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 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.

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

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 \( \subseteq \)-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 \( q(X) \), an answer \( \tilde{a} \) is called consistent if \( q(\tilde{a}) \) 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 \( F\mathcal{K}_0 := \{R[1] \rightarrow \text{DOCS}, R[2] \rightarrow \text{AUTHORS}\} \). 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.
There is one foreign-key violation: the fact \( R(d1,o3) \) is *dangling*, because \( o3 \) is not an existing ORCiD in the table \( \text{AUTHORS} \). There is also one primary-key violation, because there are two distinct tuples with ORCiD \( o1 \) in the table \( \text{AUTHORS} \). This database has an infinite number of repair

To repair the primary-key violation, we can delete the fact \( R(d1,o3) \), or insert a new fact \( \text{AUTHORS}(o3, f1, la) \), where \( f1 \) 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:

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

We note that \( q_0 \) satisfies the foreign keys in \( F^K_0 \) (when distinct variables are treated as distinct constants) and every relation name that occurs in \( F^K_0 \) also occurs in \( q_0 \), in which case we say that \( F^K_0 \) 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 \( \text{CERTAINTY}(q, F^K) \) as the following problem:

**Problem** \( \text{CERTAINTY}(q, F^K) \).

**Input:** A database instance \( db \).

**Question:** Is \( q \) true in every repair w.r.t. foreign keys in \( F^K \) and primary keys?

Obviously, if \( F^K = \emptyset \), then \( \text{CERTAINTY}(q, F^K) \) becomes the well studied problem \( \text{CERTAINTY}(q) \).

Of special interest is the case where \( \text{CERTAINTY}(q, F^K) \) is in the complexity class \( \text{FO} \), 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 \( F^K \) that is about \( q \), it can be decided whether or not \( \text{CERTAINTY}(q, F^K) \) is in \( \text{FO} \). Furthermore, if \( \text{CERTAINTY}(q, F^K) \) is in \( \text{FO} \), 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:

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

The third atom may look redundant in the latter query. However, \( F^K_0 \) is not about the shorter query \( \{ \text{DOCS}(\text{x, t, '2016'}, R(x, 'o1')) \}, \) in which \( R(x, 'o1') \) is dangling with respect to \( R[2] \rightarrow \text{AUTHORS} \).

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 \( \text{CERTAINTY}(q, F^K) \) is L-hard or NL-hard (and thus not in \( \text{FO} \)) under some conditions. Section 8 shows that if these conditions are not satisfied, then \( \text{CERTAINTY}(q, F^K) \) is in \( \text{FO} \). 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 \( \text{CERTAINTY}(q, F^K) \), 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 \( \text{CERTAINTY}(q) \) 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, \( \text{CERTAINTY}(q) \) is either in \( \text{FO} \), L-complete, or \( \text{coNP} \)-complete, and it is decidable, given \( q \), which case applies.

A few extensions beyond this trichotomy result are known. The complexity of \( \text{CERTAINTY}(q) \) 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 \( \text{CERTAINTY}(q) \) is in \( \text{FO} \) [24]. The complexity landscape of \( \text{CERTAINTY}(q) \) for path queries, a subclass of (not necessarily self-join-free) conjunctive queries, was settled in [21].
unions of conjunctive queries \( q \), Fontaine [10] established interesting relationships between \( \text{CERTAINTY}(q) \) and Bulatov’s dichotomy theorem for conservative CSP [4].

The counting variant of \( \text{CERTAINTY}(q) \), denoted \( \sharp \text{CERTAINTY}(q) \), asks to count the number of repairs that satisfy some Boolean query \( q \). For self-join-free conjunctive queries, \( \sharp \text{CERTAINTY}(q) \) exhibits a dichotomy between \( \mathbf{FP} \) and \( \mathbf{2P} \)-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 \( \text{CERTAINTY}(q) \) 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 \( R(t_1, \ldots, t_k, t_{k+1}, \ldots, t_n) \) is an \( R \)-atom (or simply atom).

If \( F \) is an atom, then \( \{F\} \) denotes the set of variables that occur in \( F \), and \( \text{key}(F) \) 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 \( A = B \), 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 \( \text{vars}(q) \) for the set of variables that occur in \( q \), and \( \text{const}(q) \) for the set of constants that occur in \( q \). If \( x \in \text{vars}(q) \) and \( c \) is a constant, then \( q_{[x \leftarrow c]} \) 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 \( \theta \) 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 \( db \models q \), if there is a valuation over \( \text{vars}(q) \) such that \( \theta(q) \subseteq db \). A block of \( db \) is a maximal subset of key-equal facts. If \( A \) is a fact in \( db \), then block \((A, db)\) denotes the block of \( db \) that contains \( A \). If \( A = R(\bar{a}, \bar{b}) \), then block \((A, db)\) is also denoted by \( R(\bar{a}, \bar{b}^+) \), and a fact in this block is said to be of the form \( R(\bar{a}, \bar{b}) \).

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 \( \text{CERTAINTY}(q) \) 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:

\[
\forall x_1, \ldots, x_n \forall y_{k+1} \ldots \forall y_n \left( R(x_1, \ldots, x_k, x_{k+1}, \ldots, y_n) \land R(x_1, \ldots, x_k, y_{k+1}, \ldots, y_n) \right) \rightarrow \left( \wedge_{i=k+1}^n y_i = y_i \right). \tag{1}
\]

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

The complexity classification of \( \text{CERTAINTY}(q) \) uses the notion of attack graph [22] recalled next. For a query \( q \) in sjfBCQ, we write \( \mathcal{K}(q) \) for the set \( \{\text{key}(F) \to \text{vars}(F) \mid F \in q\} \), which is a set of functional dependencies over \( \text{vars}(q) \). For an atom \( F \in q \), we define \( F^+ = \{x \in \text{vars}(q) \mid \mathcal{K}(q \setminus \{F\}) \models \text{key}(F) \to x\}. \) Informally, \( F^+ \) is the set of variables that are functionally dependent on \( F \) 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 \( F \rightarrow^q G \), if \( F \neq G \) and there exists a sequence of variables \( x_0, x_1, \ldots, x_n \), all belonging to \( \text{vars}(q) \setminus F^+ \), such that \( x_0 \in \text{vars}(F), x_n \in \text{vars}(G) \), 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 \( \text{CERTAINTY}(q) \) is in \( \mathcal{FO} \); otherwise \( \text{CERTAINTY}(q) \) is \( \mathcal{L} \)-hard.

\( \mathcal{FO} \) 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 \( R[i] \rightarrow S \) 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 \( R(a_1, \ldots, a_n) \in \text{db} \) is dangling (in \( \text{db} \)) with respect to this foreign key if \( \text{db} \) contains no \( S \)-fact \( S(b_1, b_2, \ldots, b_n) \) such that \( a_i = b_i \). A fact of \( \text{db} \) 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 \( \text{db} \) if \( \text{db} \) contains no dangling facts. Remark that foreign keys are unary by definition.
We write $\mathcal{F}_K^*$ for the set that contains every foreign key that is logically implied by $\mathcal{F}_K$ (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 $\mathcal{F}_K$ of foreign keys is a directed graph. There is a vertex $(R, i)$ whenever $R$ is a relation name that occurs in $\mathcal{F}_K$, say with signature $[n, k]$, and $i \in [n]$. Such a pair $(R, i)$ will be called a *position*.

Let $\mathcal{F}_K$ be a set of foreign keys. A $\oplus$-repair of $\text{db}$ with respect to $\mathcal{F}_K \cup \text{PK}$ (or repair for short) is a database instance $r$ such that: (i) $r$ satisfies $\mathcal{F}_K \cup \text{PK}$; and (ii) there is no database instance $s$ such that $s \preceq_{\text{db}} r$ and $s$ satisfies $\mathcal{F}_K \cup \text{PK}$. A subset-repair is a $\oplus$-repair $r$ where $s \preceq r$ satisfying $r \subseteq \text{db}$, and a superset-repair is a $\oplus$-repair $r$ satisfying $\text{db} \subseteq r$.

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

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 $q = \{R[x, y], S(y, z), T(z)\}$ and $\mathcal{F}_K = \{R[2] \rightarrow S, S[2] \rightarrow T\}$. Let $\text{db} = \{(q, b, S(b, c))\}$. Then the following are three $\oplus$-repairs:

$$
\begin{align*}
& r_1 = \emptyset, \\
& r_2 = \{R(q, b), S(b, 1), T(1)\}, \\
& r_3 = \{R(q, b), S(b, c), T(1)\}.
\end{align*}
$$

$r_1$ is a subset-repair, and $r_2$ is a superset-repair. It may be counter-intuitive that $r_3$ is not strictly $\oplus$-closer to $\text{db}$ than $r_2$. Note however:

$$
\text{db} @ r_2 = \{S(b, c), S(b, 1), T(1)\},
$$

$$
\text{db} @ r_3 = \{T(1)\}.
$$

Since the latter two sets are not comparable by $\subseteq$, we have that $r_2$ and $r_3$ are not comparable by $\preceq_{\text{db}}$. □

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

The following is relative to a fixed problem $\text{CERTAINTY}(q, \mathcal{F}_K)$. A *consistent first-order rewriting* is a closed first-order formula $\varphi$ such that a database instance is a "yes"-instance of the problem $\text{CERTAINTY}(q, \mathcal{F}_K)$ if and only if it satisfies $\varphi$. 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 $\text{CERTAINTY}(q, \mathcal{F}_K)$. Its definition is technical, but the following example should be helpful to convey the intuition.

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

$$
\begin{array}{|c|c|c|c|}
\hline
N & x & c & y \\
\hline
b_1 & c & 1 & \Box \\
\hline
b_2 & d & 2 & \Box \\
\hline
b_3 & c & 3 & \Box \\
\hline
b_4 & d & 4 & \Box \\
\hline
b_5 & c & n & \Box \\
\hline
b_n & d & n + 1 & \Box \\
\hline
\hline
O & y & \Box \\
\hline
1 & \Box \\
\hline
\end{array}
$$

\[1\text{Recall that } \mathcal{P}_K^e \text{ is the set of primary-key constraints, of the form (i), that can be derived from the relation names in } \text{db.}\]
Note that all N-facts, except the first one, are dangling. Our goal is to construct a \( r \)-repair \( r \) that falsifies \( q \). Such a \( r \)-repair must obviously choose \( N(b_1, d) \) in the first \( N \)-block, which implies that \( O(2) \) must be inserted. But then \( N(b_2, c) \) is no longer dangling, and, as a consequence, \( r \) must contain an \( N \)-fact from the second \( N \)-block. In order to falsify \( q \), \( r \) must choose \( N(b_2, d, 3) \) in the second block, which implies that \( O(3) \) must be inserted. By repeating the same reasoning, \( r \) must be as follows:

\[
\begin{array}{c|c|c|c}
N & x & c & y \\
\hline
b_1 & d & 2 & 1 \\
b_2 & d & 3 & 2 \\
\vdots & \vdots & \vdots & \vdots \\
b_n & d & n + 1 & n + 1 \\
b_{n+1} & \Box & n + 1 & 1 \\
\end{array}
\]

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

Informally, in deciding whether or not \( db \) is a "yes"-instance of CERTAINTY(\( q, FK \)), we had to start from the first \( N \)-block, then repeatedly move to the next \( N \)-block, and finally inspect the value of \( \Box \) in the last \( N \)-block. It is now unsurprising that CERTAINTY(\( q, FK \)) 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 \( N(x, c, y) \) 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, \( N(x, y, y) \) (and adapt \( db \) accordingly). On the other hand, block-interference disappears if we replace \( N(x, c, y) \) with \( N(x, z, y) \) in \( q \), where \( z \) is a fresh variable occurring only once.

- Block-interference will also disappear if we replace \( O(y) \) with \( O(y, c) \) or \( O(y, y) \) in the above example, because then the \( O \)-facts in \( r \setminus db \) can take the form \( O(i, \Box) \) for some fresh constant \( \Box \) which cannot be used for making the query true. On the other hand, if we replace \( O(y) \) with \( O(y, w) \) 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 \( FK \) a set of foreign keys about \( q \). Let \( F \) 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 \( q_F^{FK} \) as the smallest subset of \( q \) such that if the closure \( P_{FK} \) contains a position \( (S, i) \), then \( q_F^{FK} \) contains the \( S \)-atom of \( q \). We also write \( q_F^{FK} \) as a shorthand for \( q_F^{FK} \), where \( P_{FK} := \{(R, i) \mid i \in \{k + 1, \ldots, n\}\} \).

Let the \( R \)-atom of \( q \) be \( F = R(z, t_{k+1}, \ldots, t_n) \), and define \( F_p := R(z, u_{k+1}, \ldots, u_n) \) 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 \( FK \) and \( q \)) if

\[
q \setminus q_F^{FK} \cup \{F_p\} \models q, \tag{2}
\]

where it is to be noted that the logical entailment in the other direction holds vacuously true (and therefore we also get \( \equiv FK \)-equivalence). Furthermore, we say that atom \( F \) is obedient (over \( FK \) and \( q \)) if the set of positions \( \{(R, i) \mid i \in \{k + 1, \ldots, n\}\} \) is obedient (over \( FK \) and \( q \)). If \( FK \) 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 \( q = \{N(x, c, y), O(y)\} \) with \( FK = \{N[3] \rightarrow O\} \). We first argue that \( P_0 := \{(N, 2)\} \) is not obedient. We have \( q_{P_0}^{FK} = \{N(x, c, y)\} \), 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 \( \{N(x, w_2, y), O(y)\} \), which is not \( \equiv FK \)-equivalent to \( q \).

We next argue that \( P_1 := \{(N, 3)\} \) is obedient. We have \( q_{P_1}^{FK} = q \), 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 \( \{N(x, c, y_3)\} \). We have \( \{N(x, c, y_3)\} \equiv FK \{N(x, c, u_3), O(u_3)\} \), and the latter query is obviously \( \equiv FK \)-equivalent to \( q \).

Note finally that the atom \( O(y) \) 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 \( FK \) 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 \( FK \):

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

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

Informally, Theorem 4.3 implies that if a set \( P \) of positions is obedient, then \( P_{FK} \) 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.
We say that the pair \( x \) and \( j \) partition 4.5, we obtain block-interference by letting the (pairwise distinct) variables \( y_1, \ldots, y_15 \) occupying \( P_{FK} \). The polygon encloses \( q_p^{FK} \), the green boxes mark \( P_{FK} \).

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 \( G_V(q) \) for the undirected edge 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 \( FK \) a set of foreign keys about \( q \). Let \( N[j] \rightarrow O \) be a strong foreign key in \( FK \). Let \( N(t_1, \ldots, t_k, t_{k+1}, \ldots, t_n) \) and \( O(t_{j}, y) \) be atoms in \( q \) (since the foreign key is strong, \( j > k \)). Let \( V = \{v \in \text{vars}(q') \mid \mathcal{K}(q) \not\models \emptyset \rightarrow \{v\} \} \), where \( q' \coloneqq q \setminus \{N(t_1, \ldots, t_k, t_{k+1}, \ldots, t_n)\} \). We say that this foreign key is block-interfering \( \text{(in } q \text{)} \) if the following hold:

1. the atom \( O(t_j, y) \) is obedient;
2. \( t_j \) is a variable in \( V \) (thus \( \mathcal{K}(q) \not\models \emptyset \rightarrow \{t_j\} \)); 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 \( G_V(q') \).

We say that the pair \( (q, FK) \) has block-interference if some foreign key of \( FK \) 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 \( q = \{N(x, c, y), O(y)\} \) with \( FK = \{N[3] \rightarrow O\} \), where the atom \( O(y) \) is obedient. Following 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 \( q_0 = \{N'(x, y), O(y), T(x, y)\} \) and \( FK = \{N'[2] \rightarrow O\} \). 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 \( N(b_i, c, i) \) with two facts \( N'(b_i, i) \) and \( T(b_i, i) \), while every “falsifying” fact \( N(b_i, d, i + 1) \) is replaced with a single fact \( N'(b_i, i + 1) \) (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 \( R(a, x) \) (yielding a functional dependency \( \emptyset \rightarrow \{x\} \)), because no @-repair can contain both \( R(a, b_i) \) and \( R(a, c_j) \) with \( i \neq j \).

5 MAIN THEOREM

The following theorem refines Theorem 1.1 by adding the conditions to decide whether or not \( \text{CERTAINTY}(q, FK) \) is in FO. To show that a problem \( \text{CERTAINTY}(q, FK) \) 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 \( FK \) be a set of unary foreign keys about \( q \). Then,

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

Moreover, it can be decided, given \( q \) and \( FK \), 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

\[
FK = \{N[3] \rightarrow O\};
\]

\[
q_1 = \{N(x, u, y), O(y, w)\};
\]

\[
q_2 = \{N(x, c, y), O(y, w)\};
\]

\[
q_3 = \{N(x, c, y), O(y, c)\}.
\]

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

- \( \text{CERTAINTY}(q_1, FK) \) is in FO because \( N[3] \rightarrow O \) is not block-interfering in \( q_1 \), even though the atom \( O(y, w) \) 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 \( \text{CERTAINTY}(q_1, FK) \) is the query \( q_1 \) itself. Remarkably, this is different from the consistent first-order rewriting for \( \text{CERTAINTY}(q_1) \) (i.e., in the absence of foreign keys). To see
the difference, note that the following database instance is a "yes"-instance of CERTAINTY(q₁, F𝐾), but a "no"-instance of CERTAINTY(q₁).

\[
\begin{array}{ccc|c}
N & x & u & y & O \\
& c & 1 & a & y \\
& c & 2 & b & \ \\
\end{array}
\]

- CERTAINTY(q₂, F𝐾) is NL-hard, because N[3] → O is block-interfering in q₂. Informally, this is because the position (N, 2) is now occupied by a constant c and therefore not obedient.
- CERTAINTY(q₃, F𝐾) is again in FO, because N[3] → O is not block-interfering in q₂. 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(q₃, F𝐾) and CERTAINTY(q₃) 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, F𝐾) is L-hard if F𝐾 = ∅ 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 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 F𝐾 be a set of foreign keys about q. If q has a cyclic attack graph, then CERTAINTY(q, F𝐾) is L-hard.

For example, since the attack graph of q = \{R(x, y), S(y, x)\} is cyclic, CERTAINTY(q, F𝐾) is L-hard, for every F𝐾 that is a (possibly empty) subset of \{R[2] → S, S[2] → 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 F𝐾 be a set of unary foreign keys about q. If (q, F𝐾) has block-interference, then the problem CERTAINTY(q, F𝐾) is NL-hard.

Further, consider again the example with q = \{N(x, c, y), O(y)\} and F𝐾 = \{N[3] → O\}, elaborated in the beginning of Section 4, where it was argued that CERTAINTY(q, 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, 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.

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

\[
\begin{array}{ccc|c}
N & x & c & y & O \\
& s & d & 2 & y \\
& 2 & d & t & \ \\
\end{array}
\]

On the other hand, it can be easily verified that there would be no falsifying \(⊕\)-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, F𝐾) is NL-complete for q = \{N(x, c, y), O(y)\} and F𝐾 = \{N[2] → O\}.

**PROPOSITION 7.3.** CERTAINTY(q, F𝐾) is P-complete for q = \{N(x, c, y), O(y)\} and F𝐾 = \{N[3] → O\}.

A fine-grained complexity classification for all problems in the set \{CERTAINTY(q, F𝐾) | q ∈ sjfBCQ and 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 F𝐾 a set of unary foreign keys about q. If the attack graph of q is acyclic and (q, F𝐾) has no block-interference, then CERTAINTY(q, F𝐾) is in FO (and its consistent first-order rewriting can be effectively constructed).
We sketch how the previous lemma is proved. For two decision problems \( P_1 \) and \( P_2 \), we write \( P_1 \leq_{m}^{\text{FO}} P_2 \) 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 \( \text{sijfBCQ} \) such that \( q' \) has an acyclic attack graph and

\[
\text{CERTAINTY}(q, \mathcal{FK}) \leq_{m}^{\text{FO}} \text{CERTAINTY}(q', \emptyset). \tag{3}
\]

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 \( R[i] \rightarrow S \) is of a type in \([o \rightarrow o, d \rightarrow d]\), depending on whether the \( R \)-atom or \( S \)-atom are obedient (symbol \( o \)) or disobedient (symbol \( d \)). Note that there is no type \( o \rightarrow d \), 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 \( \text{weak} \).

Note in Definition 4.5 that only foreign keys of type \( d \rightarrow o \) 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 \( d \rightarrow o \).

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

\[
\exists y \left(N(z) \land O(y)\right) \land \forall y \left(N(z) \rightarrow P(y)\right).
\]

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 \( P(a) \) or \( P(b) \) turns it into a "no"-instance.

\[
\begin{array}{ccc}
N & c & y \\
O & c & a \\
& b & .
\end{array} \quad \begin{array}{ccc}
O & y & a \\
& b & .
\end{array} \quad \begin{array}{ccc}
P & y & a \\
& b & .
\end{array}
\]

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 \( \{\text{CERTAINTY}(q, \mathcal{FK}) \mid q \in \text{sijfBCQ} \text{ and } \mathcal{FK} \text{ is about } q\} \), 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 \( R(x, y, z), S(x, z, y) \), and foreign key \( R[1, 3] \rightarrow S \). 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 \( q = \{E(x, y)\} \) does not satisfy \( E[2] \rightarrow E \) (when \( x \) and \( y \) are treated as distinct constants).

**Proposition 9.1.** Let \( q = \{E(x, y)\} \) and \( \mathcal{FK} = \{E[2] \rightarrow E\} \). Then, \( \text{CERTAINTY}(q, \mathcal{FK}) \) 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 \( q' = \{E(x, y), E(y, x)\} \). 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, Leond Lifkin, and Andreas Pieris. 2018. An Operational Approach to Consistent Query Answering. In PODS. ACM, 239–251.

[8] Aklul 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, 2019), 233–246.

[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.
We write \( \text{db}|_q \) for the restriction of \( \text{db} \) to those facts whose relation name occurs in \( q \). We write \( \mathcal{F}K|_q \) for the set of those foreign keys in \( \mathcal{F}K \) that only use relation names in \( q \). Clearly, if \( \mathcal{F}K \) is about \( q \), then \( \mathcal{F}K|_q = q \).

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

**Lemma A.1.** Let \( \mathcal{P}K \cup \mathcal{F}K \) be a set of primary keys and foreign keys. Let \( \text{db} \) be a (possibly inconsistent) database instance, and let \( t \) be a @-repair of \( \text{db} \). Let \( s \) be a database instance such that \( s \subseteq r \cup \text{db} \) and \( s \models \mathcal{P}K \cup \mathcal{F}K \). For every fact \( A \in s \cap \text{db} \), there is a fact \( A' \in r \cap \text{db} \) such that \( A' \sim A \).

**Proof.** Let \( s \setminus r = \{A_1, A_2, \ldots, A_n\} \). Since \( s \subseteq r \cup \text{db} \), each \( A_i \) belongs to \( \text{db} \). Let \( t_0 = r \). For \( i = 1, 2, \ldots, n \),

1. if \( r \cap \text{db} \) contains a fact that is key-equivalent to \( A_i \), let \( t_i = t_{i-1} \);
2. if \( r \setminus \text{db} \) contains a fact \( A'_i \) that is key-equivalent to \( A_i \), let \( t_i := (t_{i-1} \cup \{A'_i\}) \cup \{A_i\} \);
3. if \( r \) contains no fact that is key-equivalent to \( A_i \), let \( t_i := t_{i-1} \cup \{A_i\} \).

Let \( t := t_n \). From \( r \models \mathcal{P}K \) and \( s \models \mathcal{P}K \), it follows that \( t \models \mathcal{P}K \) by construction.

We show that \( t \models \mathcal{F}K \). To this end, let \( R[1] \rightarrow S \) be a foreign key in \( \mathcal{F}K \), and let \( R(a_1, \ldots, a_n) \) be a fact in \( t \). If \( R(a_1, \ldots, a_n) \in s \), then this foreign key is saturated by \( t \) because \( r \models \mathcal{F}K \) and, by construction, every fact in \( r \) is key-equivalent to a fact in \( t \). Assume next that \( R(a_1, \ldots, a_n) \in s \setminus r \). Since \( R(a_1, \ldots, a_n) \in s \) and \( s \models \mathcal{F}K \), \( s \) contains a fact \( S(a_{i_\sim}) \). By construction, \( t \) will contain a fact that is key-equivalent to \( S(a_{i_\sim}) \).

By construction, \( r \cap \text{db} \subseteq t \) and \( t \subseteq r \cup \text{db} \). It follows that \( t \models \mathcal{db} \). If (2) or (3) are applied once or more, then \( t <_\mathcal{db} r \), contradicting that \( r \) is a @-repair. It follows that only (1) applies, which means that for every \( A \in s \cap \text{db} \), \( r \) contains a fact of block \((A, \text{db})\).

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

1. \( \theta(q) \subseteq \mathcal{P}K \cup \mathcal{F}K \); and
2. there is a fact \( A \in \theta(q) \setminus r \) such that \( r \cap \text{db} \) contains no fact that is key-equivalent to \( A \).

Then \( r \) is not a @-repair.

**Proof.** Since \( q \models \mathcal{F}K \) and since \( q \in \text{sjfBCQ} \), we have \( \theta(q) \models \mathcal{P}K \cup \mathcal{F}K \). Assume towards a contradiction that \( r \) is a @-repair. By Lemma A.1, for every fact \( A \in \theta(q) \cap \text{db} \), there is a fact \( A' \in r \cap \text{db} \) 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 $db$, define
\[
\text{keyconst}(db) := \text{adom}(\{R'(\overline{a}) \mid \exists \overline{b} : R(\overline{a}, \overline{b}) \in db\});
\]
in words, keyconst(db) is the set of constants that appear at a primary-key position in db.

**Lemma B.1.** Let $q$ be a query in sfjBQC, and let $\mathcal{F}'\mathcal{K}$ be a set of unary foreign keys that is about $q$. Let $db$ be a database instance, and let $A = R(\overline{a}, b_{k+1}, \ldots, b_n) \in db$. 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 $h_1$ is an orphan constant of $db$ that does not belong to $\text{const}(q)$, for all $(R, i) \in P$. Then, there exists a database instance $db_{A,P}$ such that

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

**Example B.2.** Let $q = \{N(\langle x, y \rangle), O(\langle x, y \rangle)\}$ and $\mathcal{F}'\mathcal{K} = \{N[2] \rightarrow N, N[2] \rightarrow O\}$. Consider the following database instance $db$:
\[
\begin{align*}
\begin{array}{ccc}
\text{db} = N & x & x \\
& a & a \\
& b & c \\
\end{array}
\begin{array}{ccc}
O & x & y \\
& a & b \\
& c & \bot \\
\end{array}
\end{align*}
\]
Select $A = N(b, c), P = \{(N, 2)\}$, 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 $db_{A,P}$ satisfying all the items of the lemma statement:
\[
\begin{align*}
\begin{array}{ccc}
\text{db}_{A,P} = N & x & x \\
& a & a \\
& b & c \\
\end{array}
\begin{array}{ccc}
O & x & y \\
& a & b \\
& c & \bot \\
\end{array}
\end{align*}
\]
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 $F = R(\overline{x}, t_{k+1}, \ldots, t_n)$, and define
\[
q' := (q \setminus \text{vars}(F_p)) \cup \{F_p\},
\]
where $F_p$ is obtained from $F$ by substituting fresh variables for the terms occurring at positions of $P$ (see Definition 4.1).

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

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., $q' \models q$.

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

Let $\{G_1, \ldots, G_m\}$ list the atoms of $q'p_{\mathcal{F}'\mathcal{K}}$ in such an order that

- $G_1 = F$, and
- for all $j \in [m - 1]$ there is some $k \in [j]$ such that $S_k[l] \rightarrow S_{j+1} \in \mathcal{F}'\mathcal{K}$ 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 $\overline{\theta}_j$ over $\text{vars}(q_j)$ such that $\overline{\theta}_j(q_j) \subseteq db$, where
\[
q_j := \left(q \setminus \text{vars}(\mathcal{F}'\mathcal{K})\right) \cup \{G_1, \ldots, G_j\}.
\]

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 $q_1$ = $(q \setminus \text{vars}(\mathcal{F}'\mathcal{K})) \cup \{F\}$, let us make a few observations. First, we note that $P^c \subseteq P_{\mathcal{F}'\mathcal{K}}$, because otherwise some position of $P$ would belong to a cycle, contradicting condition (I).

Second, every position of a relation name appearing in $q \setminus \text{vars}(\mathcal{F}'\mathcal{K})$ belongs to $P_{\mathcal{F}'\mathcal{K}}$, by definition. Third, it readily holds that $P \subseteq P_{\mathcal{F}'\mathcal{K}}$.

We conclude that a position $(T, k)$ of a relation name $T$ that appears in $q_1$ belongs to $P_{\mathcal{F}'\mathcal{K}}$ if and only if it belongs to $P$. It follows by conditions (II)-(IV) that the positions of $P$ are obedient by variables that are orphan in $q_1$. Clearly, we can extend $\theta_0$ to these orphan variables to obtain $\theta_1$ such that $\theta_1(F) = \theta(F)$. In particular, we obtain that $\theta_1(q_1) \subseteq db$, where $q_1 = (q \setminus \text{vars}(\mathcal{F}'\mathcal{K})) \cup \{F\}$.

For the induction step suppose $j \in [m-1]$. The induction claim is that $\theta_{j+1}(q_{j+1}) \subseteq db$ for some valuation $\theta_{j+1}$ over $\text{vars}(q_{j+1})$, given the induction hypothesis that there is a valuation $\overline{\theta}_j$ over $\text{vars}(q_j)$ such that $\theta_j(q_j) \subseteq db$. Let $k \in [j]$ be such that $S_k[l] \rightarrow S_{j+1} \in \mathcal{F}'\mathcal{K}$ for some integer $l$. Assuming $G_k = S_k(s_1, s_2, \ldots, s_{k+1}, \ldots, s_j)$, we can write $G_{j+1} = S_{j+1}(s_1, s_2, \ldots, s_j)$ since $\mathcal{F}'\mathcal{K}$ is about $q$. Since $\theta_j(s_1) = S_k[l] \rightarrow S_{j+1}$, we find a fact $\theta_{j+1}(s_1, s_2, \ldots, s_j) \in db$ and $db \models S_k[l] \rightarrow S_{j+1}$. We observe by condition (IV) that $u_{j+1}, \ldots, u_n$ are pairwise distinct variables. Hence $\theta_{j+1} := \overline{\theta}_j \cup \{(u_i, b_i)\}_{i = j+1}^{n}$ is a well-defined valuation over $\text{vars}(q_{j+1})$ such that $\theta_{j+1}(q_{j+1}) \subseteq db$, if we can establish the following claim.

**Claim 1.** $U \cap \text{vars}(q_j) = \emptyset$, for $U := \{u_2, \ldots, u_n\}$.

**Proof of Claim 1.** Let $P' := \{(S_{j+1}, 2), \ldots, (S_{j+1}, c)\}$. Let us first turn attention to $q_1 = (q \setminus \text{vars}(\mathcal{F}'\mathcal{K})) \cup \{F\}$. Recall that $P_{\mathcal{F}'\mathcal{K}}$ contains $P^c$ as well as the positions of relation names appearing in $q \setminus \text{vars}(\mathcal{F}'\mathcal{K})$. Since $P' \subseteq P_{\mathcal{F}'\mathcal{K}}$, 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 $\text{vars}(q \setminus \text{vars}(\mathcal{F}'\mathcal{K}))$. 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 $U \cap \text{vars}(q_1) = \emptyset$. 446
For the sake of contradiction, suppose now the claim is false, i.e., $u_p \in \text{vars}(q_j)$ for some $p \in \{2, \ldots, c\}$. Let $h \leq j$ be the smallest integer such that $u_p \in \text{vars}(q_h)$. We may assume, by the previous paragraph, that $h > 1$. Suppose $G_h = S_h(v_1, v_2, \ldots, v_q)$. By construction of the sequence $(G_1, \ldots, G_m)$, and since $q$ is self-join free and $\mathcal{F}K$ 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 $u_p \notin v_1$ and, consequently, $u_p$ occurs at a non-primary-key position in $S_h(v_1, v_2, \ldots, v_q)$; i.e., $u_p = u_p'$ for some $p \in \{2, \ldots, d\}$. But then $(S_h, p^r)$ and $(S_{h+1}, p)$ are two distinct non-primary-key positions of $P_{\mathcal{F}K}$ 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 $\theta(q) \subseteq \text{db}$ for $\theta := \vartheta_n$. 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 $R[j_1, j_2, \ldots, j_k] \rightarrow S$ with $j_1,j_2,\ldots,j_k$ distinct integers in $[n]$. Given a database instance $\text{db}$, an $R$-fact $r(a_1, \ldots, a_n)$ in $\text{db}$ is dangling with respect to this foreign key if $\text{db}$ contains no $S$-fact $S(b_1, b_2, b_{k+1}, \ldots, b_n)$ such that $a_{j_1} = b_1, a_{j_2} = b_2, \ldots, a_{j_{k-1}} = b_{k-1},$ 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 $F \leadsto G \leadsto F$. For two constants $a$ and $b$, define the following valuation $\Theta^q_b$ over $\text{vars}(q)$:

$$\Theta^q_b(x) = \begin{cases} a & \text{if } x \in F^* \setminus G^* \setminus q, \\ b & \text{if } x \in G^* \setminus F^* \setminus q, \\ \bot & \text{if } x \in F^* \setminus G^* \setminus q, \\ (a,b) & \text{if } x \in \text{vars}(q) \setminus (F^* \cup G^* \cup q). \end{cases}$$

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

$$\text{db}_{RS} := \left\{ \Theta^q_b(H) \mid H \in q \setminus \{F, G\}, (a,b) \in R \cup S \right\} \cup \left\{ \Theta^q_b(F) \mid (a,b) \in R \right\} \cup \left\{ \Theta^q_b(G) \mid (a,b) \in S \right\}.$$ 

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

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

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

1. $\text{db}_{RS}$ is a "no"-instance of $\text{CERTAINTY}(q, \mathcal{P}K)$;

2. $\text{db}_{RS}$ is a "no"-instance of $\text{CERTAINTY}(q, \mathcal{P}K \cup \mathcal{F}K)$.

Let $r$ be a repair of $\text{db}_{RS}$ with respect to $\mathcal{P}K$ such that $r \not\models q$. Informally, we construct a repair $r'$ of $\text{db}_{RS}$ with respect to $\mathcal{P}K \cup \mathcal{F}K$ by closing each dangling fact of $r$ by a cycle that is long enough. Initialize $r'$ as $r$, and chase $r'$ by the following rule: Whenever there is some fact $A \in r'$ that is dangling with respect to some foreign key $H[j] \rightarrow H'$ in $\mathcal{F}K$, pick constants $a, b$ such that $A = \Theta^q_b(H)$.

1. if $H' \subseteq q \setminus \{F, G\}$, then add $\Theta^q_b(H')$ to $r'$;

2. if $H' = F$, then add $\Theta^q_b(F)$ to $r'$, where $c$ is a fresh constant; and

3. if $H' = G$, then add $\Theta^q_b(G)$ to $r'$, where $c$ is a fresh constant.

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

Assume for the sake of contradiction that $\text{CERTAINTY}(q) \subseteq \text{db}$ for some valuation $\mu$. The attacks between $F$ and $G$ imply that $\{\Theta^q_b(F), \Theta^q_b(G)\} \subseteq \mu(q)$ 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 $\{\Theta^q_b(F), \Theta^q_b(G)\} \subseteq \mu(q) \cap \mathcal{R}$. Moreover, $\text{db}_{RS}$ is consistent with respect to primary keys in $q \setminus \{F, G\}$, and thus by construction, $\Theta^q_b(q \setminus \{F, G\}) \subseteq \mathcal{R}$. We obtain $\text{CERTAINTY}(q) \subseteq \mathcal{R}$, hence $r \models q$, a contradiction. We conclude by contradiction that $r'$ does not satisfy $q$. □

D PROOFS FOR SECTION 8

We start with a helping lemma.

Lemma D.1. Let $q$ be query in $\text{sfBCQ}$, and $\mathcal{F}K$ a set of foreign keys about $q$. Assume that every foreign key in $\mathcal{F}K$ is strong. Let $R[\cdot] \rightarrow S$ be a foreign key in $\mathcal{F}K$, where $S$ is obedient over $\mathcal{F}K$ and $q$. Assume that $q_{\mathcal{F}K} = \{S\}$. Assume that at least one of the following properties holds:

1. The attack graph of $q$ is acyclic, and key($F$) $\neq \emptyset$ for every $F \in q$.

2. $R$ is obedient over $\mathcal{F}K$ and $q$.

Let $q_0 = q \setminus \{S\}$ and $\mathcal{F}K_0 = \mathcal{F}K \setminus \{R[i] \rightarrow S\}$. Suppose $(q, F\mathcal{K})$ has no block-interference. Then $\mathcal{F}K_0$ is about $q_0$, and $(q_0, F\mathcal{K}_0)$ 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 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 $\mathcal{FK}^0 = \mathcal{FK}$. 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 $\mathcal{FK}^S_0 = \mathcal{FK}^\text{weak} \setminus \{S\}$. Let $\mathcal{q}$ be a query in sjfBCQ such that $\mathcal{FK}$ is about $\mathcal{q}$. Then, $\mathcal{FK}_0$ is about $\mathcal{q}$, and the following hold:

- $\text{CERTAINTY}(\mathcal{q}, \mathcal{FK}) \leq I_0 \text{CERTAINTY}(\mathcal{q}, \mathcal{FK}_0)$; and
- if $(\mathcal{q}, \mathcal{FK})$ has no block-interference, then $(\mathcal{q}, \mathcal{FK}_0)$ has no block-interference.

**Lemma D.3 ($\overset{\text{str}}{\rightarrow}$ $\overset{\text{d}}{\rightarrow}$ removal).** Let $\mathcal{q}$ be a query in sjfBCQ, and $\mathcal{FK}$ a set of foreign keys about $\mathcal{q}$. Let $R[i] \rightarrow S$ be a strong foreign key of type $d \overset{\text{str}}{\rightarrow} o$ in $\mathcal{FK}$. Assume that $\mathcal{FK}^0 = \{S\}$. Following Lemma D.2, assume that every foreign key in $\mathcal{FK}$ is strong. Let $\mathcal{q}_0 = \{\mathcal{q}\} \setminus \{S\}$ and $\mathcal{FK}_0 = \mathcal{FK} \setminus \{R[i] \rightarrow S\}$. Suppose $(\mathcal{q}, \mathcal{FK})$ has no block-interference. Then, $\mathcal{FK}_0$ is about $\mathcal{q}_0$, and the following hold:

- $\text{CERTAINTY}(\mathcal{q}, \mathcal{FK}) \leq I_0 \text{CERTAINTY}(\mathcal{q}, \mathcal{FK}_0)$; and
- if $(\mathcal{q}, \mathcal{FK})$ has no block-interference, then $(\mathcal{q}, \mathcal{FK}_0)$ has no block-interference.

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

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

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

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

- $\text{CERTAINTY}(\mathcal{q}, \mathcal{FK}) \leq I_0 \text{CERTAINTY}(\mathcal{q}, \mathcal{FK}_0)$; and
- if $(\mathcal{q}, \mathcal{FK})$ has no block-interference, then $(\mathcal{q}, \mathcal{FK}_0)$ has no block-interference; and
- the attack graph of $\mathcal{q}_0$ is acyclic.

**Proof sketch of Lemma D.5.** From Lemma D.1, it follows that $\mathcal{FK}_0$ is about $\mathcal{q}_0$ and that the second item holds. Since $\mathcal{FK}_0$ is about $\mathcal{q}_0$, it readily follows that $\mathcal{FK}_0 \setminus \{\sigma\} \equiv \mathcal{FK}$. 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 $\mathcal{q}$ is of the form $N(y_1, y_2, \ldots, y_n)$ where $y_i$ is a variable and $(\{y_k, \ldots, y_n\} \rightarrow y_i)$ is a set of orphan variables. Only the $i$th position of $N$ can have outgoing foreign keys. Therefore, we can write $\mathcal{FK}[N[i] \rightarrow ] \equiv \{N[i] \rightarrow O_1, \ldots, N[i] \rightarrow O_m\}$.

Moreover, it will be the case that no variable of $\mathcal{q}'$ is connected to $y_i$ in the query $\mathcal{q}'$ defined by $\mathcal{q}' := \mathcal{q} \setminus \{N(y_1, y_2, \ldots, y_n)\}$.

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

Let $db$ be a database instance that is input to $\text{CERTAINTY}(\mathcal{q}, \mathcal{FK})$. We define $db_0$ as the smallest database instance satisfying the following two conditions:

- for every relation name $R$ that occurs in $\mathcal{q}$ such that $R \notin \{N, O\}$, $db_0$ contains all $R$-facts of $db$; and
- Relevance restriction: for the relation name $N$, $db_0$ includes all (and only) those $N$-blocks of $db$ that contain at least one fact that is not dangling with respect to $\mathcal{FK}[N \rightarrow ]$.

Clearly, $db_0 \subseteq db$. The following claim has an easy proof.

**Claim 2.** Every repair of $db$ with respect to foreign keys in $\mathcal{FK}$ and primary keys contains an $N$-fact from every $N$-block of $db_0$.

It suffices to show the following:

1. if $db$ is a “no”-instance of $\text{CERTAINTY}(\mathcal{q}, \mathcal{FK})$, then $db_0$ is a “no”-instance of $\text{CERTAINTY}(\mathcal{q} \setminus \{O\}, \mathcal{FK} \setminus \{N[i] \rightarrow O\})$;
2. the converse of (A).

**Proof of (A).** Assume that $db$ is a “no”-instance of the problem $\text{CERTAINTY}(\mathcal{q}, \mathcal{FK})$. We can assume a repair $r$ with respect to foreign keys in $\mathcal{FK}$ and primary keys such that $r \notin \mathcal{q}$. We construct $r_0$ from $r$ by applying the following steps:

**Deletion step 1:** First, delete from $r$ all $N$-facts that are not in $db_0$, 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 $r \cap db_0$ 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 $\mathcal{q} \setminus \{O'\}$ 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 $r_0$ satisfies foreign keys in $\mathcal{FK} \setminus \{N[i] \rightarrow O\}$ and primary keys. By Claim 2, $r_0$ contains an $N$-fact from every $N$-block of $db_0$. By construction, $r \cap db_0 \subseteq r_0 \subseteq r$. It can be easily verified that $r_0 \neq \mathcal{q} \setminus \{O\}$.

Let $r_0^* \in db_0$ be a database instance, consistent with respect to foreign keys in $\mathcal{FK} \setminus \{N[i] \rightarrow O\}$ and primary keys, such that and $r_0^* \leq db_0$.

That is,

$$r_0 \cap db_0 \subseteq r_0^* \quad \text{(5)}$$

$$r_0^* \leq db_0 \cup r_0 \subseteq db \cup r \quad \text{(6)}$$
It suffices to show $r^* \not\models q \setminus \{O\}$. Assume for the sake of contradiction that there is a valuation $\theta$ over $\text{vars}(q \setminus \{O\})$ such that $\theta(q \setminus \{O\}) \subseteq r^*$. By (6), $\theta(q \setminus \{O\}) \subseteq \mathcal{db} \cup r$. Since $r \models N[i] \rightarrow O$ and since the $O$-atom is obedient, $\theta$ can be extended to a valuation $\theta^*$ over $\text{vars}(q)$ such that $\theta^*(q) \subseteq \mathcal{db} \cup r$.

Since $r_0 \not\models q \setminus \{O\}$, there must be a set $A \subseteq \theta(q \setminus \{O\})$ such that $A \in \mathcal{db}_0$. Assume towards a contradiction that $A \not\models r_0$. By (6), $A \subseteq \mathcal{db}_0$. Since, as argued before, $r_0$ contains an $N$-fact of every $N$-block of $\mathcal{db}_0$, applying (5) it can be seen that $A$ cannot be an $N$-fact. We now obtain that $A \in \theta^*(q) \setminus r$. Moreover, if $r_0$ contains an atom $A'$ such that $A' \sim A$, then $A' \not\in \mathcal{db}_0$ by (5). Hence, we also obtain that $r \cap \mathcal{db}$ contains no fact that is key-equal to $A$.

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

Proof of (B) Assume that $\mathcal{db}_0$ is a "no"-instance of the problem CERTAINTY$(q \setminus \{O\}, \mathcal{FK} \setminus \{N[i] \rightarrow O\})$. Among all repairs (with respect to foreign keys in $\mathcal{FK} \setminus \{N[i] \rightarrow O\}$ and primary keys) of $\mathcal{db}_0$ that falsify $q \setminus \{O\}$ (there is at least one such repair), let $r_0$ be one that $\leq$-maximizes the set of $N$-facts that are not dangling in $\mathcal{db}$ with respect to $\mathcal{FK}$ on $\mathcal{N} \rightarrow \mathcal{r}$. Recall that in moving from $\mathcal{db}$ to $\mathcal{db}_0$, an $N$-block is removed only if all its facts are dangling in $\mathcal{db}$ with respect to $\mathcal{FK}$ on $\mathcal{N} \rightarrow \mathcal{r}$. Thus, $\mathcal{db}_0$ can contain $N$-facts that are dangling in $\mathcal{db}$ with respect to $\mathcal{FK}$ on $\mathcal{N} \rightarrow \mathcal{r}$. It can be easily verified that $r_0$ will contain a fact from every $N$-block in $\mathcal{db}_0$. The proof now constructs a repair of $\mathcal{db}$, called $r$, that falsifies $q$.

We construct $r$ from $r_0$ by applying the following steps:

Insertion step 1: First, insert into $r_0$ all $O$-facts of $\mathcal{db}$. Then, chase $r_0$ with the foreign key $N[i] \rightarrow O$. That is, if is there is a fact $N(\bar{a}, b_{k+1}, \ldots, b_n)$ that is dangling with respect to $N[i] \rightarrow O$, then insert $O(b_t, \bar{c})$ for some sequence $\bar{c}$ of fresh constants.

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

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

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

Claim 4. $r \not\models q$.

Proof sketch of Claim 4. Assume towards a contradiction that there is a valuation $\theta$ over vars$(q)$ such that $\theta(q) \subseteq r$. Let the (unique) $N$-fact in $\theta(q)$ be $N(\bar{a}, b_{k+1}, \ldots, b_n)$. Since $r_0 \not\models q \setminus \{O\}$, we observe that the fact $N(\bar{a}, b_{k+1}, \ldots, b_n)$ does not belong to $\mathcal{db}_0$ and was inserted in Insertion step 2. Thus, every fact in the block $N(\bar{a}, \bar{c})$ is dangling in $\mathcal{db}$ with respect to $\mathcal{FK}$ on $\mathcal{N} \rightarrow r$. Then, there is a fact $N(\bar{a}, \bar{c}, p_{k+1}, \ldots, p_n) \in r_0 \cap \mathcal{db}$ that is dangling in $\mathcal{db}$ with respect to $\mathcal{FK}$ on $\mathcal{N} \rightarrow r$ such that $b_t = p_i$. Informally, due to $N(\bar{a}, \bar{c}, p_{k+1}, \ldots, p_n) \in r_0$, we insert, in Insertion step 1, the invented fact $O(p_i)$ which in turn entails the insertion, in Insertion step 2, of $N(\bar{a}, b_{k+1}, \ldots, b_n)$. By our choice of $r_0$, there is a fact $N(\bar{a}, d_{k+1}, \ldots, d_n)$ that is not dangling in $\mathcal{db}$ with respect to $\mathcal{FK}$ on $\mathcal{N} \rightarrow r$, and a valuation $\mu$ over vars$(q)$ such that $\mu(q \setminus \{O\}) \subseteq r_0 \cap \{N(\bar{a}, d_{k+1}, \ldots, d_n)\}$.

We define a valuation $\gamma$ over vars$(q \setminus \{O\})$ as follows. Let $\gamma(y_j) = p_j$ for $j \in \{k+1, \ldots, n\} \setminus \{i\}$. For every $u \in$ vars$(q \setminus \{O\})$ such that $u \not\in \{y_{k+1}, \ldots, y_n\}$, 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 $\theta(q) \subseteq r$ and (7), it can be easily seen that $q \setminus \{O, N\} \subseteq r_0$. It takes some more effort to show that $\gamma \in \mathcal{db}_0$ using, among others, that no variable in $\mathcal{db}$ is connected to $y_i$ in $q'$. But then $\gamma(q \setminus \{O\}) \subseteq r_0$, contradicting our assumption that $r_0$ falsifies $q \setminus \{O\}$.

The proof of Lemma D.5 is now concluded.

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 of $\mathcal{a} \rightarrow \mathcal{d} \rightarrow \mathcal{o}$. Whenever the resulting query contains an atom $F$ such that $\text{key}(F) = \emptyset$, 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 CERTAINITY$(q''', \mathcal{FK''})$ with $\mathcal{FK''} = \emptyset$, such that the attack graph of $q'''$ is acyclic. The latter problem is known to be in $\text{FO}$. The desired result holds by induction on the number of reductions, since for every intermediate problem CERTAINITY$(q', \mathcal{FK'})$, it holds that the attack graph of $q'$ is acyclic and $(q', \mathcal{FK'})$ has no block-interference.