatment, it will often be convenient to use $\mathcal{PK}$ for the set that contains such a formula for every relation name in the database schema under consideration.

The complexity classification of CERTAINTY(q) uses the notion of *attack graph* [22] recalled next. For a query q in sjfBCQ, we write $\mathcal{K}(q)$ for the set $\{\mathsf{key}(F) \to \mathsf{vars}(F) \mid F \in q\}$, which is a set of functional dependencies over $\mathsf{vars}(q)$. For an atom $F \in q$, we define $F^{+,q} := \{x \in \mathsf{vars}(q) \mid \mathcal{K}(q \setminus \{F\}) \models \mathsf{key}(F) \to x\}$. Informally, $F^{+,q}$ is the set of variables that are functionally dependent on $\mathsf{key}(F)$ via the functional dependencies in $\mathcal{K}(q \setminus \{F\})$. The *attack graph of q* is a directed graph whose vertices are the atoms of q; there is a directed edge from F to G, called an *attack* and denoted $F \overset{q}{\rightsquigarrow} G$, if $F \neq G$ and there exists a sequence of variables $x_0, x_1, \ldots, x_n$, all belonging to $\mathsf{vars}(q) \setminus F^{+,q}$, such that $x_0 \in \mathsf{vars}(F)$, $x_n \in \mathsf{vars}(G)$, and every two adjacent variables occur together in some atom of q. Moreover, F is said to attack every variable in such a sequence. The following result obtains.

Theorem 3.1 ([22]). *Let q be a query in* sjfBCQ. *If the attack graph of q is acyclic, then the problem* CERTAINTY(q) *is in* FO*; otherwise* CERTAINTY(q) *is* L*-hard.*

FO is used for the class of decision problems that take a database instance as input, and that can be solved by a closed first-order formula.

3.2 Foreign keys

Let R be a relation name with signature $[n, k]$, and S a relation name with signature $[m, 1]$. Possibly $R = S$. A *foreign key* is an expression $R[i] \to S$ such that $1 \leq i \leq n$. It is called *weak* if $i \leq k$, and *strong* otherwise. We say that this foreign key is *outgoing from R* and *referencing S*. We say that a fact $R(a_1, \ldots, a_n)$ of **db** is *dangling (in* **db***)* with respect to this foreign key if **db** contains no S-fact $S(\underline{b_1}, b_2, \ldots, b_n)$ such that $a_i = b_1$. A fact of **db** is dangling with respect to a set of foreign keys if it is dangling with respect to some foreign key of the set. A set of foreign keys is *satisfied* by **db** if **db** contains no dangling facts. Remark that foreign keys are unary by definition.

We write $\mathcal{FK}^*$ for the set that contains every foreign key that is logically implied by $\mathcal{FK}$ (and that only uses relation names of the database schema under consideration), where logical implication has its standard definition.

The following notion of *dependency graph* is borrowed and adapted from [9, Def. 3.7], where it was defined for general tgds. The *dependency graph of a set* $\mathcal{FK}$ of foreign keys is a directed graph. There is a vertex (R, i) whenever R is a relation name that occurs in $\mathcal{FK}$, say with signature $[n, k]$, and $i \in [n]$. Such a pair (R, i) will be called a *position*. More specifically, we say that (R, i) is a *primary-key position* if $i \in [k]$, and otherwise a *non-primary-key position*. Each foreign key $R[i] \to S$ in $\mathcal{FK}$, where S has signature $[m, 1]$, induces a directed edge from (R, i) to (S, j), for every $j \in [m]$. An edge from (R, i) to (S, j) is called *special* if $j \neq 1$. For a set of positions P, we define the *closure* $P_{\mathcal{FK}}$ *of* P *under* $\mathcal{FK}$ as the set of all positions (R, i) such that there is a path (possibly of length 0) from some position in P to (R, i) in the dependency graph of $\mathcal{FK}$. The *complement of* $P_{\mathcal{FK}}$ (with respect to all positions of the database schema under consideration), denoted $P^{\text{co}}_{\mathcal{FK}}$, is the set of positions $(R, i) \notin P_{\mathcal{FK}}$. Note that if a relation name R of arity n occurs in a query but not in $\mathcal{FK}$, then $P^{\text{co}}_{\mathcal{FK}}$ includes $\{(R, i) | i \in [n]\}$, even though the positions in the latter set are not vertices of the dependency graph.

Example 3.2. Let $\mathcal{FK} = \{R[1] \to S, R[3] \to T\}$, where R has signature $[3, 2]$, and S and T both have signature $[2, 1]$. The foreign key $R[1] \to S$ is weak, and $R[3] \to T$ is strong. The dependency graph of $\mathcal{FK}$ contains directed edges from $(R, 1)$ to every position in $\{(S, 1), (S, 2)\}$, and directed edges from $(R, 3)$ to every position in $\{(T, 1), (T, 2)\}$. The edges ending in $(S, 2)$ or $(T, 2)$ are special. □

The following definition of query containment under foreign keys is borrowed and adapted from [17], where it was studied for general inclusion dependencies. For Boolean queries, containment boils down to logical entailment. Let $\mathcal{FK}$ be a set of foreign keys, and let q and q' be two Boolean queries. We say that q *entails* q' *under* $\mathcal{FK}$, written $q \overset{\mathcal{FK}}{\models} q'$, if for every database instance $\mathbf{db}$ that satisfies $\mathcal{FK}$, if $\mathbf{db} \models q$, then $\mathbf{db} \models q'$. We say that q and q' are *equivalent under* $\mathcal{FK}$, written $q \overset{\mathcal{FK}}{\equiv} q'$, if $q \overset{\mathcal{FK}}{\models} q'$ and $q' \overset{\mathcal{FK}}{\models} q$. For example, if R and S have arity 1 and $\mathcal{FK} = \{R[1] \to S\}$, then $\{R(\underline{x})\} \overset{\mathcal{FK}}{\equiv} \{R(\underline{x}), S(\underline{x})\}$.

Finally, we will restrict the sets $\mathcal{FK}$ of foreign keys that will be allowed for a query q in sjfBCQ. We say that $\mathcal{FK}$ is *about* q if every foreign key in $\mathcal{FK}$ is satisfied by q (when distinct variables are treated as distinct constants) and, moreover, every relation name that occurs in $\mathcal{FK}$ also occurs in q.

3.3 CQA for Primary and Foreign Keys

Symmetric-difference repairs were defined in [1] as follows, for any set of integrity constraints.

We write $\oplus$ for symmetric set difference. Let $\mathbf{db}$ be a database instance. Whenever $\mathbf{r}$, $\mathbf{s}$ are database instances, we write $\mathbf{r} \preceq_{\mathbf{db}} \mathbf{s}$ if $\mathbf{db} \oplus \mathbf{r} \subseteq \mathbf{db} \oplus \mathbf{s}$. If $\mathbf{r} \preceq_{\mathbf{db}} \mathbf{s}$, we also say that $\mathbf{r}$ is $\oplus$*-closer* to $\mathbf{db}$ than $\mathbf{s}$. It can be easily verified that $\preceq_{\mathbf{db}}$ is a partial order on the set of all database instances. We write $\mathbf{r} \prec_{\mathbf{db}} \mathbf{s}$ if $\mathbf{r} \preceq_{\mathbf{db}} \mathbf{s}$ and $\mathbf{r} \neq \mathbf{s}$.

Let $\mathcal{FK}$ be a set of foreign keys. A $\oplus$*-repair* of $\mathbf{db}$ with respect to $\mathcal{FK} \cup \mathcal{PK}$[1](or *repair* for short) is a database instance $\mathbf{r}$ such that: *(i)* $\mathbf{r}$ satisfies $\mathcal{FK} \cup \mathcal{PK}$, and *(ii)* there is no database instance $\mathbf{s}$ such that $\mathbf{s} \prec_{\mathbf{db}} \mathbf{r}$ and $\mathbf{s}$ satisfies $\mathcal{FK} \cup \mathcal{PK}$. A *subset-repair* is a $\oplus$-repair $\mathbf{r}$ satisfying $\mathbf{r} \subseteq \mathbf{db}$, and a *superset-repair* is a $\oplus$-repair $\mathbf{r}$ satisfying $\mathbf{db} \subseteq \mathbf{r}$.

The next example shows that $\oplus$-repairs can be less intuitive and more diverse than subset-repairs or superset-repairs alone.

Example 3.3. Let $q = \{R(\underline{x}, y), S(\underline{y}, z), T(\underline{z})\}$ and $\mathcal{FK} = \{R[2] \to S, S[2] \to T\}$. Let $\mathbf{db} = \{R(\underline{a}, b), S(\underline{b}, c)\}$. Then the following are three $\oplus$-repairs:

$$\mathbf{r}_1 = \{\},$$
$$\mathbf{r}_2 = \{R(\underline{a}, b), S(\underline{b}, 1), T(\underline{1})\},$$
$$\mathbf{r}_3 = \{R(\underline{a}, b), S(\underline{b}, c), T(\underline{c})\}.$$

$\mathbf{r}_1$ is a subset-repair, and $\mathbf{r}_3$ a superset-repair. It may be counter-intuitive that $\mathbf{r}_3$ is not strictly $\oplus$-closer to $\mathbf{db}$ than $\mathbf{r}_2$. Note however:

$$\mathbf{db} \oplus \mathbf{r}_2 = \{S(\underline{b}, c), S(\underline{b}, 1), T(\underline{1})\},$$
$$\mathbf{db} \oplus \mathbf{r}_3 = \{T(\underline{c})\}.$$

Since the latter two sets are not comparable by $\subseteq$, we have that $\mathbf{r}_2$ and $\mathbf{r}_3$ are not comparable by $\preceq_{\mathbf{db}}$. □

Let q be a query in sjfBCQ, and $\mathcal{FK}$ a set of foreign keys about q. We write CERTAINTY$(q, \mathcal{FK})$ for the decision problem that takes as input a database instance and asks whether q is true in every $\oplus$-repair with respect to $\mathcal{FK} \cup \mathcal{PK}$.

The following is relative to a fixed problem CERTAINTY$(q, \mathcal{FK})$. A *consistent first-order rewriting* is a closed first-order formula φ such that a database instance is a "yes"-instance of the problem CERTAINTY$(q, \mathcal{FK})$ if and only if it satisfies φ. Clearly, the existence of a consistent first-order rewriting coincides with the problem being in the complexity class FO.

4 BLOCK-INTERFERENCE

Block-interference is a novel notion that plays a significant role in the complexity classification of CERTAINTY$(q, \mathcal{FK})$. Its definition is technical, but the following example should be helpful to convey the intuition.

Let $q = \{N(\underline{x}, c, y), O(\underline{y})\}$ with $\mathcal{FK} = \{N[3] \to O\}$, where c is a constant. Consider the following database instance, where the value □ in the last N-fact is yet unspecified.

$\mathbf{db} =$

N	$\underline{x}$	c	y
	b_1	c	1
	b_1	d	2
	b_2	c	2
	b_2	d	3
	b_3	c	3
	b_3	d	4
	⋮	⋮	⋮
	b_n	c	n
	b_n	d	$n+1$
	b_{n+1}	□	$n+1$

O	$\underline{y}$
	1

[1]Recall that $\mathcal{PK}$ is the set of primary-key constraints, of the form (1), that can be derived from the relation names in $\mathbf{db}$.

Note that all N-facts, except the first one, are dangling. Our goal is to construct a $\oplus$-repair $\mathbf{r}$ that falsifies q. Such a $\oplus$-repair must obviously choose $N(\underline{b_1}, d, 2)$ in the first N-block, which implies that $O(\underline{2})$ must be inserted. But then $N(\underline{b_2}, c, 2)$ is no longer dangling, and, as a consequence, $\mathbf{r}$ must contain an N-fact from the second N-block. In order to falsify q, $\mathbf{r}$ must choose $N(\underline{b_2}, d, 3)$ in the second block, which implies that $O(\underline{3})$ must be inserted. By repeating the same reasoning, $\mathbf{r}$ must be as follows:

$\mathbf{r} =$

N	$\underline{x}$	c	y
	b_1	d	2
	b_2	d	3
	$\vdots$	$\vdots$	$\vdots$
	b_n	d	$n+1$
	b_{n+1}	$\square$	$n+1$

O	$\underline{y}$
	1
	2
	3
	$\vdots$
	$n+1$

This is a falsifying $\oplus$-repair if (and only if) $\square \neq c$. It is now correct to conclude that $\mathbf{db}$ is a "yes"-instance of CERTAINTY$(q, \mathcal{FK})$ if and only if $\square = c$. Note that for $\mathbf{db}' := \mathbf{db} \setminus \{O(\underline{1})\}$, we have that the empty database instance is a $\oplus$-repair of $\mathbf{db}'$, and hence $\mathbf{db}'$ is a "no"-instance of CERTAINTY$(q, \mathcal{FK})$.

Informally, in deciding whether or not $\mathbf{db}$ is a "yes"-instance of CERTAINTY$(q, \mathcal{FK})$, we had to start from the first N-block, then repeatedly move to the next N-block, and finally inspect the value of $\square$ in the last N-block. It is now unsurprising that CERTAINTY$(q, \mathcal{FK})$ is not in FO (as formally proved in Section 7), because our movement from block to block goes well beyond the locality of first-order logic [26, Chapter 4]. The notion of *block-interference* will capture what is going on in this example. Two more things are to notice:

- The occurrence of the constant c in $N(\underline{x}, c, y)$ is important in the above example, because it is used to distinguish, within each N-block, between satisfying and falsifying N-facts. Instead of a constant, we could have used two occurrences of a same variable, for example, $N(\underline{x}, y, y)$ (and adapt $\mathbf{db}$ accordingly). On the other hand, block-interference disappears if we replace $N(\underline{x}, c, y)$ with $N(\underline{x}, z, y)$ in q, where z is a fresh variable occurring only once.
- Block-interference will also disappear if we replace $O(\underline{y})$ with $O(\underline{y}, c)$ or $O(\underline{y}, y)$ in the above example, because then the O-facts in $\mathbf{r} \setminus \mathbf{db}$ can take the form $O(\underline{i}, \bot)$ for some fresh constant $\bot$ which cannot be used for making the query true. On the other hand, if we replace $O(\underline{y})$ with $O(\underline{y}, w)$ in q, where w is a fresh variable occurring only once, then block-interference will remain.

We now proceed with formalizing block-interference in a number of steps. First, we introduce a concept called *obedience* which, as we will see, plays a central role in block-interference.

Definition 4.1 (Obedience). Let q be a query in sjfBCQ, and $\mathcal{FK}$ a set of foreign keys about q. Let R be a relation name of signature $[n, k]$, and let $P \subseteq \{(R, i) \mid i \in \{k+1, \ldots, n\}\}$ be a set of positions. Define $q_P^{\mathcal{FK}}$ as the smallest subset of q such that if the closure $P_{\mathcal{FK}}$ contains a position (S, j), then $q_P^{\mathcal{FK}}$ contains the S-atom of q. We also write $q_R^{\mathcal{FK}}$ as a shorthand for $q_{P_R}^{\mathcal{FK}}$, where $P_R := \{(R, i) \mid i \in \{k+1, \ldots, n\}\}$.

Let the R-atom of q be $F = R(\vec{s}, t_{k+1}, \ldots, t_n)$, and define $F_P := R(\vec{s}, u_{k+1}, \ldots, u_n)$ where for every $i \in \{k+1, \ldots, n\}$, $u_i = t_i$ if $(R, i) \notin P$, and u_i is a fresh variable otherwise. We say that the set P of positions is *obedient (over $\mathcal{FK}$ and q)* if

$$\left(q \setminus q_P^{\mathcal{FK}}\right) \cup \{F_P\} \overset{\mathcal{FK}}{\models} q, \tag{2}$$

where it is to be noted that the logical entailment in the other direction holds vacuously true (and therefore we also get $\overset{\mathcal{FK}}{\equiv}$-equivalence). Furthermore, we say that atom F is *obedient (over $\mathcal{FK}$ and q)* if the set of positions $\{(R, i) \mid i \in \{k+1, \ldots, n\}\}$ is obedient (over $\mathcal{FK}$ and q). If $\mathcal{FK}$ and q are clear from the context, we may simply say that a set of positions or an atom is obedient. A set of positions (or an atom) is called *disobedient* if it is not obedient.

Example 4.2. Consider again $q = \{N(\underline{x}, c, y), O(\underline{y})\}$ with $\mathcal{FK} = \{N[3] \to O\}$. We first argue that $P_0 := \{(N, 2)\}$ is not obedient. We have $q_{P_0}^{\mathcal{FK}} = \{N(\underline{x}, c, y)\}$, because the dependency graph has an empty path from $(N, 2)$ to itself, and no path from $(N, 2)$ to $(O, 1)$. The left-hand expression in (2) then becomes $\{N(\underline{x}, u_2, y), O(\underline{y})\}$, which is not $\overset{\mathcal{FK}}{\equiv}$-equivalent to q.

We next argue that $P_1 := \{(N, 3)\}$ is obedient. We have $q_{P_1}^{\mathcal{FK}} = q$, because the dependency graph has an empty path from $(N, 3)$ to itself, and an edge from $(N, 3)$ to $(O, 1)$. The left-hand expression in (2) becomes $\{N(\underline{x}, c, u_3)\}$. We have $\{N(\underline{x}, c, u_3)\} \overset{\mathcal{FK}}{\equiv} \{N(\underline{x}, c, u_3), O(\underline{u_3})\}$, and the latter query is obviously $\overset{\mathcal{FK}}{\equiv}$-equivalent to q.

Note finally that the atom $O(\underline{y})$ is obviously obedient, because it has no non-primary-key positions. □

The concept of obedience can also be given a purely syntactic description, which will be useful in the technical treatment. The proof of the following theorem is sketched in Appendix B.

Theorem 4.3 (Syntactic obedience). *Let q be a query in* sjfBCQ*, and $\mathcal{FK}$ a set of unary foreign keys about q. Let $P \subseteq \{(R, i) \mid i \in \{k+1, \ldots, n\}\}$ for some relation name R of signature $[n, k]$. Then, P is obedient if and only if all the following conditions hold true on the dependency graph of $\mathcal{FK}$:*

(I) no position of P belongs to a cycle;
(II) no constant occurs in q at a position of $P_{\mathcal{FK}}$;
(III) no variable occurs in q both at a position of $P_{\mathcal{FK}}$ and a position of $P_{\mathcal{FK}}^{\text{co}}$; and
(IV) no variable occurs in q at two distinct non-primary-key positions of $P_{\mathcal{FK}}$.

Theorem 4.3 has the following immediate corollary, which implies that obedience can be treated as a property of single positions.

Corollary 4.4. *Let q, $\mathcal{FK}$, and P be as in the statement of Theorem 4.3. Then, P is obedient over $\mathcal{FK}$ and q if and only if $\{(R, i)\}$ is obedient over $\mathcal{FK}$ and q for all $(R, i) \in P$.*

Informally, Theorem 4.3 implies that if a set P of positions is obedient, then $P_{\mathcal{FK}}$ is of the form depicted in Fig. 2, where arrows represent foreign keys and primary-key positions are boxed (relation names are omitted). In particular, the figure shows the absence of cycles, constants, and variables that are repeated within a same atom.

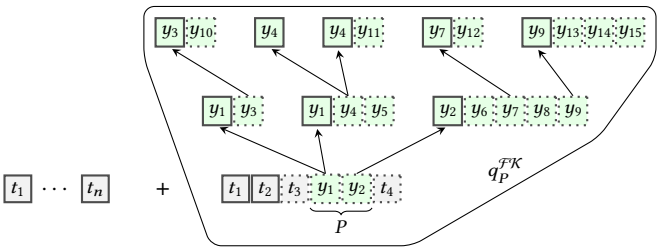

Figure 2: Structure of $q_P^{\mathcal{FK}}$ over obedient P (omitting weak foreign keys). Terms $t_1, \ldots, t_n$ occupying $P^{\text{co}}_{\mathcal{FK}}$ do not occur among the (pairwise distinct) variables $y_1, \ldots, y_{15}$ occupying $P_{\mathcal{FK}}$. The polygon encloses $q_P^{\mathcal{FK}}$; the green boxes mark $P_{\mathcal{FK}}$.

We now come to Definition 4.5 of block-interference, which uses the following adapted notion of Gaifman graph [26, Def. 4.1]. For a query in sjfBCQ and $V \subseteq \text{vars}(q)$, we define $\mathcal{G}_V(q)$ for the undirected graph whose vertex-set is V, and where $\{x, y\}$ is an undirected edge if $x = y$ or there is $F \in q$ such that $\{x, y\} \subseteq \text{vars}(F) \cap V$.

Definition 4.5 (Block-interfering). Let q be a query in sjfBCQ, and $\mathcal{FK}$ a set of foreign keys about q. Let $N[j] \to O$ be a strong foreign key in $\mathcal{FK}^*$. Let $N(\underline{t_1, \ldots, t_k}, t_{k+1}, \ldots, t_n)$ and $O(\underline{t_j}, \vec{y})$ be atoms in q (since the foreign key is strong, $j > k$). Let $V = \{v \in \text{vars}(q') \mid \mathcal{K}(q) \not\models \emptyset \to \{v\}\}$, where $q' := q \setminus \{N(\underline{t_1, \ldots, t_k}, t_{k+1}, \ldots, t_n)\}$. We say that this foreign key is *block-interfering (in q)* if the following hold:

(1) the atom $O(\underline{t_j}, \vec{y})$ is obedient;
(2) t_j is a variable in V (thus $\mathcal{K}(q) \not\models \emptyset \to \{t_j\}$); and
(3) at least one of the following holds true:
   (a) $\{(N, k+1), \ldots, (N, n)\} \setminus \{(N, j)\}$ is not obedient; or
   (b) for some $i \in \{1, \ldots, k\}$, t_i and t_j are (not necessarily distinct) variables that are connected in $\mathcal{G}_V(q')$.

We say that the pair $(q, \mathcal{FK})$ has *block-interference* if some foreign key of $\mathcal{FK}^*$ is block-interfering in q. □

It can be seen that, due to properties (3a) or (3b) in Definition 4.5, the N-atom in this definition will itself be disobedient.

Example 4.6. Continuing Example 4.2, consider again the query $q = \{N(\underline{x}, c, y), O(\underline{y})\}$ with $\mathcal{FK} = \{N[3] \to O\}$, where the atom $O(\underline{y})$ is obviously obedient. Following the notations of Definition 4.5, we obtain block-interference by letting $j = 3$ and therefore $t_j = y$. The set difference in item (3a) of Definition 4.5 becomes $\{(N, 2)\}$, which is not obedient as shown in Example 4.2. □

The following example shows the use of property (3b) in Definition 4.5.

Example 4.7. Consider $q_0 = \{N'(\underline{x}, y), O(\underline{y}), T(\underline{x}, \underline{y})\}$ and $\mathcal{FK} = \{N'[2] \to O\}$. In comparison with the previous Example 4.6, we removed the constant c that allowed us to distinguish, within an N-block, between satisfying and falsifying N-facts. However, since x and y occur together in the T-atom of q_0, we can now use T-facts to make this distinction. Indeed, in the database **db** at the beginning of this section, we can replace every "satisfying" fact $N(\underline{b_i}, c, i)$ with two facts $N'(\underline{b_i}, i)$ and $T(\underline{b_i}, \underline{i})$, while every "falsifying" fact $N(\underline{b_i}, d, i+1)$ is replaced with a single fact $N'(\underline{b_i}, i+1)$ (for $1 \leq i \leq n+1$). Informally, the role of the constant c is now played by T.

To illustrate the role of the set V in Definition 4.5, we note that our construction with T-facts would fail if for some constant a, the query q_0 also contained $R(\underline{a}, x)$ (yielding a functional dependency $\emptyset \to \{x\}$), because no $\oplus$-repair can contain both $R(\underline{a}, b_i)$ and $R(\underline{a}, b_j)$ with $i \neq j$. □

5 MAIN THEOREM

The following theorem refines Theorem 1.1 by adding the conditions to decide whether or not CERTAINTY$(q, \mathcal{FK})$ is in FO. To show that a problem CERTAINTY$(q, \mathcal{FK})$ is not in FO, we show that it is L-hard or NL-hard.

Theorem 5.1. *Let q be a query in* sjfBCQ*, and let $\mathcal{FK}$ be a set of unary foreign keys about q. Then,*

(1) if the attack graph of q is acyclic and $(q, \mathcal{FK})$ has no block-interference, then CERTAINTY$(q, \mathcal{FK})$ *is in* FO *(and its consistent first-order rewriting can be effectively constructed);*
(2) if the attack graph of q is cyclic, then CERTAINTY$(q, \mathcal{FK})$ *is* L*-hard (and therefore not in* FO*); and*
(3) if $(q, \mathcal{FK})$ has block-interference, then CERTAINTY$(q, \mathcal{FK})$ *is* NL*-hard (and therefore not in* FO*).*

Moreover, it can be decided, given q and $\mathcal{FK}$, which case applies.

There is an easy proof for the last line in the statement of the above theorem. Indeed, it is known that, given q in sjfBCQ, it can be decided in quadratic time whether or not q's attack graph is acyclic [22, Theorem 3.2]. Moreover, it is clear that the existence of block-interference is decidable in polynomial time by inspecting the conditions in Definition 4.5 and the syntactic characterization of obedience in Theorem 4.3.

The following example illustrates Theorem 5.1, and shows that consistent query answering over foreign keys depends in a subtle way on the syntax of the query.

Example 5.2. For variables x, y, z, w, and a constant c, let

$$\mathcal{FK} = \{N[3] \to O\};$$
$$q_1 = \{N(\underline{x}, u, y), O(\underline{y}, w)\};$$
$$q_2 = \{N(\underline{x}, c, y), O(\underline{y}, w)\};$$
$$q_3 = \{N(\underline{x}, c, y), O(\underline{y}, c)\}.$$

Note that q_2 and q_3 can be obtained from q_1 by replacing variables with constants: $q_2 = q_{1[u \to c]}$ and $q_3 = q_{1[u,w \to c,c]}$. The attack graph of each query is acyclic, and hence CERTAINTY(q_i) is in FO for $i \in \{1, 2, 3\}$. The complexity and consistent first-order rewritings change as follows in the presence of $\mathcal{FK}$.

- CERTAINTY$(q_1, \mathcal{FK})$ is in FO because $N[3] \to O$ is not block-interfering in q_1, even though the atom $O(\underline{y}, w)$ is obedient. It can be formally verified that condition (3a) in Definition 4.5 is not satisfied: the position $(N, 2)$ in q_1 is obedient, because it is occupied by a variable that occurs only once in the query. The consistent first-order rewriting for CERTAINTY$(q_1, \mathcal{FK})$ is the query q_1 itself. Remarkably, this is different from the consistent first-order rewriting for CERTAINTY(q_1) (i.e., in the absence of foreign keys). To see

the difference, note that the following database instance is a "yes"-instance of CERTAINTY($q_1, \mathcal{FK}$), but a "no"-instance of CERTAINTY(q_1).

N	$\underline{x}$	u	y
	c	1	a
	c	2	b

O	$\underline{y}$	w
	a	3

- CERTAINTY($q_2, \mathcal{FK}$) is NL-hard, because $N[3] \to O$ is block-interfering in q_2. Informally, this is because the position $(N, 2)$ is now occupied by a constant c and therefore not obedient.
- CERTAINTY($q_3, \mathcal{FK}$) is again in FO, because $N[3] \to O$ is not block-interfering in q_2. The reason is that the O-atom is no longer obedient because its non-primary-key position is now occupied by a constant. With some effort, one can see that CERTAINTY($q_3, \mathcal{FK}$) and CERTAINTY(q_3) have the same consistent first-order rewriting.

To conclude, replacing a variable by a constant can increase or decrease the complexity, depending on where the variable occurs. This behavior is typical of foreign keys, and does not occur in the case of only primary keys. □

The following sections are devoted to the proof of Theorem 5.1. In Section 6, we prove item (2) of Theorem 5.1, and in Section 7 we prove item (3). Finally, item (1) is shown in Section 8.

6 L-HARDNESS

We know from Theorem 3.1 that CERTAINTY($q, \mathcal{FK}$) is L-hard if $\mathcal{FK} = \emptyset$ and the attack graph of q is cyclic. The following lemma tells us that this complexity lower bound remains valid if we add foreign keys to $\mathcal{FK}$. It is worth mentioning that it can be proved for foreign keys that need not be unary (see Appendix C).

Lemma 6.1. *Let q be a query in* sjfBCQ*, and $\mathcal{FK}$ be a set of foreign keys about q. If q has a cyclic attack graph, then* CERTAINTY($q, \mathcal{FK}$) *is* L*-hard.*

For example, since the attack graph of $q = \{R(\underline{x}, y), S(\underline{y}, x)\}$ is cyclic, CERTAINTY($q, \mathcal{FK}$) is L-hard, for every $\mathcal{FK}$ that is a (possibly empty) subset of $\{R[2] \to S, S[2] \to R\}$.

7 NL-HARDNESS

The following lemma, proven in [15], restates item (3) of Theorem 5.1.

Lemma 7.1. *Let q be a query in* sjfBCQ*, and $\mathcal{FK}$ be a set of unary foreign keys about q. If $(q, \mathcal{FK})$ has block-interference, then the problem* CERTAINTY($q, \mathcal{FK}$) *is* NL*-hard.*

For an intuition why block-interference leads to NL-hardness, consider again the example with $q = \{N(\underline{x}, c, y), O(\underline{y})\}$ and $\mathcal{FK} = \{N[3] \to O\}$, elaborated in the beginning of Section 4, where it was argued that CERTAINTY($q, \mathcal{FK}$) goes beyond locality of first-order logic. With this preceding example in mind, it should not come as a surprise that directed graph reachability can be reduced to (the complement of) CERTAINTY($q, \mathcal{FK}$). In graph reachability, the input consists of a directed graph and two vertices (s and t), and the question is whether there is a directed path from s to t.

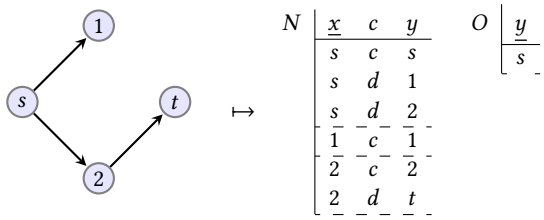

Figure 3: Reduction from graph reachability.

The problem is NL-hard, even if the graphs are acyclic. Figure 3 illustrates a straightforward reduction: for every vertex v such that $v \neq t$, add an N-fact $N(\underline{v}, c, v)$; for every directed edge (u, w), add $N(\underline{u}, d, w)$. Finally, add $O(\underline{s})$. The path from s to t (via vertex 2) in the database instance of Fig. 3 can be cooked into the following $\oplus$-repair that falsifies q:

N	$\underline{x}$	c	y
	s	d	2
	2	d	t

O	$\underline{y}$
	s
	2
	t

On the other hand, it can be easily verified that there would be no falsifying $\oplus$-repair if every path starting from s ended in a vertex other than t. The reasoning is analogous to the one used in the beginning of Section 4.

The previous example gives a correct intuition for the proof of Lemma 7.1. The reason why its proof is technically much more involved is that Definition 4.5 (and especially condition (3a) in it) exhibits several ways in which block-interference can arise. In the previous example, we only looked at the very simple case where block-interference uses a constant. In more difficult situations, block-interference arises from cycles in the dependency graph or repetitions of variables.

In the absence of foreign keys, for every q in sjfBCQ, the problem CERTAINTY(q) is either in FO, L-complete, or coNP-complete [25]. Interestingly, in the presence of foreign keys, NL-completeness and P-completeness also pop up, as shown next.

Proposition 7.2. CERTAINTY($q, \mathcal{FK}$) *is* NL*-complete for $q = \{N(\underline{x}, x), O(\underline{x})\}$ and $\mathcal{FK} = \{N[2] \to O\}$.*

Proposition 7.3. CERTAINTY($q, \mathcal{FK}$) *is* P*-complete for $q = \{N(\underline{x}, c, y), O(\underline{y})\}$ and $\mathcal{FK} = \{N[3] \to O\}$.*

A fine-grained complexity classification for *all* problems in the set $\{\text{CERTAINTY}(q, \mathcal{FK}) \mid q \in \text{sjfBCQ}$ and $\mathcal{FK}$ is about $q\}$ is open; in the current paper, we succeed in tracing the FO-boundary in the above set.

8 FIRST-ORDER REWRITABILITY

The following lemma restates item (1) of Theorem 5.1.

Lemma 8.1. *Let q be a query in* sjfBCQ*, and $\mathcal{FK}$ a set of unary foreign keys about q. If the attack graph of q is acyclic and $(q, \mathcal{FK})$ has no block-interference, then* CERTAINTY($q, \mathcal{FK}$) *is in* FO *(and its consistent first-order rewriting can be effectively constructed).*

R-atom	S-atom	Type	
		$\overset{\text{weak}}{\longrightarrow}$	Lemma D.2
obedient	obedient	$\text{o} \overset{\text{str}}{\longrightarrow} \text{o}$	Lemma D.3
disobedient	disobedient	$\text{d} \overset{\text{str}}{\longrightarrow} \text{d}$	Lemma D.4
disobedient	obedient	$\text{d} \overset{\text{str}}{\longrightarrow} \text{o}$	Lemmas D.5 and D.6

Figure 4: Reductions that remove foreign keys $R[i] \to S$.

We sketch how the previous lemma is proved. For two decision problems P_1 and P_2, we write $P_1 \leq_m^{\mathsf{FO}} P_2$ if there exists a first-order many-one reduction from P_1 to P_2.

Let q and $\mathcal{FK}$ be as stated in Lemma 8.1 such that the attack graph of q is acyclic and $(q, \mathcal{FK})$ has no block-interference. The proof strategy is to show that one can construct a query q' in sjfBCQ such that q' has an acyclic attack graph and

$$\mathsf{CERTAINTY}(q, \mathcal{FK}) \leq_m^{\mathsf{FO}} \mathsf{CERTAINTY}(q', \emptyset). \tag{3}$$

Since the latter problem has an empty set of foreign keys, it is in FO by Theorem 3.1.

Equation (3) is shown by a composition of first-order reductions, each of which removes at least one foreign key, and some of which remove obedient atoms or replace variables with constants. The helping lemmas that define these reductions are summarized in Fig. 4 and are given in Appendix D. We distinguish between four *types of foreign keys*. A *strong* foreign key $R[i] \to S$ is of a type in $\{\text{o} \overset{\text{str}}{\longrightarrow} \text{o}, \text{d} \overset{\text{str}}{\longrightarrow} \text{d}, \text{d} \overset{\text{str}}{\longrightarrow} \text{o}\}$, depending on whether the R-atom or S-atom are obedient (symbol o) or disobedient (symbol d). Note that there is no type $\text{o} \overset{\text{str}}{\longrightarrow} \text{d}$, because if the R-atom is obedient and the foreign key is strong, then the S-atom is necessarily obedient as well. For weak foreign keys there is only one type, denoted $\overset{\text{weak}}{\longrightarrow}$.

Note in Definition 4.5 that only foreign keys of type $\text{d} \overset{\text{str}}{\longrightarrow} \text{o}$ can be block-interfering. Unsurprisingly, the requirement, in Lemma 8.1, that $(q, \mathcal{FK})$ has no block-interference is used in (and only in) the helping Lemma D.5 that shows the removal of foreign keys of type $\text{d} \overset{\text{str}}{\longrightarrow} \text{o}$.

It becomes apparent from the proofs of the helping lemmas that whenever $\mathsf{CERTAINTY}(q, \mathcal{FK})$ is in FO, its consistent first-order rewriting is very similar to that of $\mathsf{CERTAINTY}(q)$ [22], except for obedient atoms referenced by strong foreign keys. For example, consider $q = \{N(\underline{c}, y), O(\underline{y}), P(\underline{y})\}$ with $\mathcal{FK} = \{N[2] \to O\}$, where O is referenced but P is not. The following is a consistent first-order rewriting for $\mathsf{CERTAINTY}(q, \mathcal{FK})$:

$$\exists y \left(N(\underline{c}, y) \wedge O(\underline{y}) \right) \wedge \forall y \left(N(\underline{c}, y) \to P(\underline{y}) \right).$$

Note the asymmetric treatment of O and P in the above formula. In this respect, it is instructive to note that the following database instance satisfies the previous formula and hence is a "yes"-instance. However, removing either $P(\underline{a})$ or $P(\underline{b})$ turns it into a "no"-instance.

N		
	$\underline{c}$	y
	c	a
	c	b

O	
	$\underline{y}$
	a

P	
	$\underline{y}$
	a
	b

9 DISCUSSION

While CQA for primary keys was successfully studied in the past 15 years, CQA with respect to both primary and foreign keys remained largely unexplored. We made a significant contribution by tracing the FO-boundary in the set $\{\mathsf{CERTAINTY}(q, \mathcal{FK}) \mid q \in \text{sjfBCQ}$ and $\mathcal{FK}$ is about $q\}$, under the restriction that foreign keys are unary (but primary keys can be composite). If $\mathcal{FK} = \emptyset$, then these problems only have primary-key constraints, in which case a complete complexity classification in FO, L-complete, and coNP-complete is already known [25]. For non-empty sets $\mathcal{FK}$, a complete complexity classification beyond FO is left open. Our paper nevertheless shows that the complexity landscape is more diverse than for primary keys alone, as Propositions 7.2 and 7.3 show that there are NL-complete and P-complete problems in the above set of problems.

It is an open research task to release our restrictions that foreign-keys are unary and are about the query, as discussed next.

- Our assumption that all foreign keys are unary excludes, for example, a query with atoms $R(\underline{x}, y, z)$, $S(\underline{x, z}, y)$ and foreign key $R[1, 3] \to S$. The difficulty here is that the foreign key covers both a primary-key and a non-primary-key position of R. In future research, we will investigate how our constructs of obedience and block-interfering can be generalized to composite foreign keys.
- Our assumption that all foreign keys are about the query excludes, for example, the problem in the following Proposition 9.1, because $q = \{E(\underline{x}, y)\}$ does not satisfy $E[2] \to E$ (when x and y are treated as distinct constants).

  Proposition 9.1. *Let $q = \{E(\underline{x}, y)\}$ and $\mathcal{FK} = \{E[2] \to E\}$. Then,* $\mathsf{CERTAINTY}(q, \mathcal{FK})$ *is* NL*-hard.*

  Concerning the previous proposition, note that every conjunctive query q' that includes q and satisfies $\mathcal{FK}$ contains a self-join. The shortest such a query is $q' = \{E(\underline{x}, y), E(\underline{y}, x)\}$. CQA for conjunctive queries with self-joins is a notorious open problem, even in the absence of foreign keys.

ACKNOWLEDGMENTS

Miika Hannula has been supported by Academy of Finland grants 308712 and 322795.

REFERENCES

[1] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. 1999. Consistent Query Answers in Inconsistent Databases. In *PODS*. ACM Press, 68–79.
[2] Leopoldo E. Bertossi. 2019. Database Repairs and Consistent Query Answering: Origins and Further Developments. In *PODS*. ACM, 48–58.
[3] Meghyn Bienvenu. 2020. A Short Survey on Inconsistency Handling in Ontology-Mediated Query Answering. *Künstliche Intell.* 34, 4 (2020), 443–451.
[4] Andrei A. Bulatov. 2011. Complexity of conservative constraint satisfaction problems. *ACM Trans. Comput. Log.* 12, 4 (2011), 24:1–24:66.
[5] Marco Calautti, Marco Console, and Andreas Pieris. 2019. Counting Database Repairs under Primary Keys Revisited. In *PODS*. ACM, 104–118.
[6] Marco Calautti, Marco Console, and Andreas Pieris. 2021. Benchmarking Approximate Consistent Query Answering. In *PODS*. ACM, 233–246.
[7] Marco Calautti, Leonid Libkin, and Andreas Pieris. 2018. An Operational Approach to Consistent Query Answering. In *PODS*. ACM, 239–251.
[8] Akhil A. Dixit and Phokion G. Kolaitis. 2019. A SAT-Based System for Consistent Query Answering. In *SAT (Lecture Notes in Computer Science, Vol. 11628)*. Springer, 117–135.
[9] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. 2005. Data exchange: semantics and query answering. *Theor. Comput. Sci.* 336, 1 (2005), 89–124.

[10] 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.
[11] Ariel Fuxman, Elham Fazli, and Renée J. Miller. 2005. ConQuer: Efficient Management of Inconsistent Databases. In *SIGMOD Conference*. ACM, 155–166.
[12] Ariel Fuxman, Diego Fuxman, and Renée J. Miller. 2005. ConQuer: A System for Efficient Querying Over Inconsistent Databases. In *VLDB*. ACM, 1354–1357.
[13] Ariel Fuxman and Renée J. Miller. 2005. First-Order Query Rewriting for Inconsistent Databases. In *ICDT (Lecture Notes in Computer Science, Vol. 3363)*. Springer, 337–351.
[14] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. 2020. Cleaning data with Llunatic. *VLDB J.* 29, 4 (2020), 867–892.
[15] Miika Hannula and Jef Wijsen. 2022. A Dichotomy in Consistent Query Answering for Primary Keys and Unary Foreign Keys. https://doi.org/10.48550/ARXIV.2203.13475
[16] Ihab F. Ilyas and Xu Chu. 2019. *Data Cleaning*. ACM.
[17] David S. Johnson and Anthony C. Klug. 1984. Testing Containment of Conjunctive Queries under Functional and Inclusion Dependencies. *J. Comput. Syst. Sci.* 28, 1 (1984), 167–189.
[18] Aziz Amezian El Khalfioui, Jonathan Joertz, Dorian Labeeuw, Gaëtan Staquet, and Jef Wijsen. 2020. Optimization of Answer Set Programs for Consistent Query Answering by Means of First-Order Rewriting. In *CIKM*. ACM, 25–34.
[19] Benny Kimelfeld, Ester Livshits, and Liat Peterfreund. 2020. Counting and enumerating preferred database repairs. *Theor. Comput. Sci.* 837 (2020), 115–157.
[20] Phokion G. Kolaitis, Enela Pema, and Wang-Chiew Tan. 2013. Efficient Querying of Inconsistent Databases with Binary Integer Programming. *Proc. VLDB Endow.* 6, 6 (2013), 397–408.
[21] Paraschos Koutris, Xiating Ouyang, and Jef Wijsen. 2021. Consistent Query Answering for Primary Keys on Path Queries. In *PODS*. ACM, 215–232.
[22] 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.
[23] Paraschos Koutris and Jef Wijsen. 2018. Consistent Query Answering for Primary Keys and Conjunctive Queries with Negated Atoms. In *PODS*. ACM, 209–224.
[24] Paraschos Koutris and Jef Wijsen. 2020. First-Order Rewritability in Consistent Query Answering with Respect to Multiple Keys. In *PODS*. ACM, 113–129.
[25] Paraschos Koutris and Jef Wijsen. 2021. Consistent Query Answering for Primary Keys in Datalog. *Theory Comput. Syst.* 65, 1 (2021), 122–178.
[26] Leonid Libkin. 2004. *Elements of Finite Model Theory*. Springer.
[27] Dany Maslowski and Jef Wijsen. 2013. A dichotomy in the complexity of counting database repairs. *J. Comput. Syst. Sci.* 79, 6 (2013), 958–983.
[28] Dany Maslowski and Jef Wijsen. 2014. Counting Database Repairs that Satisfy Conjunctive Queries with Self-Joins. In *ICDT*. OpenProceedings.org, 155–164.
[29] Slawek Staworko, Jan Chomicki, and Jerzy Marcinkowski. 2012. Prioritized repairing and consistent query answering in relational databases. *Ann. Math. Artif. Intell.* 64, 2-3 (2012), 209–246.
[30] Jef Wijsen. 2010. On the first-order expressibility of computing certain answers to conjunctive queries over uncertain databases. In *PODS*. ACM, 179–190.
[31] Jef Wijsen. 2019. Foundations of Query Answering on Inconsistent Databases. *SIGMOD Rec.* 48, 3 (2019), 6–16.

A HELPING NOTIONS AND LEMMAS

In this section, we define more preliminary notions and helping lemmas. The following definitions are relative to a database instance **db**, a query q in sjfBCQ, and a set $\mathcal{FK}$ of foreign keys.

We write $\mathrm{adom}(\mathbf{db})$ for the set of constants that occur in **db**, also called its *active domain*.

A variable $x \in \mathrm{vars}(q)$ is called *orphan (in q)* if x occurs only once in q, and this single occurrence is at a non-primary-key position. Similarly, a constant c in **db** is called *orphan (in* **db***)* if c occurs only once in **db**, and this single occurrence is at a non-primary-key position.

Two variables $x, y \in \mathrm{vars}(q)$ are said to be *connected in q* if $x = y$ or there exists a sequence $x_0, x_1, \ldots, x_\ell$ of variables in $\mathrm{vars}(q)$ such that $x_0 = x$, $x_\ell = y$, and every two adjacent variables occur together in some atom of q.

We say that a fact A of **db** is *relevant for q in* **db** if there exists a valuation θ over $\mathrm{vars}(q)$ such that $A \in \theta(q) \subseteq \mathbf{db}$ (and therefore $A \in \mathbf{db}$); otherwise A is *irrelevant*. A block of **db** is *relevant* if it contains at least one relevant fact.

We write $\mathbf{db}\restriction_q$ for the restriction of **db** to those facts whose relation name occurs in q. We write $\mathcal{FK}\restriction_q$ for the set of those foreign keys in $\mathcal{FK}$ that only use relation names in q. Clearly, if $\mathcal{FK}$ is about q, then $\mathcal{FK}\restriction_q = q$.

If R is a relation name with signature $[n, 1]$, then the (weak) foreign key $R[1] \to R$ is called *trivial*, because it cannot be falsified. If $\mathcal{FK}$ is a set of foreign keys and R a relation name, then $\mathcal{FK}[R \to]$ is the set of foreign keys in $\mathcal{FK}$ that are outgoing from R, and $\mathcal{FK}[\to R]$ is the set of foreign keys in $\mathcal{FK}$ that are referencing R.

Lemma A.1. *Let $\mathcal{PK} \cup \mathcal{FK}$ be a set of primary keys and foreign keys. Let* **db** *be a (possibly inconsistent) database instance, and let* **r** *be a $\oplus$-repair of* **db**. *Let* **s** *be a database instance such that* $\mathbf{s} \subseteq \mathbf{r} \cup \mathbf{db}$ *and* $\mathbf{s} \models \mathcal{PK} \cup \mathcal{FK}$. *For every fact $A \in \mathbf{s} \cap \mathbf{db}$, there is a fact $A' \in \mathbf{r} \cap \mathbf{db}$ such that $A' \sim A$.*

Proof. Let $\mathbf{s} \setminus \mathbf{r} = \{A_1, A_2, \ldots, A_n\}$. Since $\mathbf{s} \subseteq \mathbf{r} \cup \mathbf{db}$, each A_i belongs to **db**. Let $\mathbf{t}_0 := \mathbf{r}$. For $i = 1, 2, \ldots, n$,

(1) if $\mathbf{r} \cap \mathbf{db}$ contains a fact that is key-equal to A_i, let $\mathbf{t}_i := \mathbf{t}_{i-1}$;
(2) if $\mathbf{r} \setminus \mathbf{db}$ contains a fact A'_i that is key-equal to A_i, let $\mathbf{t}_i := \big(\mathbf{t}_{i-1} \setminus \{A'_i\}\big) \cup \{A_i\}$; and
(3) if **r** contains no fact that is key-equal to A_i, let $\mathbf{t}_i := \mathbf{t}_{i-1} \cup \{A_i\}$.

Let $\mathbf{t} := \mathbf{t}_n$. From $\mathbf{r} \models \mathcal{PK}$ and $\mathbf{s} \models \mathcal{PK}$, it follows $\mathbf{t} \models \mathcal{PK}$ by construction.

We show that $\mathbf{t} \models \mathcal{FK}$. To this end, let $R[i] \to S$ be a foreign key in $\mathcal{FK}$, and let $R(a_1, \ldots, a_n)$ be a fact in **t**. If $R(a_1, \ldots, a_n) \in \mathbf{r}$, then this foreign key is satisfied by **t** because $\mathbf{r} \models \mathcal{FK}$ and, by construction, every fact in **r** is key-equal to a fact in **t**. Assume next that $R(a_1, \ldots, a_n) \in \mathbf{s} \setminus \mathbf{r}$. Since $R(a_1, \ldots, a_n) \in \mathbf{s}$ and $\mathbf{s} \models \mathcal{FK}$, **s** contains a fact $S(\underline{a_i}, \_)$. By construction, **t** will contain a fact that is key-equal to $S(\underline{a_i}, \_)$.

By construction, $\mathbf{r} \cap \mathbf{db} \subseteq \mathbf{t}$ and $\mathbf{t} \subseteq \mathbf{r} \cup \mathbf{db}$. It follows $\mathbf{t} \preceq_{\mathbf{db}} \mathbf{r}$. If (2) or (3) are applied once or more, then $\mathbf{t} \prec_{\mathbf{db}} \mathbf{r}$, contradicting that **r** is a $\oplus$-repair. It follows that only (1) applies, which means that for every $A \in \mathbf{s} \cap \mathbf{db}$, **r** contains a fact of $\mathrm{block}(A, \mathbf{db})$. □

Lemma A.2. *Let q be a query in* sjfBCQ, *and $\mathcal{FK}$ a set of foreign keys that is satisfied by q (when distinct variables are treated as distinct constants). Let* **db** *be a (possibly inconsistent) database instance. Let* **r** *be a database instance that satisfies $\mathcal{FK} \cup \mathcal{PK}$. Let θ be a valuation over* $\mathrm{vars}(q)$ *satisfying the following conditions:*

(1) $\theta(q) \subseteq \mathbf{db} \cup \mathbf{r}$; and
(2) there is a fact $A \in \theta(q) \setminus \mathbf{r}$ such that $\mathbf{r} \cap \mathbf{db}$ contains no fact that is key-equal to A.

Then **r** *is not a $\oplus$-repair.*

Proof. Since $q \models \mathcal{FK}$ and since $q \in$ sjfBCQ, we have $\theta(q) \models \mathcal{PK} \cup \mathcal{FK}$. Assume towards a contradiction that **r** is a $\oplus$-repair. By Lemma A.1, for every fact $A \in \theta(q) \cap \mathbf{db}$, there is a fact $A' \in \mathbf{r} \cap \mathbf{db}$ such that $A' \sim A$, contradicting (2). □

B PROOFS FOR SECTION 4

In this section we show that the concept of obedience can be characterized in syntactic terms (Theorem 4.3). The proof relies on Lemma B.1 which is proven in [15].

For a database instance $\mathbf{db}$, define

$$\mathsf{keyconst}(\mathbf{db}) \coloneqq \mathsf{adom}(\{R'(\vec{a}) \mid \exists \vec{b} : R(\underline{\vec{a}}, \vec{b}) \in \mathbf{db}\});$$

in words, $\mathsf{keyconst}(\mathbf{db})$ is the set of constants that appear at a primary-key position in $\mathbf{db}$.

Lemma B.1. *Let q be a query in* sjfBCQ*, and let $\mathcal{FK}$ be a set of unary foreign keys that is about q. Let $\mathbf{db}$ be a database instance, and let $A = R(\underline{\vec{a}}, b_{k+1}, \ldots, b_n) \in \mathbf{db}$. Let $P \subseteq \{(R,i) \mid i \in \{k+1, \ldots, n\}\}$ be a set of positions that violates some of the conditions listed in Theorem 4.3. Assume that b_i is an orphan constant of $\mathbf{db}$ that does not belong to $\mathsf{const}(q)$, for all $(R,i) \in P$. Then, there exists a database instance $\mathbf{db}_{A,P}$ such that*

(1) $\mathsf{keyconst}(\mathbf{db}) \cap \mathsf{adom}(\mathbf{db}_{A,P}) = \emptyset$*;*
(2) $\mathbf{db}_{A,P} \models \mathcal{FK}$*;*
(3) A is not dangling in $\{A\} \cup \mathbf{db}_{A,P}$ with respect to any $R[i] \to S \in \mathcal{FK}$ such that $(R,i) \in P$; and
(4) every fact of $\{A\} \cup \mathbf{db}_{A,P}$ is irrelevant for q in $\mathbf{db} \cup \mathbf{db}_{A,P}$.

Example B.2. Let $q = \{N(\underline{x}, x), O(\underline{x}, y)\}$ and $\mathcal{FK} = \{N[2] \to N, N[2] \to O\}$. Consider the following database instance $\mathbf{db}$:

$$\mathbf{db} = \begin{array}{|c|c|}\hline N & \\ \hline \underline{x} & x \\ \hline a & a \\ b & c \\ \hline \end{array} \qquad \begin{array}{|c|c|}\hline O & \\ \hline \underline{x} & y \\ \hline a & b \\ \hline \end{array}\ .$$

Select $A = N(\underline{b}, c)$, $P = \{(N,2)\}$, and observe that $\{(N,2)\}$ belongs to a cycle in the dependency graph, thus violating Theorem 4.3 (I). As Lemma B.1 predicts, we find a database instance $\mathbf{db}_{A,P}$ satisfying all the items of the lemma statement:

$$\mathbf{db}_{A,P} = \begin{array}{|c|c|}\hline N & \\ \hline \underline{x} & x \\ \hline c & \bot \\ \bot & c \\ \hline \end{array} \qquad \begin{array}{|c|c|}\hline O & \\ \hline \underline{x} & y \\ \hline c & \bot \\ \bot & c \\ \hline \end{array}\ ,$$

where $\bot$ is a fresh variable. □

We next turn to the proof of Theorem 4.3.

Proof of Theorem 4.3. It is straightforward to verify that the empty set of positions is obedient and satisfies all the items listed in Theorem 4.3. From here on, we assume that P is non-empty. We also assume that the unique R-atom of q is of the form $F = R(\underline{\vec{s}}, t_{k+1}, \ldots, t_n)$, and define

$$q' \coloneqq \left(q \setminus q_P^{\mathcal{FK}}\right) \cup \{F_P\},$$

where F_P is obtained from F by substituting fresh variables for the terms occurring at positions of P (see Definition 4.1).

$\boxed{\Longrightarrow}$ We show the contraposition. Assume that some of the conditions listed in Theorem 4.3 is violated. Let θ be a one-to-one valuation mapping variables x to constants c_x (that are not from $\mathsf{const}(q)$). We can then apply Lemma B.1 to obtain a database instance $\mathbf{db}_{A,P}$ given $A \coloneqq \theta(F_P)$ and $\mathbf{db} \coloneqq \theta(q')$. The lemma states that every fact of $\{A\} \cup \mathbf{db}_{A,P}$ is irrelevant for q in $\mathbf{db} \cup \mathbf{db}_{A,P}$. This entails that no R-fact is relevant for q in $\mathbf{db} \cup \mathbf{db}_{A,P}$, whence $\mathbf{db} \cup \mathbf{db}_{A,P} \not\models q$. On the other hand, it is obvious that $\mathbf{db} \cup \mathbf{db}_{A,P} \models q'$. Finally, $\mathbf{db} \cup \mathbf{db}_{A,P} \models \mathcal{FK}$ follows by Lemma B.1 and the fact that $\mathcal{FK}$ is about q. We thus conclude that $q' \overset{\mathcal{FK}}{\not\models} q$, i.e., P is disobedient.

$\boxed{\Longleftarrow}$ Let F be the unique R-atom of q. Assuming conditions (I)–(IV) in Theorem 4.3 hold true, we show that P is obedient, i.e., $q' \overset{\mathcal{FK}}{\models} q$.

Suppose $\mathbf{db}$ is a database that satisfies both q' and $\mathcal{FK}$. We need to show that $\mathbf{db}$ satisfies also q. Let θ_0 be a valuation such that $\theta_0(q') \subseteq \mathbf{db}$. In what follows, we will extend θ_0 to a valuation θ such that $\theta(q) \subseteq \mathbf{db}$.

Let $(G_1, \ldots, G_m)$ list the atoms of $q_P^{\mathcal{FK}}$ in such an order that

- $G_1 = F$, and
- for all $j \in [m-1]$ there is some $k \in [j]$ such that $S_k[l] \to S_{j+1} \in \mathcal{FK}$ for some integer l,

where it is to be assumed that for each $h \in [m]$, S_h is the relation name of G_h.

We show by induction that, for all $j \in [m]$, there exists a valuation θ_j over $\mathsf{vars}(q_j)$ such that $\theta_j(q_j) \subseteq \mathbf{db}$, where

$$q_j \coloneqq \left(q \setminus q_P^{\mathcal{FK}}\right) \cup \{G_1, \ldots, G_j\}.$$

For the base step suppose $j = 1$. Denote by P^c the set of positions $\{(R,i) \mid (R,i) \notin P, i \in [n]\}$. Concerning the positions over relation names appearing in $q_1 = \left(q \setminus q_P^{\mathcal{FK}}\right) \cup \{F\}$, let us make a few observations. First, we note that $P^c \subseteq P^{\mathsf{co}}_{\mathcal{FK}}$, because otherwise some position of P would belong to a cycle, contradicting condition (I). Second, every position of a relation name appearing in $q \setminus q_P^{\mathcal{FK}}$ belongs to $P^{\mathsf{co}}_{\mathcal{FK}}$ by definition. Third, it readily holds that $P \subseteq P_{\mathcal{FK}}$. We conclude that a position (T,k) of a relation name T that appears in q_1 belongs to $P_{\mathcal{FK}}$ if and only if it belongs to P. It follows by conditions (II)–(IV) that the positions of P are occupied in F by variables that are orphan in q_1. Clearly, we can extend θ_0 to these orphan variables to obtain θ_1 such that $\theta_0(F_P) = \theta_1(F)$. In particular, we obtain that $\theta_1(q_1) \subseteq \mathbf{db}$, where $q_1 = \left(q \setminus q_P^{\mathcal{FK}}\right) \cup \{F\}$.

For the induction step suppose $j \in [m-1]$. The induction claim is that $\theta_{j+1}(q_{j+1}) \subseteq \mathbf{db}$ for some valuation θ_{j+1} over $\mathsf{vars}(q_{j+1})$, given the induction hypothesis that there is a valuation θ_j over $\mathsf{vars}(q_j)$ such that $\theta_j(q_j) \subseteq \mathbf{db}$. Let $k \in [j]$ be such that $S_k[l] \to S_{j+1} \in \mathcal{FK}$ for some integer l. Assuming $G_k = S_k(\underline{s_1, \ldots, s_a}, s_{a+1} \ldots, s_b)$, we can write $G_{j+1} = S_{j+1}(\underline{s_l}, u_2, \ldots, u_c)$ since $\mathcal{FK}$ is about q. Since $\theta_j(G_j) = S_k(\underline{\theta_j(s_1), \ldots, \theta_j(s_a)}, \theta_j(s_{a+1}) \ldots, \theta_j(s_b)) \in \mathbf{db}$ and $\mathbf{db} \models S_k[l] \to S_{j+1}$, we find a fact $S_{j+1}(\underline{\theta_j(s_l)}, b_2, \ldots, b_c) \in \mathbf{db}$. Observe by condition (IV) that $u_2, \ldots, u_c$ are pairwise distinct variables. Hence $\theta_{j+1} \coloneqq \theta_j \cup \{(u_i, b_i)\}_{i=2}^{c}$ is a well-defined valuation over $\mathsf{vars}(q_{j+1})$ such that $\theta_{j+1}(q_{j+1}) \subseteq \mathbf{db}$, if we can establish the following claim.

Claim 1. *$U \cap \mathsf{vars}(q_j) = \emptyset$, for $U \coloneqq \{u_2, \ldots, u_c\}$.*

Proof of Claim 1. Let $P' \coloneqq \{(S_{j+1}, 2), \ldots, (S_{j+1}, c)\}$. Let us first turn attention to $q_1 = \left(q \setminus q_P^{\mathcal{FK}}\right) \cup \{F\}$. Recall that $P^{\mathsf{co}}_{\mathcal{FK}}$ contains P^c as well as the positions of relation names appearing in $q \setminus q_P^{\mathcal{FK}}$. Since $P' \subseteq P_{\mathcal{FK}}$, it follows by condition (III) that U does not contain any variable that appears in F at a position of P^c, nor does it contain any variable from $\mathsf{vars}(q \setminus q_P^{\mathcal{FK}})$. Furthermore, it follows by condition (IV) that U does not contain any variable that appears in F at a position of P. We thus obtain that $U \cap \mathsf{vars}(q_1) = \emptyset$.

For the sake of contradiction, suppose now the claim is false, i.e., $u_p \in \mathsf{vars}(q_j)$ for some $p \in \{2, \ldots, c\}$. Let $h \leq j$ be the smallest integer such that $u_p \in \mathsf{vars}(q_h)$. We may assume, by the previous paragraph, that $h > 1$. Suppose $G_h = S_h(\underline{v_1}, v_2 \ldots, v_d)$. By construction of the sequence $(G_1, \ldots, G_m)$, and since q is self-join free and $\mathcal{FK}$ is about q, the primary-key term v_1 of G_h must appear in $G_{h'}$ for some $h' < h$. By minimality of h, it must be that $u_p \neq v_1$ and, consequently, u_p occurs at a non-primary-key position in $S_h(\underline{v_1}, v_2 \ldots, v_d)$; i.e., $u_p = v_{p'}$ for some $p \in \{2, \ldots, d\}$. But then (S_h, p') and (S_{j+1}, p) are two distinct non-primary-key positions of $P_{\mathcal{FK}}$ that are occupied in q by the same variable, contradicting condition (IV). We conclude by contradiction that the claim holds. □

Having concluded the induction proof, we note that $\theta(q) \subseteq \mathbf{db}$ for $\theta := \theta_n$. This concludes the proof of Theorem 4.3. □

C PROOFS FOR SECTION 6

The following proof of Lemma 6.1 goes through for foreign keys that need not be unary. The following definition of (not necessarily unary) foreign keys is standard. Let R be a relation name with arity n, and S an atom with signature $[m, k]$. An *(unrestricted) foreign key* is an expression $R[j_1, j_2, \ldots, j_k] \to S$ with $j_1, j_2, \ldots, j_k$ distinct integers in $[n]$. Given a database instance $\mathbf{db}$, an R-fact $R(a_1, \ldots, a_n)$ in $\mathbf{db}$ is *dangling* with respect to this foreign key if $\mathbf{db}$ contains no S-fact $S(\underline{b_1, \ldots, b_k}, b_{k+1}, \ldots, b_n)$ such that $a_{j_1} = b_1$, $a_{j_2} = b_2$, …, $a_{j_{k-1}} = b_{k-1}$, and $a_{j_k} = b_k$.

Proof of Lemma 6.1. Suppose q has a cyclic attack graph. Then, by [22, Lemma 3.6], there are atoms F and G such that $F \overset{q}{\rightsquigarrow} G \overset{q}{\rightsquigarrow} F$. For two constants a and b, define the following valuation Θ^a_b over $\mathsf{vars}(q)$:

$$\Theta^a_b(x) = \begin{cases} a & \text{if } x \in F^{+,q} \setminus G^{+,q}, \\ b & \text{if } x \in G^{+,q} \setminus F^{+,q}, \\ \bot & \text{if } x \in F^{+,q} \cap G^{+,q}, \\ (a, b) & \text{if } x \in \mathsf{vars}(q) \setminus \left(F^{+,q} \cup G^{+,q}\right). \end{cases}$$

Let R, S be two sets of ordered pairs of constants. Define

$$\begin{aligned} \mathbf{db}_{R,S} := &\{\Theta^a_b(H) \mid H \in q \setminus \{F, G\}, (a, b) \in R \cup S\} \\ &\cup \{\Theta^a_b(F) \mid (a, b) \in R\} \\ &\cup \{\Theta^a_b(G) \mid (a, b) \in S\}. \end{aligned}$$

The following follows from the proof of [22, Lemma 4.3]:

- $\mathbf{db}_{R,S}$ is consistent with respect to primary keys in $q \setminus \{F, G\}$; and
- $\mathsf{CERTAINTY}(q, \mathcal{PK})$ is L-hard, and remains L-hard when inputs are restricted to database instances that are equal to $\mathbf{db}_{R,S}$ for binary relations R and S.

We claim that the following are equivalent for all binary relations R and S:

(1) $\mathbf{db}_{R,S}$ is a "no"-instance of $\mathsf{CERTAINTY}(q, \mathcal{PK})$; and
(2) $\mathbf{db}_{R,S}$ is a "no"-instance of $\mathsf{CERTAINTY}(q, \mathcal{PK} \cup \mathcal{FK})$.

$\boxed{1 \Longrightarrow 2}$ Let $\mathbf{r}$ be a repair of $\mathbf{db}_{R,S}$ with respect to $\mathcal{PK}$ such that $\mathbf{r} \not\models q$. Informally, we construct a repair $\mathbf{r}'$ of $\mathbf{db}_{R,S}$ with respect to $\mathcal{PK} \cup \mathcal{FK}$ by closing each dangling fact of $\mathbf{r}$ by a cycle that is long enough. Initialize $\mathbf{r}'$ as $\mathbf{r}$, and chase $\mathbf{r}'$ by the following rule: Whenever there is some fact $A \in \mathbf{r}'$ that is dangling with respect to some foreign key $H[\vec{j}] \to H'$ in $\mathcal{FK}$, pick constants a, b such that $A = \Theta^a_b(H)$,

(1) if $H' \in q \setminus \{F, G\}$, then add $\Theta^a_b(H')$ to $\mathbf{r}'$;
(2) if $H' = F$, then add $\Theta^a_c(F)$ to $\mathbf{r}'$, where c is a fresh constant; and
(3) if $H' = G$, then add $\Theta^c_b(G)$ to $\mathbf{r}'$, where c is a fresh constant.

We only make one exception to this rule. Suppose that, according to (3), we should add to $\mathbf{r}'$ a G-fact, say $\Theta^e_d(G)$ with e a fresh constant, while having already added $\Theta^a_b(F)$, $\Theta^c_b(G)$, and $\Theta^c_d(F)$. Then, instead of introducing a fresh value, we add $\Theta^a_d(G)$. We deal symmetrically with additions of F-facts. It is now easy to see that the chase terminates, and that $\mathbf{r}'$ is a repair with respect to $\mathcal{PK} \cup \mathcal{FK}$.

Assume for the sake of contradiction that $\mu(q) \subseteq \mathbf{r}'$ for some valuation μ. The attacks between F and G imply that $\{\Theta^a_b(F), \Theta^{a'}_{b'}(G)\} \subseteq \mu(q)$ if and only if $a = a'$ and $b = b'$. Thus no added F-fact or G-fact is in $\mu(q)$, and hence we find constants a, b such that $\{\Theta^a_b(F), \Theta^a_b(G)\} \subseteq \mu(q) \cap \mathbf{r}$. Moreover, $\mathbf{db}_{R,S}$ is consistent with respect to primary keys in $q \setminus \{F, G\}$, and thus by construction, $\Theta^a_b(q \setminus \{F, G\}) \subseteq \mathbf{r}$. We obtain $\Theta^a_b(q) \subseteq \mathbf{r}$, hence $\mathbf{r} \models q$, a contradiction. We conclude by contradiction that $\mathbf{r}'$ does not satisfy q.

$\boxed{2 \Longrightarrow 1}$ Let $\mathbf{r}$ be a repair of $\mathbf{db}_{R,S}$ with respect to $\mathcal{PK} \cup \mathcal{FK}$ such that $\mathbf{r} \not\models q$. Note that $\mathbf{r}$ need not be a repair of $\mathbf{db}_{R,S}$ with respect to $\mathcal{PK}$, because

- some facts of $\mathbf{r}$ may not belong to $\mathbf{db}_{R,S}$; or
- some blocks of $\mathbf{db}_{R,S}$ may be disjoint with $\mathbf{r}$.

Let $\mathbf{s}$ be a $\subseteq$-minimal database instance such that

- $\mathbf{r} \cap \mathbf{db}_{R,S} \subseteq \mathbf{s}$; and
- for every block $\mathbf{blk}$ of $\mathbf{db}_{R,S}$ such that $\mathbf{r} \cap \mathbf{db}_{R,S} = \emptyset$, $\mathbf{s}$ contains a fact arbitrarily picked from $\mathbf{blk}$.

By construction, $\mathbf{s} \subseteq \mathbf{db}_{R,S}$. It is easily verified that $\mathbf{s}$ is a repair of $\mathbf{db}_{R,S}$ with respect to $\mathcal{PK}$. Note incidentally that since $\mathbf{s} \preceq_{\mathbf{db}_{R,S}} \mathbf{r}$ is easily verified, it must hold that either $\mathbf{s} = \mathbf{r}$ or $\mathbf{s} \not\models \mathcal{FK}$.

It suffices to show that $\mathbf{s}$ falsifies q. Suppose for the sake of contradiction that $\mathbf{s} \models q$. Then, we can assume a valuation θ such that $\theta(q) \subseteq \mathbf{s}$, and therefore $\theta(q) \subseteq \mathbf{db}_{R,S}$. Since $\theta(q) \not\subseteq \mathbf{r}$, there is a fact $A \in \theta(q) \setminus \mathbf{r}$. Moreover, from the construction of $\mathbf{s}$, it follows that $\mathbf{r} \cap \mathbf{db}_{R,S}$ contains no fact that is key-equal to A. Then, by Lemma A.2, $\mathbf{r}$ is not a repair, a contradiction. □

D PROOFS FOR SECTION 8

We start with a helping lemma.

Lemma D.1. *Let q be query in* sjfBCQ*, and $\mathcal{FK}$ a set of foreign keys about q. Assume that every foreign key in $\mathcal{FK}$ is strong. Let $R[i] \to S$ be a foreign key in $\mathcal{FK}$, where S is obedient over $\mathcal{FK}$ and q. Assume that $q^{\mathcal{FK}}_S = \{S\}$. Assume that at least one of the following properties holds:*

(1) The attack graph of q is acyclic, and $\mathsf{key}(F) \neq \emptyset$ for every $F \in q$.
(2) R is obedient over $\mathcal{FK}$ and q.

Let $q_0 = q \setminus \{S\}$ and $\mathcal{FK}_0 = \mathcal{FK} \setminus \{R[i] \to S\}$. Suppose $(q, \mathcal{FK})$ has no block-interference. Then $\mathcal{FK}_0$ is about q_0, and $(q_0, \mathcal{FK}_0)$ has no block-interference.

Lemma D.2 assumes that $\mathcal{FK}$ is closed under logical implication. Under this assumption, it is not sufficient to remove one weak foreign key σ at a time, because it may be that $\mathcal{FK}^* \setminus \{\sigma\} \equiv \mathcal{FK}^*$. Instead, all weak foreign keys referencing a same relation name are removed at once.

Lemma D.2 ($\overset{\text{weak}}{\rightarrow}$ removal). *Let $\mathcal{FK}$ be a set of foreign keys such that $\mathcal{FK}^* = \mathcal{FK}$. Let $\mathcal{FK}_{\text{weak}}$ be the set of weak foreign keys in $\mathcal{FK}$. Assume that some non-trivial foreign key in $\mathcal{FK}_{\text{weak}}$ references S, and let $\mathcal{FK}_0 = \mathcal{FK} \setminus \mathcal{FK}_{\text{weak}}[\rightarrow S]$. Let q be a query in* sjfBCQ *such that $\mathcal{FK}$ is about q. Then, $\mathcal{FK}_0$ is about q, and the following hold:*

- $\mathsf{CERTAINTY}(q, \mathcal{FK}) \leq_m^{\mathsf{FO}} \mathsf{CERTAINTY}(q, \mathcal{FK}_0)$*; and*
- *if $(q, \mathcal{FK})$ has no block-interference, then $(q, \mathcal{FK}_0)$ has no block-interference.*

Lemma D.3 ($\mathsf{o} \overset{\text{str}}{\rightarrow} \mathsf{o}$ removal). *Let q be query in* sjfBCQ*, and $\mathcal{FK}$ a set of foreign keys about q. Let $R[i] \rightarrow S$ be a strong foreign key of type $\mathsf{o} \overset{\text{str}}{\rightarrow} \mathsf{o}$ in $\mathcal{FK}$. Assume that $q^{\mathcal{FK}}_S = \{S\}$. Following Lemma D.2, assume that every foreign key in $\mathcal{FK}$ is strong. Let $q_0 = q \setminus \{S\}$ and $\mathcal{FK}_0 = \mathcal{FK} \setminus \{R[i] \rightarrow S\}$. Suppose $(q, \mathcal{FK})$ has no block-interference. Then, $\mathcal{FK}_0$ is about q_0, and the following hold:*

- $\mathsf{CERTAINTY}(q, \mathcal{FK}) \leq_m^{\mathsf{FO}} \mathsf{CERTAINTY}(q_0, \mathcal{FK}_0)$*;*
- *$(q_0, \mathcal{FK}_0)$ has no block-interference; and*
- *if the attack graph of q is acyclic, then the attack graph of q_0 is acyclic.*

Lemma D.4 ($\mathsf{d} \overset{\text{str}}{\rightarrow} \mathsf{d}$ removal). *Let q be query in* sjfBCQ*, and $\mathcal{FK}$ a set of foreign keys about q. Let $R[i] \rightarrow S$ be a strong foreign key of type $\mathsf{d} \overset{\text{str}}{\rightarrow} \mathsf{d}$ in $\mathcal{FK}$. Following Lemma D.2, assume that every foreign key in $\mathcal{FK}$ is strong. Let $\mathcal{FK}_0 = \mathcal{FK} \setminus \{R[i] \rightarrow S\}$. Then, $\mathcal{FK}_0$ is about q, and the following hold:*

- $\mathsf{CERTAINTY}(q, \mathcal{FK}) \leq_m^{\mathsf{FO}} \mathsf{CERTAINTY}(q, \mathcal{FK}_0)$*; and*
- *if $(q, \mathcal{FK})$ has no block-interference, then $(q, \mathcal{FK}_0)$ has no block-interference.*

Finally, we have two lemmas for removing strong foreign keys of type $\mathsf{d} \overset{\text{str}}{\rightarrow} \mathsf{o}$. Lemma D.5 deals with queries q such that $\mathsf{vars}(F) \neq \emptyset$ for every $F \in q$. Lemma D.6 deals with queries containing an atom F with $\mathsf{vars}(F) = \emptyset$.

Lemma D.5 ($\mathsf{d} \overset{\text{str}}{\rightarrow} \mathsf{o}$ removal). *Let q be query in* sjfBCQ*, and $\mathcal{FK}$ a set of foreign keys about q. Following Lemmas D.2, D.3, and D.4, assume that all foreign keys in $\mathcal{FK}$ are strong and of type $\mathsf{d} \overset{\text{str}}{\rightarrow} \mathsf{o}$. Assume the following:*

(1) for every $F \in q$, $\mathsf{key}(F) \neq \emptyset$;
(2) $(q, \mathcal{FK})$ has no block-interference; and
(3) the attack graph of q is acyclic.

Let $N[i] \rightarrow O$ belong to $\mathcal{FK}$ (and therefore, by our previous assumption, $q^{\mathcal{FK}}_O = \{O\}$). Let $q_0 = q \setminus \{O\}$ and $\mathcal{FK}_0 = \mathcal{FK} \setminus \{N[i] \rightarrow O\}$. Then, $\mathcal{FK}_0$ is about q_0, and the following hold:

- $\mathsf{CERTAINTY}(q, \mathcal{FK}) \leq_m^{\mathsf{FO}} \mathsf{CERTAINTY}(q_0, \mathcal{FK}_0)$*;*
- *$(q_0, \mathcal{FK}_0)$ has no block-interference; and*
- *the attack graph of q_0 is acyclic.*

Proof sketch of Lemma D.5. From Lemma D.1, it follows that $\mathcal{FK}_0$ is about q_0 and that the second item holds. Since $\mathcal{FK}_0$ is about q_0, it readily follows that

$$\mathcal{FK}[\rightarrow O] = \{N[i] \rightarrow O\} \text{ and } \mathcal{FK}[O \rightarrow] = \emptyset.$$

That is, $\mathcal{FK}$ contains only one foreign key in which O occurs, and that foreign key is "incoming" in O.

With some effort, it can be shown that the N-atom in q is of the form $N(\vec{t}, y_{k+1}, \ldots, y_n)$ where y_i is a variable and $\{y_{k+1}, \ldots, y_n\} \setminus \{y_i\}$ is a set of orphan variables. Only the ith position of N can have outgoing foreign keys. Therefore, we can write

$$\mathcal{FK}[N \rightarrow] = \{N[i] \rightarrow O_1, \ldots, N[i] \rightarrow O_m\}.$$

Moreover, it will be the case that no variable of $\vec{t}$ is connected to y_i in the query q' defined by

$$q' := q \setminus \{N(\vec{t}, y_{k+1}, \ldots, y_n)\}. \tag{4}$$

The third item has an easy proof. We now sketch a proof for the first item.

Let $\mathbf{db}$ be a database instance that is input to $\mathsf{CERTAINTY}(q, \mathcal{FK})$. We define $\mathbf{db}_0$ as the smallest database instance satisfying the following two conditions:

- for every relation name R that occurs in q such that $R \notin \{N, O\}$, $\mathbf{db}_0$ contains all R-facts of $\mathbf{db}$; and
- *Relevance restriction:* for the relation name N, $\mathbf{db}_0$ includes all (and only) those N-blocks of $\mathbf{db}$ that contain at least one fact that is not dangling with respect to $\mathcal{FK}[N \rightarrow]$.

Clearly, $\mathbf{db}_0 \subseteq \mathbf{db}$. The following claim has an easy proof.

Claim 2. *Every repair of $\mathbf{db}$ with respect to foreign keys in $\mathcal{FK}$ and primary keys contains an N-fact from every N-block of $\mathbf{db}_0$.*

It suffices to show the following:

(A) if $\mathbf{db}$ is a "no"-instance of $\mathsf{CERTAINTY}(q, \mathcal{FK})$, then $\mathbf{db}_0$ is a "no"-instance of $\mathsf{CERTAINTY}(q \setminus \{O\}, \mathcal{FK} \setminus \{N[i] \rightarrow O\})$;
(B) the converse of (A).

Proof of (A) Assume that $\mathbf{db}$ is a "no"-instance of the problem $\mathsf{CERTAINTY}(q, \mathcal{FK})$. We can assume a repair $\mathbf{r}$ with respect to foreign keys in $\mathcal{FK}$ and primary keys such that $\mathbf{r} \not\models q$. We construct $\mathbf{r}_0$ from $\mathbf{r}$ by applying the following steps:

Deletion step 1: First, delete from $\mathbf{r}$ all N-facts that are not in $\mathbf{db}_0$, and delete all O-facts.

Deletion step 2: Then, for each $N[i] \rightarrow O'$ in $\mathcal{FK}[N \rightarrow]$, delete all O'-facts of $\mathbf{r} \setminus \mathbf{db}$ that are no longer referenced by an N-fact.

Since $\mathcal{FK}[\rightarrow N] = \emptyset$, it follows that *Deletion step 1* does not introduce dangling facts. Regarding *Deletion step 2*, observe that $\mathcal{FK} \setminus \{N[i] \rightarrow O'\}$ is about $q \setminus \{O'\}$ by Lemma D.1. Consequently, $\mathcal{FK}[\rightarrow O'] = \{N[i] \rightarrow O'\}$, wherefore *Deletion step 2* does not introduce dangling facts. We conclude that $\mathbf{r}_0$ satisfies foreign keys in $\mathcal{FK} \setminus \{N[i] \rightarrow O\}$ and primary keys. By Claim 2, $\mathbf{r}_0$ contains an N-fact from every N-block in $\mathbf{db}_0$. By construction, $\mathbf{r} \cap \mathbf{db}_0 \subseteq \mathbf{r}_0 \subseteq \mathbf{r}$. It can be easily verified that $\mathbf{r}_0 \not\models q \setminus \{O\}$.

Let $\mathbf{r}_0^*$ be a database instance, consistent with respect to foreign keys in $\mathcal{FK} \setminus \{N[i] \rightarrow O\}$ and primary keys, such that and $\mathbf{r}_0^* \preceq_{\mathbf{db}_0} \mathbf{r}_0$. That is,

$$\mathbf{r}_0 \cap \mathbf{db}_0 \subseteq \mathbf{r}_0^* \tag{5}$$

$$\mathbf{r}_0^* \subseteq \mathbf{db}_0 \cup \mathbf{r}_0 \subseteq \mathbf{db} \cup \mathbf{r} \tag{6}$$

It suffices to show $\mathbf{r}_0^* \not\models q \setminus \{O\}$. Assume for the sake of contradiction that there is a valuation θ over $\mathrm{vars}(q \setminus \{O\})$ such that $\theta(q \setminus \{O\}) \subseteq \mathbf{r}_0^*$. By (6), $\theta(q \setminus \{O\}) \subseteq \mathbf{db} \cup \mathbf{r}$. Since $\mathbf{r} \models N[i] \to O$ and since the O-atom is obedient, θ can be extended to a valuation θ^+ over $\mathrm{vars}(q)$ such that $\theta^+(q) \subseteq \mathbf{db} \cup \mathbf{r}$.

Since $\mathbf{r}_0 \not\models q \setminus \{O\}$, there must be a fact $A \in \theta(q \setminus \{O\})$ such that $A \notin \mathbf{r}_0$. By (6), $A \in \mathbf{db}_0$. Since, as argued before, $\mathbf{r}_0$ contains an N-fact of every N-block of $\mathbf{db}_0$, applying (5) it can be seen that A cannot be an N-fact. We now obtain that $A \in \theta^+(q) \setminus \mathbf{r}$. Moreover, if $\mathbf{r}_0$ contains an atom A' such that $A' \sim A$, then $A' \notin \mathbf{db}_0$ by (5). Hence, we also obtain that $\mathbf{r} \cap \mathbf{db}$ contains no fact that is key-equal to A.

By Lemma A.2, it is now correct to conclude that $\mathbf{r}$ is not a repair with respect to foreign keys in $\mathcal{FK}$ and primary keys, a contradiction.

Proof of (B) Assume that $\mathbf{db}_0$ is a "no"-instance of the problem $\mathsf{CERTAINTY}(q \setminus \{O\}, \mathcal{FK} \setminus \{N[i] \to O\})$. Among all repairs (with respect to foreign keys in $\mathcal{FK} \setminus \{N[i] \to O\}$ and primary keys) of $\mathbf{db}_0$ that falsify $q \setminus \{O\}$ (there is at least one such repair), let $\mathbf{r}_0$ be one that $\subseteq$-maximizes the set of N-facts that are not dangling in $\mathbf{db}$ with respect to $\mathcal{FK}[N \to]$. Recall that in moving from $\mathbf{db}$ to $\mathbf{db}_0$, an N-block is removed only if *all* its facts are dangling in $\mathbf{db}$ with respect to $\mathcal{FK}[N \to]$. Thus, $\mathbf{db}_0$ can contain N-facts that are dangling in $\mathbf{db}$ with respect to $\mathcal{FK}[N \to]$. It can be easily verified that $\mathbf{r}_0$ will contain a fact from every N-block in $\mathbf{db}_0$. The proof now constructs a repair of $\mathbf{db}$, called $\mathbf{r}$, that falsifies q.

We construct $\mathbf{r}$ from $\mathbf{r}_0$ by applying the following steps:

Insertion step 1: First, insert into $\mathbf{r}_0$ all O-facts of $\mathbf{db}$. Then, chase $\mathbf{r}_0$ with the foreign key $N[i] \to O$. That is, if there is a fact $N(\vec{a}, b_{k+1}, \ldots, b_n)$ that is dangling with respect to $N[i] \to O$, then insert $O(\underline{b_i}, \vec{c})$ for some sequence $\vec{c}$ of fresh constants.

Insertion step 2: Consider every N-block of $\mathbf{db}$ that is not in $\mathbf{db}_0$. If, due to the insertions in the previous step, one fact of such N-block is no longer dangling with respect to $\mathcal{FK}[N \to]$, then insert a fact from that block.

By construction, $\mathbf{r}$ is consistent with respect to foreign keys in $\mathcal{FK}$ and primary keys. The following claims finish the proof.

Claim 3. *$\mathbf{r}$ is a repair of $\mathbf{db}$ with respect to foreign keys in $\mathcal{FK}$ and primary keys.*

Claim 4. $\mathbf{r} \not\models q$.

Proof sketch of Claim 4. Assume towards a contradiction that there is a valuation θ over $\mathrm{vars}(q)$ such that $\theta(q) \subseteq \mathbf{r}$. Let the (unique) N-fact in $\theta(q)$ be $N(\vec{a}, b_{k+1}, \ldots, b_n)$. Since $\mathbf{r}_0 \not\models q \setminus \{O\}$, we observe that the fact $N(\vec{a}, b_{k+1}, \ldots, b_n)$ does not belong to $\mathbf{db}_0$ and was inserted in *Insertion step 2*. Thus, every fact in the block $N(\vec{a}, *)$ is dangling in $\mathbf{db}$ with respect to $\mathcal{FK}[N \to]$. Then, there is a fact $N(\vec{c}, p_{k+1}, \ldots, p_n) \in \mathbf{r}_0 \cap \mathbf{db}_0$ that is dangling in $\mathbf{db}$ with respect to $\mathcal{FK}[N \to]$ such that $b_i = p_i$. Informally, due to $N(\vec{c}, p_{k+1}, \ldots, p_n) \in \mathbf{r}_0$, we insert, in *Insertion step 1*, the invented fact $O(\underline{p_i})$ which in turn entails the insertion, in *Insertion step 2*, of $N(\vec{a}, b_{k+1}, \ldots, b_n)$.

By our choice of $\mathbf{r}_0$, there is a fact $N(\vec{c}, d_{k+1}, \ldots, d_n)$ that is not dangling in $\mathbf{db}$ with respect to $\mathcal{FK}[N \to]$, and a valuation μ over $\mathrm{vars}(q)$ such that

$$\mu(q \setminus \{O\}) \subseteq \big(\mathbf{r}_0 \setminus \{N(\underline{\vec{c}}, p_{k+1}, \ldots, p_n)\}\big) \cup \{N(\underline{\vec{c}}, d_{k+1}, \ldots, d_n)\}. \tag{7}$$

We define a valuation γ over $\mathrm{vars}(q \setminus \{O\})$ as follows. Let $\gamma(y_j) = p_j$ for $j \in \{k+1, \ldots, n\} \setminus \{i\}$. For every $u \in \mathrm{vars}(q \setminus \{O\})$ such that $u \notin \{y_{k+1}, \ldots, y_n\} \setminus \{y_i\}$, let

$$\gamma(u) = \begin{cases} \theta(u) & \text{if } u \text{ is connected to } y_i \text{ in } q', \\ \mu(u) & \text{otherwise,} \end{cases}$$

where q' is the query defined in Eq. (4).

Using $\theta(q) \subseteq \mathbf{r}$ and (7), it can be easily seen that $\gamma(q \setminus \{O, N\}) \subseteq \mathbf{r}_0$. It takes some more effort to show that $\gamma(N(\underline{\vec{t}}, y_{k+1}, \ldots, y_n)) = N(\underline{\vec{c}}, p_{k+1}, \ldots, p_n)$, using, among others, that no variable in $\vec{t}$ is connected to y_i in q' But then $\gamma(q \setminus \{O\}) \subseteq \mathbf{r}_0$, contradicting our assumption that $\mathbf{r}_0$ falsifies $q \setminus \{O\}$. □

The proof of Lemma D.5 is now concluded. □

Lemma D.6 (Variable-free keys). *Let q be query in* sjfBCQ*, and $\mathcal{FK}$ a set of foreign keys about q. Following Lemmas D.2, D.3, and D.4, assume that all foreign keys in $\mathcal{FK}$ are strong and of type* $\mathsf{d} \xrightarrow{\mathsf{str}} \mathsf{o}$. *Let N be an atom of q such that* $\mathrm{key}(N) = \emptyset$*, and let $\vec{x}$ be the variables of* $\mathrm{vars}(N)$*. Let b be an arbitrary constant, and $\vec{b} = \langle b, b, \ldots, b \rangle$, a sequence of the same length as $\vec{x}$. Let $q_0 = q \setminus q_N^{\mathcal{FK}}$ and $\mathcal{FK}_0 = \mathcal{FK}{\restriction_{q_0}}$. Then $\mathcal{FK}_0$ is about q_0, and the following hold:*

- $\mathsf{CERTAINTY}(q, \mathcal{FK}) \leq_m^{\mathsf{FO}} \mathsf{CERTAINTY}(q_{0[\vec{x} \to \vec{b}]}, \mathcal{FK}_0)$*;*
- *if $(q, \mathcal{FK})$ has no block-interference, then $(q_{0[\vec{x} \to \vec{b}]}, \mathcal{FK}_0)$ has no block-interference; and*
- *if the attack graph of q is acyclic, then the attack graph of $q_{0[\vec{x} \to \vec{b}]}$ is acyclic.*

We can now give a proof of Lemma 8.1.

Proof of Lemma 8.1. Assume that the attack graph of q is acyclic and $(q, \mathcal{FK})$ has no block-interference. We first repeatedly apply the reduction of Lemma D.2 to remove all weak foreign keys. Then we apply the reductions of Lemmas D.3 and D.4 to remove strong foreign keys of a type in $\{\mathsf{o} \xrightarrow{\mathsf{str}} \mathsf{o}, \mathsf{d} \xrightarrow{\mathsf{str}} \mathsf{d}\}$. Whenever the resulting query contains an atom F such that $\mathrm{key}(F) = \emptyset$, we apply Lemma D.6. Whenever every atom in the resulting query has a variable at some primary-key position, we apply Lemma D.5. Eventually, we have reduced to some problem $\mathsf{CERTAINTY}(q'', \mathcal{FK}'')$ with $\mathcal{FK}'' = \emptyset$, such that the attack graph of q'' is acyclic. The latter problem is known to be in FO. The desired result holds by induction on the number of reductions, since for every intermediate problem $\mathsf{CERTAINTY}(q', \mathcal{FK}')$, it holds that the attack graph of q' is acyclic and $(q', \mathcal{FK}')$ has no block-interference. □