A Dichotomy in the Complexity of Consistent Query Answering for Queries with Two Atoms

Phokion G. Kolaitis, Enela Pema

Computer Science Department, University of California Santa Cruz, Santa Cruz, CA 95060, USA

Abstract

We establish a dichotomy in the complexity of computing the consistent answers of a Boolean conjunctive query with exactly two atoms and without self-joins. Specifically, we show that the problem of computing the consistent answers of such a query either is in P or it is coNP-complete. Moreover, we give an efficiently checkable criterion for determining which of these two possibilities holds for a given query.

Keywords:
databases, key constraints, database repairs, consistent query answering

1. Introduction

An *inconsistent* database, often also referred to as an *uncertain* database, is a database that violates one or more integrity constraints that the data are required to obey. Inconsistent databases arise for a variety of reasons (e.g., lack of support by the system at hand) and in a variety of settings (e.g., when integrating data from heterogeneous sources that obey mutually incompatible integrity constraints).

One way to cope with inconsistency is data cleaning: the inconsistent database is transformed to a cleansed version that satisfies the integrity constraints, and then the cleansed database is used to answer queries. Data cleaning, however, often entails making choices that are arbitrary as, typically, there is a large number of cleansed databases that arise from an inconsistent database. An alternative and more principled approach, introduced in [1], is consistent query answering. In this approach, the inconsistencies in the database are kept, but are handled at query time by considering all possible *repairs* of the inconsistent database. More precisely, if Σ is a set of integrity constraints, then a repair of an inconsistent database I is a consistent database r (i.e., $r \models \Sigma$) that differs from I in a "minimal" way. By definition, the *consistent answers* of a query q on a database I is the intersection $\bigcap\{q(r) : r \text{ is a repair of } I\}$. If q is a Boolean query, computing the consistent answers of q is the following decision problem, denoted by $\text{CERTAINTY}_\Sigma(q)$ or, simply, $\text{CERTAINTY}(q)$: given a database I, is $q(r)$ true on every repair r of I? The notion of consistent answers is closely related to the notion of the *certain answers* in data integration [2]. In fact, the consistent answers coincide with the certain answers when the set of *possible worlds* is taken to be the set of all repairs.

From now on, we assume that q is a Boolean conjunctive query and Σ is a set of key constraints with one key per each relation symbol in some fixed relational schema $\mathbf{R}$. In this case, $\text{CERTAINTY}(q)$ is always in coNP [3]. Depending on the keys and the query, however, the actual complexity of $\text{CERTAINTY}(q)$ may vary widely, as illustrated by the following examples in which the underlined variables indicate that the corresponding attributes form the key:

- If q_1 is the query $\exists x, y, z. R_1(\underline{x}, y) \wedge R_2(\underline{y}, z)$, then $\text{CERTAINTY}(q_1)$ is first-order expressible, that is to say, there is a first-order query q'_1 such that on every instance I, we have that q_1 is true on every repair of I if and only if $q'_1(I)$ is true [4, 5]. Hence, $\text{CERTAINTY}(q_1)$ is in P; actually, it is in the much lower complexity class AC^0.
- If q_2 is the query $\exists x, y. R_1(\underline{x}, y) \wedge R_2(\underline{y}, x)$, then $\text{CERTAINTY}(q_2)$ is in P, but it is not first-order expressible [5].
- If q_3 is the query $\exists x, x', y. R_1(\underline{x}, y) \wedge R_2(\underline{x'}, y)$, then $\text{CERTAINTY}(q_3)$ is coNP-complete [4].

How can these differences in complexity be explained? Also, is there an algorithm and, in particular, is there an efficient algorithm that can be used to pinpoint the exact complexity of computing $\text{CERTAINTY}(q)$? Major progress in this direction was made by Wijsen [6], who gave a necessary and sufficient condition for $\text{CERTAINTY}(q)$ to be first-order expressible, provided q is a Boolean acyclic conjunctive query without self-joins; moreover, this condi-

Email addresses: kolaitis@cs.ucsc.edu (Phokion G. Kolaitis), epema@cs.ucsc.edu (Enela Pema)

tion can be checked in quadratic time in the size of the query q. What can we say about the complexity of CERTAINTY(q), if CERTAINTY(q) is not first-order expressible? It has been conjectured (e.g., in [7]) that a *dichotomy theorem* holds for the complexity of CERTAINTY(q), namely, either CERTAINTY(q) is in P or CERTAINTY(q) is coNP-complete. To appreciate the point of this conjecture and the significance of a dichotomy theorem, recall that Ladner [8] has shown that if P $\neq$ NP, then there are decision problems that are in coNP, but are neither in P nor are coNP-complete; thus, the existence of a dichotomy theorem for a class of decision problems cannot be taken for granted a priori.

The three examples given earlier involve conjunctive queries with exactly two atoms; in fact, most of the concrete conjunctive queries analyzed in [4, 5] have exactly two atoms In this article, we establish a dichotomy theorem for CERTAINTY(q), where q is a Boolean conjunctive query with exactly two atoms and without self-joins and such that CERTAINTY(q) is not first-order expressible. Specifically, assume that q is such a query and let R_1 and R_2 be the two relation symbols occurring in q, let L be the set of variables shared by the two atoms of q, and let $key(R_i)$ be the set of variables that occur as key attributes of R_i, $i = 1, 2$. Our dichotomy theorem asserts that

- if $key(R_1) \cup key(R_2) \subseteq L$, then CERTAINTY($q$) is in P;
- if $key(R_1) \cup key(R_2) \nsubseteq L$, then CERTAINTY($q$) is coNP-complete.

When combined with Wijsen's necessary and sufficient condition for CERTAINTY(q) to be first-order expressible, our dichotomy theorem implies that the complexity of the consistent answers of Boolean conjunctive queries with exactly two atoms and without self-joins exhibits a *trichotomy*. Moreover, it yields a linear-time algorithm for determining, given such a query, which of the following three collectively exhaustive possibilities holds: CERTAINTY(q) is first-order expressible, or CERTAINTY(q) is in P but is not first-order expressible, or CERTAINTY(q) is coNP-complete.

2. Preliminaries

A *relational database schema* is a finite collection $\mathbf{R}$ of relation symbols, each with an associated arity. The *attributes* of a relation symbol R in $\mathbf{R}$ need not have names. Thus, if R is a n-ary relation symbol, then its attributes can be identified with the *positions* $1, \dots, n$, which means that the set Attr(R) of the attributes of R coincides with the set $\{1, \dots, n\}$.

If R is a relation symbol in $\mathbf{R}$ and I is an instance over $\mathbf{R}$, then R^I denotes the interpretation of R on I. A *fact* of an instance I is an expression of the form $R^I(a_1, \dots, a_n)$ such that $(a_1, \dots, a_n) \in R^I$; in this case, we will also say that $R^I(a_1, \dots, a_n)$ is an *R-fact* of the instance I. For simplicity of notation and whenever the instance I at hand is understood from the context, we will write $R(a_1, \dots, a_n)$, instead of $R^I(a_1, \dots, a_n)$. A *key* of R is a subset X of the set Attr(R) = $\{1, \dots, n\}$ of the attributes of R such that for every instance I over $\mathbf{R}$, the interpretation R^I of R on I does not contain two distinct facts that agree on all positions in X. In other words, a key is a set X of positions such that the functional dependency $X \rightarrow$ Attr(R) holds; such a dependency is called a *key constraint*. We assume that each relation symbol comes with a fixed key.

A *conjunctive query* is a first-order formula built from atomic formulas, conjunctions, and existential quantification. Thus, every conjunctive query is logically equivalent to an expression of the form $q(\mathbf{z}) = \exists \mathbf{w}.R_1(\mathbf{x_1}) \wedge ... \wedge R_m(\mathbf{x_m})$, where each $\mathbf{x}_i$ is a tuple of variables and constants, $\mathbf{z}$ and $\mathbf{w}$ are tuples of variables, and the variables in $\mathbf{x_1}, \dots, \mathbf{x_m}$ appear in exactly one of $\mathbf{z}$ and $\mathbf{w}$. A *Boolean* conjunctive query is a conjunctive query in which all variables are existentially quantified. A conjunctive query contains a *self-join* if it has repeated relation names. We refer to queries that do not contain self-joins as *self-join free* queries.

In what follows, whenever we write a conjunctive query, we underline in each atom variables and constants that appear in the positions of the key of the relation symbol; such variables are called *key variables*. For instance, by writing $\exists x, y.R_1(\underline{x}, y) \wedge R_2(\underline{y}, x)$, we indicate that the first position (attribute) of R_1 and the first position (attribute) of R_2 are, respectively, the key of R_1 and the key of R_2; furthermore, x is a key variable of the atom $R_1(\underline{x}, y)$, while y is a key variable of the atom $R_2(\underline{y}, x)$. In general, when a conjunctive query is presented in this form, we omit explicitly specifying the schema and the key constraints, since they can be derived from the formulation of the query itself.

Let q be a conjunctive query and $R(\underline{\mathbf{x}}, \mathbf{y})$ one of its atoms. We define $vars(R(\underline{\mathbf{x}}, \mathbf{y}))$ to be the set of variables appearing in the atom $R(\underline{\mathbf{x}}, \mathbf{y})$. We define $key(R(\underline{\mathbf{x}}, \mathbf{y}))$ to be the set of variables appearing in the positions of the key in the atom $R(\underline{\mathbf{x}}, \mathbf{y})$. Note that constants may occur in $\mathbf{x}$, but are not members of $key(R(\underline{\mathbf{x}}, \mathbf{y}))$; in particular, $key(R(\underline{\mathbf{x}}, \mathbf{y}))$ may be the empty set. We define $nkey(R(\underline{\mathbf{x}}, \mathbf{y}))$ to be the set of the variables appearing in positions that do not belong to the key in the atom $R(\underline{\mathbf{x}}, \mathbf{y})$. Note that it is possible to have that $key(R(\underline{\mathbf{x}}, \mathbf{y})) \cap nkey(R(\underline{\mathbf{x}}, \mathbf{y})) \neq \emptyset$. For simplicity, given a self-join free conjunctive query, we will refer to the atoms with the name of the corresponding relations. Thus, we write $vars(R)$ instead of $vars(R(\underline{\mathbf{x}}, \mathbf{y}))$, and $key(R)$ instead of $key(R(\underline{\mathbf{x}}, \mathbf{y}))$.

Next, we give precise definitions of the notions of a subset repair, consistent answers, and CERTAINTY(q).

Definition 1. *Let* $\mathbf{R}$ *be a relational database schema and* Σ *a set of integrity constraints over* $\mathbf{R}$.

- *Let I be an instance. An instance r is a* subset repair *or, simply, a* repair *of I w.r.t. Σ if r is a maximal sub-instance of I that satisfies Σ, i.e., $r \models \Sigma$ and there is no instance r' such that $r' \models \Sigma$ and $r \subset r' \subseteq I$.*
- *Let q be a query and I an instance. We say that a tuple t is a* consistent answer for *q if for every repair r of I w.r.t Σ, we have $t \in q(r)$.*
- *Let q be a Boolean query.*
  - *If I is an instance, then the notation $I \models_\Sigma q$ denotes that q is true in every repair of I w.r.t. Σ, whereas the notation $I \not\models_\Sigma q$ denotes that q is false in at least one repair of I w.r.t. Σ.*
  - CERTAINTY(q) *is the following decision problem: given an instance I, does $I \models_\Sigma q$?*

As mentioned in the Introduction, if Σ is a set of key constraints and q is a Boolean conjunctive query, then CERTAINTY(q) is in coNP.

3. FO-Expressibility of the Certain Answers of Acyclic Self-join Free Conjunctive Queries

This section contains an overview of the results in [6] for the first-order expressibility of CERTAINTY(q).

Definition 2. *Let q be conjunctive query.*

- *The* complete intersection graph *of q is a labeled graph that has the atoms of q as vertices, and an edge between every two distinct atoms F and G labeled by the set of variables that F and G share.*
- *An* intersection tree *for q is a spanning tree of the complete intersection graph of q.*
- *A* join tree *for q is an intersection tree that satisfies the following connectedness condition: whenever the same variable x occurs in two atoms F and G, then x occurs in every atom on the unique path linking F and G.*
- *We say that q is* acyclic *if it has a join tree.*

Wijsen [6] found a necessary and sufficient condition for the first-order expressibility of CERTAINTY(q). This condition involves the notion of the *attack graph*, which we will present after introducing some notation. Every atom F in a conjunctive query q gives rise to a functional dependency among the variables occurring in F. For example, the atom $R(\underline{x, y}, z)$ gives rise to $\{x, y\} \rightarrow z$. As a special case, the atom $R(\underline{a}, x)$, where a is a constant, gives rise to $\{\} \rightarrow x$.

Let q be a Boolean conjunctive query.

- We write $K(q)$ to denote the set of all functional dependencies that arise from the atoms of q. In symbols, $K(q) = \{key(A) \rightarrow vars(A) : A \in q\}$.
- Let U be the set of variables occurring in q. If F is an atom of q, then F^+ denotes the attribute closure of the set $key(F)$ w.r.t. the set of all functional dependencies that arise in the atoms $q\backslash\{F\}$. In symbols, $F^+ = \{x \in U : K(q\backslash\{F\}) \models key(F) \rightarrow x\}$.

Definition 3. *Let ρ be an intersection tree for a Boolean conjunctive query q. The* attack graph *of ρ is the directed graph whose vertices are the atoms of q, and there is a directed edge from an atom F to an atom G if for every label L on the unique path from F to G in ρ, we have that $L \nsubseteq F^+$.*

We write $F \rightsquigarrow G$ to denote that there is an edge from F to G in the attack graph, and we say that F *attacks* G. A *cycle of size n* in the attack graph is a sequence of edges $F_0 \rightsquigarrow F_1 \rightsquigarrow ... \rightsquigarrow F_{n-1} \rightsquigarrow F_0$. We are now ready to state the main result in [6].

Theorem 1. *Let q be an acyclic self-join free Boolean conjunctive query and let τ be a join tree for q. Then the following two statements are equivalent:*

1. CERTAINTY(q) *is first-order expressible.*
2. *The attack graph of τ is acyclic.*

Every self-join free conjunctive query with two atoms R_1 and R_2 is acyclic and has only one join tree that is a single edge connecting the two atoms. Hence, the attack graph can only have a cycle of length 2, which arises precisely when $L \nsubseteq R_1^+$ and $L \nsubseteq R_2^+$. Thus, Theorem 1 yields the following corollary.

Corollary 1. *Let q be a self-join free Boolean conjunctive query with two atoms R_1 and R_2, and let L be the set of variables shared by R_1 and R_2. Then the following two statements are equivalent:*

1. CERTAINTY(q) *is first-order expressible.*
2. *$L \subseteq R_1^+$ or $L \subseteq R_2^+$.*

Consider the queries q_1, q_2, q_3 from the Introduction. For the query $q_1 = \exists x, y, z.R_1(\underline{x}, y) \wedge R_2(\underline{y}, z)$, we have that $L = \{y\}$, $R_1^+ = \{x\}$, and $R_2^+ = \{y\}$; since $L \subseteq R_2^+$, it follows that CERTAINTY(q_1) is first-order expressible. In contrast, for the query $q_2 = \exists x, y.R_1(\underline{x}, y) \wedge R_2(\underline{y}, x)$, we have that $L = \{x, y\}$, $R_1^+ = \{x\}$, and $R_2^+ = \{y\}$; since $L \nsubseteq R_1^+$ and $L \nsubseteq R_2^+$, it follows that CERTAINTY(q_2) is not first-order expressible. Similarly, CERTAINTY(q_3) is not first-order expressible.

4. Dichotomy of the Certain Answers of Self-join Free Conjunctive Queries with Two Atoms

We will now prove a *dichotomy* in the complexity of CERTAINTY(q), where q is a self-join free Boolean conjunctive query with exactly two atoms and such that CERTAINTY(q) is not first-order expressible.

Theorem 2. *Let q be a self-join free Boolean conjunctive query with two atoms R_1 and R_2 such that* CERTAINTY(q) *is not first-order expressible. Then either* CERTAINTY(q) *is in P or* CERTAINTY(q) *is coNP-complete. Moreover, the complexity of* CERTAINTY(q) *is determined by the following criterion:*

1. *If $key(R_1) \cup key(R_2) \subseteq L$, then* CERTAINTY($q$) *is in P;*
2. *If $key(R_1) \cup key(R_2) \nsubseteq L$, then* CERTAINTY($q$) *is coNP-complete,*

where R_1, R_2 are the two atoms of q, and L is the set of variables shared by R_1 and R_2.

To illustrate Theorem 2, let us consider again the queries q_2 and q_3 from the Introduction. As stated earlier, Corollary 1 implies that neither CERTAINTY(q_2) nor CERTAINTY(q_3) is first-order expressible. For the query $q_2 = \exists x, y.R_1(\underline{x}, y) \wedge R_2(\underline{y}, x)$, we have that $key(R_1) = \{x\}$, $key(R_2) = \{y\}$, and $L = \{x, y\}$; since $key(R_1) \cup key(R_2) \subseteq L$, it follows that CERTAINTY(q_2) is in P. In contrast, for the query $q_3 = \exists x, x', y.R_1(\underline{x}, y) \wedge R_2(\underline{x'}, y))$, we have that $key(R_1) = \{x\}$, $key(R_2) = \{x'\}$, and $L = \{y\}$; since $key(R_1) \cup key(R_2) \nsubseteq L$, it follows that CERTAINTY(q_3) is coNP-complete.

If CERTAINTY(q) is first-order expressible, then CERTAINTY(q) is in P. Consequently, Theorem 2 yields the following dichotomy theorem for self-join free Boolean conjunctive queries with exactly two atoms.

Corollary 2. *If q is a self-join free Boolean conjunctive query with exactly two atoms, then either* CERTAINTY(q) *is in P or* CERTAINTY(q) *is coNP-complete.*

As mentioned in Section 2, every self-join free Boolean conjunctive with exactly two atoms is acyclic; moreover, the edge connecting the two atoms of the query is the only join tree of the query. Also, it is well known that there is a linear-time algorithm for computing the closure of a given set of attributes w.r.t. a given set of functional dependencies [9]. These facts together with Corollary 1 imply that there is linear-time algorithm to determine, given a self-joint free Boolean conjunctive query q with exactly two atoms, whether or not CERTAINTY(q) is first-order expressible. By combining the preceding remarks with Theorem 2, we obtain the following result.

Corollary 3. *Let q be a self-join free Boolean conjunctive query with two atoms R_1 and R_2, and let L be the set of variables shared by R_1 and R_2. Then the following statements are true.*

1. *If $L \subseteq R_1^+$ or $L \subseteq R_2^+$, then* CERTAINTY(q) *is first-order expressible.*
2. *If $L \nsubseteq R_1^+$, $L \nsubseteq R_2^+$, and $key(R_1) \cup key(R_2) \subseteq L$, then* CERTAINTY($q$) *is in P but is not first-order expressible.*
3. *If $L \nsubseteq R_1^+$, $L \nsubseteq R_2^+$, and $key(R_1) \cup key(R_2) \nsubseteq L$, then* CERTAINTY($q$) *is coNP-complete.*

Furthermore, there is a linear-time algorithm to determine, given a self-join free Boolean conjunctive query q with exactly two atoms, if CERTAINTY(q) *is first-order expressible, or* CERTAINTY(q) *is in P but not first-order expressible, or* CERTAINTY(q) *is coNP-complete.*

Before embarking on the proof of Theorem 2, we describe briefly our strategy. Let q be a self-join free Boolean conjunctive query with two atoms R_1 and R_2 such that CERTAINTY(q) is not first-order expressible. In Section 4.1, we prove the intractability side of the dichotomy, that is, we show that if the query q is such that $key(R_1) \cup key(R_2) \nsubseteq L$, then CERTAINTY($q$) is coNP-hard. As a stepping stone, in Lemma 1, we show that CERTAINTY(q') is coNP-hard, where q' is the query $\exists x, y, z.S_1(\underline{x, z}, y) \wedge S_2(\underline{y}, x)$; this is done via a polynomial-time reduction from MONOTONE SAT, a problem well known to be NP-complete (see [10] and [11]). After this, in Lemma 2, we show that if q is a query such that $key(R_1) \cup key(R_2) \nsubseteq L$, then CERTAINTY($q'$) can be reduced in polynomial time to CERTAINTY(q); hence, CERTAINTY(q) is coNP-hard.

In Section 4.2, we prove the tractability side of the dichotomy, that is, we show that if the query q is such that $key(R_1) \cup key(R_2) \subseteq L$, then CERTAINTY($q$) is in P. For this, we introduce the notion of the *conflict-join* graph and show that CERTAINTY(q) can be reduced in polynomial time to the problem of finding an independent set of a certain size in a *conflict-join* graph. In general, the problem of finding an independent set of a certain size in a given graph is NP-complete. However, there are families of graphs on which this problem can be solved in polynomial time. One such family is the class of all *claw-free* graphs. In Lemma 4, we show that if q is a query with two atoms that satisfies the condition $key(R_1) \cup key(R_2) \subseteq L$, then the *conflict-join* graph of q is *claw-free*; hence, CERTAINTY(q) is in *P*. Theorem 2 then follows immediately by combining Lemma 2 with Lemma 4.

4.1. The Intractability Side of the Dichotomy

We begin by observing that if q is a query such that CERTAINTY(q) is not first-order expressible and $key(R_1) \cup key(R_2) \nsubseteq L$, then the variables of q exhibit a particular pattern.

Proposition 1. *Let q be a self-join free Boolean conjunctive query with two atoms such that* CERTAINTY(q) *is not first-order expressible. Let R_1, R_2 be the two atoms of q, and let L be the set of variables shared by R_1 and R_2. Then the following hold:*

1. *There exist four variables u, v, w, w' with the property that $u \in key(R_1) \setminus key(R_2)$, $v \in key(R_2) \setminus key(R_1)$, $w \in L \setminus key(R_1)$, and $w' \in L \setminus key(R_2)$.*

2. *If, in addition, $key(R_1) \cup key(R_2) \nsubseteq L$, then the variable u can be chosen to also satisfy $u \in key(R_1) \setminus L$ or the variable v can be chosen to also satisfy $v \in key(R_2) \setminus L$.*

Proof. Since CERTAINTY(q) is not first-order expressible, Corollary 1 tells that $L \nsubseteq R_1^+$ and $L \nsubseteq R_2^+$. We claim that $key(R_1) \nsubseteq key(R_2)$ and $key(R_2) \nsubseteq key(R_1)$. Indeed, if $key(R_1) \subseteq key(R_2)$, then $key(R_1) \subseteq R_2^+$ and also $nkey(R_1) \subseteq R_2^+$. Consequently, $L \subseteq R_2^+$, which contradicts the hypothesis. A similar argument shows that $key(R_2) \nsubseteq key(R_1)$. Thus, there are variables u and v such that $u \in key(R_1) \setminus key(R_2)$ and $v \in key(R_2) \setminus key(R_1)$. Since $L \nsubseteq R_1^+$ and $key(R_1) \subseteq R_1^+$, there is a variable w such that $w \in L \setminus key(R_1)$. Similarly, since $L \nsubseteq R_2^+$ and $key(R_2) \subseteq R_2^+$, there is a variable w' such that $w' \in L \setminus key(R_2)$.

Assume that, in addition, $key(R_1) \cup key(R_2) \nsubseteq L$ holds. This means that $key(R_1) \nsubseteq L$ or $key(R_2) \nsubseteq L$. In the first case, there exists a variable $u \in key(R_1) \setminus L$ (hence, also $u \in key(R_1) \setminus key(R_2)$). In the second case, there exists a variable $v \in key(R_2) \setminus L$ (hence, also $v \in key(R_2) \setminus key(R_1)$). ∎

Let q' be the query $\exists x, y, z. S_1(\underline{x, z}, y) \wedge S_2(\underline{y}, x)$. Corollary 1 implies that CERTAINTY(q) is not first-order expressible. Moreover, we have that $key(S_1) \cup key(S_2) \nsubseteq L$. It is easy to verify directly that the variables of q' exhibit the pattern described in Proposition 1. Specifically, the role of u is played by z, the roles of both v and w is played by y, and the role of w' is played by x.

Lemma 1. *Let q' be the query $\exists x, y, z. S_1(\underline{x, z}, y) \wedge S_2(\underline{y}, x)$. Then CERTAINTY($q'$) is coNP-hard.*

Proof. We will reduce Monotone SAT to CERTAINTY(q') in polynomial time. Let φ be a Boolean formula in conjunctive normal form such that each clause has either positive literals only (positive clause) or negative literals only (negative clause); without loss of generality, assume that each variable of φ occurs in some positive clause and in some negative clause. Construct an instance I over the schema of q' as follows:

- For every positive clause c_i and every variable p in it, generate a fact $S_1(1, c_i, p)$ in I.
- For every negative clause c_j and every variable p in it, generate a fact $S_1(0, c_j, p)$ in I.
- For every variable p, generate two facts $S_2(p, 0)$ and $S_2(p, 1)$ in I.

We will now prove that φ is satisfiable if and only if there is a repair of I that does not satisfy q'.

Assume first that there exists a satisfying assignment θ for φ. Construct the following instance r:

- For every positive clause c_i of φ, choose a variable p in c_i such that $\theta(p) = 1$. Add $S_1(1, c_i, p)$ to r.
- For every negative clause c_j of φ, choose a variable p in c_j such that $\theta(p) = 0$. Add $S_1(0, c_j, p)$ to r.
- For every variable p of φ, if $\theta(p) = 1$, then add $S_2(p, 0)$ to r; otherwise, add $S_2(p, 1)$ to r.

It is easy to see that r is a repair of I and $r \not\models q'$.

Next, assume that r is a repair of I such that $r \not\models q'$. Let θ be the following truth assignment.

- For every fact $S_1(1, c_i, p)$ in r, set $\theta(p) = 1$.
- For every fact $S_1(0, c_j, p)$ in r, set $\theta(p) = 0$.
- For every variable p for which there is no fact of the form $S_1(\_, \_, p)$ in r, assign to p value 0 or 1 arbitrarily.

It is easy to see that θ is a valid truth assignment that satisfies φ. ∎

We will use the following terminology and notation. We say that two facts $R^I(a_1, \ldots, a_n)$ and $R^I(b_1, \ldots, b_n)$ *form a conflict* if the two tuples $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ witness a violation of the key constraint of R; we also say that these two facts are *key-equal*. If a and b are constants, then $a \cdot b$ is a new constant encoding the pair (a, b) in a unique way; in other words, the function $(a, b) \mapsto a \cdot b$ is injective.

Lemma 2. *Let q be a self-join free Boolean conjunctive query with two atoms such that CERTAINTY(q) is not first-order expressible. Let R_1, R_2 be the two atoms of q, and let L be the set of variables shared by R_1 and R_2. If $key(R_1) \cup key(R_2) \nsubseteq L$, then CERTAINTY($q$) is coNP-hard.*

Proof. Let q' be the query $\exists x, y, z. S_1(\underline{x, z}, y) \wedge S_2(\underline{y}, x)$ of Lemma 1. We will show that CERTAINTY(q') can be reduced to CERTAINTY(q) in polynomial time. To this effect, given an instance I' over the schema of q', we will construct an instance I over the schema of q such that there is a repair of I' on which q' is false if and only if there is a repair of I on which q is false.

Assume that the two atoms of q are $R_1(s_1, \ldots, s_n)$ and $R_2(t_1, \ldots, t_m)$, where each s_i and each t_j is a variable or a constant (clearly, these variables need not be pairwise distinct). Let V be the set of variables occurring in q. From Proposition 1, there are variables u, v, w, w' such that $u \in key(R_1) \setminus key(R_2)$, $v \in key(R_2) \setminus key(R_1)$, $w \in L \setminus key(R_1)$, $w' \in L \setminus key(R_2)$. Moreover, $u \in key(R_1) \setminus L$ or $v \in key(R_2) \setminus L$ holds. Assume that $u \in key(R_1) \setminus L$ (the other case is similar). Let $P = \{u, v, w, w'\}$, let $Q = V \setminus P$, and c be a fixed constant. We are now ready to describe the construction of the instance I from I'. The intuition behind this construction is that the variable u in q plays the role of the variable z in q', the variable w' in q plays the role of the variable x in q', while the variables v and w in q play the role of the variable y in q'.

Consider first the atom $R_1(s_1, \dots, s_n)$ of q. Every fact $S_1(a_1, a_3, a_2)$ of I' generates a fact $R_1(b_1, \dots, b_n)$ of I, where each b_i is defined as follows:

1. If $s_i = u$, then $b_i = a_1 \cdot a_3$.
2. If $s_i = v$, then $b_i = a_2$.
3. If $s_i = w = w'$, then $b_i = a_1 \cdot a_2$.
4. If $s_i = w$ and $w \neq w'$, then $b_i = a_2$.
5. If $s_i = w'$ and $w' \neq w$, then $b_i = a_1$.
6. If s_i is a constant, then $b_i = s_i$.
7. In all other cases, $b_i = c$.

Next, consider the atom $R_2(t_1, \dots, t_m)$ of q. Every fact $S_2(a_2, a_1)$ of I' generates a fact $R_2(b_1, \dots, b_m)$ of I, where each b_i is defined by the preceding conditions 2 to 7 and with t_i in place of s_i. Note that the first condition is not applicable because $u \in key(R_1) \setminus L$, hence u cannot be among the variables occurring in the atom $R_2(t_1, \dots, t_m)$.

In the preceding construction, each b_i is defined in a unique way. The reason is that u is different from v, w, and w', and also v is different from w'; thus, no s_i can meet two of the conditions 1 to 7 at the same time.

Let f be an S_i-fact of I' and let g be an R_i-fact of I, $i = 1, 2$. We write $f \rightrightarrows g$ to denote that g has been generated by f in the way described above. Thus, $I = \{g : \text{there is a fact } f \text{ of } I' \text{ such that } f \rightrightarrows g\}$. The preceding construction ensures two important properties that we now state and prove.

Property 1. *For $i = 1, 2$, let f_1, f_2 be S_i-facts of I' and let g_1, g_2 be R_i-facts of I such that $f_1 \rightrightarrows g_1$ and $f_2 \rightrightarrows g_2$. The following statements are equivalent.*

1. *The facts f_1 and f_2 are key-equal.*
2. *The facts g_1 and g_2 are key-equal.*

To verify that Property 1 holds, assume first that f_1, f_2 are S_1-facts and that g_1, g_2 are R_1-facts. Assume that $g_1 = R_1(b_1, \dots, b_n)$ and $g_2 = R_1(b'_1, \dots, b'_n)$. If f_1 and f_2 are key equal, then they must be of the form $S_1(a_1, a_3, a_2$ and $S_1(a_1, a_3, a'_2)$, respectively. The preceding construction implies that the values of the keys of R_1-facts depend only on the value of the variable u or only on the values of the variables u and w', provided $w' \neq w$ (if $w' = w$, then $w' \notin key(R_1)$). If $s_i = u$, then, by construction, we have that $b_i = a_1 \cdot a_3 = b'_i$; furthermore, if $w' \neq w$, then, by construction, we have that $b_i = a_1 = b'_i$. Consequently, g_1 and g_2 are key-equal facts. For the other direction, assume that the facts g_1 and g_2 are key-equal. Assume that $f_1 = S_1(a_1, a_3, a_2)$ and $f_2 = S_1(a'_1, a'_3, a'_2)$. Since $u \in key(R_1)$, there is some i such that $u = s_i$, hence $b_i = b'_i$. Furthermore, by construction, we have that $b_i = a_1 \cdot a_3$ and $b'_i = a'_1 \cdot a'_3$. Consequently, $a_1 \cdot a_3 = a'_1 \cdot a'_3$, which implies that $a_1 = a'_1$ and $a_3 = a'_3$. Thus, f_1 and f_2 are key-equal facts. A similar argument shows that Property 1 holds also when f_1, f_2 are S_2-facts and g_1, g_2 are R_2-facts.

Property 2. *For $i = 1, 2$, if f_1, f_2 are S_i-facts of I' and g is an R_i-fact of I such that $f_1 \rightrightarrows g$ and $f_2 \rightrightarrows g$, then $f_1 = f_2$.*

To verify that Property 2 holds, assume first that $f_1 = S_1(a_1, a_3, a_2)$, $f_2 = S_1(a'_1, a'_3, a'_2)$, and $g = R_1(b_1, \dots, b_n)$. By Property 1, the facts f_1 and f_2 are key-equal, hence $a_1 = a'_1$ and $a_3 = a'_3$. Since $w \in L$, there is some i such that $s_i = w$. If $w = w'$, then, by construction, $a_1 \cdot a_2 = b_i = a_1 \cdot a'_2$, hence $a_2 = a'_2$. If $w \neq w'$, then $a_2 = b_i = a'_2$. In either case, we have that $a_2 = a'_2$ and so $f_1 = f_2$. A similar argument shows that Property 2 holds also for the case in which f_1, f_2 are S_2-facts and g is an R_2-fact.

We continue with the proof of the lemma. We will show that there is a repair of I' on which q' is false if and only if there is a repair of I on which q is false.

Assume that r' is a repair of I' such that $r' \not\models q'$. Let $r = \{g \in I : \text{there is a fact } f \in r' \text{ such that } f \rightrightarrows g\}$. We claim that r is a repair of I such that $r \not\models q$.

Properties 1 and 2 imply that r is a repair of I. Indeed, to show that r is a consistent instance, let g_1, g_2 be two key-equal facts of r. Let f_1, f_2 be two facts of r' such that $f_1 \rightrightarrows g_1$ and $f_2 \rightrightarrows g_2$. Property 1 implies that the facts f_1 and f_2 are key equal. Since r is a consistent instance, it follows that $f_1 = f_2$, hence $g_1 = g_2$. To show that r is a maximal consistent subinstance of I, let g be a fact of I such that $r \cup \{g\}$ is consistent. Let f be a fact of I' such that $f \rightrightarrows g$. We claim that $r' \cup \{f\}$ is consistent. Indeed, assume that f' is a fact of r' such that f and f' are key-equal, and let $g' \in r$ be such that $f' \rightrightarrows g'$. By Property 1, we have that g and g' are key-equal facts, hence (since $r \cup \{g\}$ is consistent) $g = g'$. Property 2 implies that $f' = f$, hence $g \in r$; this completes the proof that r is a repair of I.

Next, we show that r does not satisfy q. Towards a contradiction, assume that $R_1(b_1, \dots, b_n)$ and $R_2(b'_1, \dots, b'_m)$ are two facts of r that satisfy q. Let $S_1(a_1, a_3, a_2)$ and $S_2(a'_2, a'_1)$ be two facts of r' such that $S_1(a_1, a_3, a_2) \rightrightarrows R_1(b_1, \dots, b_n)$ and $S_2(a'_2, a'_1) \rightrightarrows R_2(b'_1, \dots, b'_m)$. Consider the variables w, w' and recall that $w \in L$ and $w' \in L$. Let i and j be such that $s_i = w$ and $t_j = w$. We distinguish two cases. If $w = w'$, then $b_i = a_1 \cdot a_2$ and $b'_j = a'_1 \cdot a'_2$. Since the facts $R_1(b_1, \dots, b_n)$ and $R_2(b'_1, \dots, b'_m)$ satisfy q, we have that $b_i = b'_j$, hence $a_1 = a'_1$ and $a_2 = a'_2$, which implies that the facts $S_1(a_1, a_3, a_2)$ and $S_2(a'_2, a'_1)$ satisfy q', a contradiction. If $w \neq w'$, then $b_i = a_1$ and $b'_j = a'_1$. Since $b_i = b'_j$, we have that $a_1 = a'_1$. Furthermore, let k and l be such that $s_k = w'$ and $t_l = w'$. Then $b_k = a_2$ and $b'_l = a'_2$. Since $b_k = b'_l$, we have that $a_2 = a'_2$, which implies that the facts $S_1(a_1, a_3, a_2)$ and $S_2(a'_2, a'_1)$ satisfy q', a contradiction.

In the other direction, assume that r is a repair of I such that $r \not\models q$. Let $r' = \{f \in I' :$

there is a fact $g \in r$ such that $f \rightrightarrows g\}$. We claim that r' is a repair of I' such that $r' \not\models q'$.

As before, Properties 1 and 2 imply that r' is a repair of I'. Indeed, if f_1 and f_2 are two key-equal facts of r', then, by Property 1, the facts g_1 and g_2 of r are key-equal, where $f_1 \rightrightarrows g_1$ and $f_2 \rightrightarrows g_2$. It follows that $g_1 = g_2$ and so, by Property 2, we have that $f_1 = f_2$. Similarly, if $r' \cup \{f\}$ is consistent and $f \rightrightarrows g$, then $r \cup \{g\}$ is consistent, hence $g \in r$ and so $f \in r'$. Finally, we show that r' does not satisfy q'. Towards a contradiction, assume that $S_1(a_1, a_3, a_2)$ and $S_2(a_2, a_1)$ are two facts of r' that satisfy q'. Let g_1 and g_2 be the facts of r such that $S_1(a_1, a_3, a_2) \rightrightarrows g_1$ and $S_2(a_2, a_1) \rightrightarrows g_2$. By the construction of I from I' and since the variable u is not in L, we have that the facts g_1 and g_2 of r agree on all values corresponding to variables in L. Consequently, the facts g_1 and g_2 satisfy the query q, contrary to the hypothesis. This completes the proof of Lemma 2. ∎

As an illustration of Lemma 2, it follows that CERTAINTY(q_3) is coNP-hard, where q_3 is the query $\exists x, x', y. R_1(\underline{x}, y) \wedge R_2(\underline{x'}, y)$ from the Introduction. In addition, CERTAINTY(q) is coNP-hard if q is one of the following queries:

$$\exists x, x', y, z. R_1(\underline{x, z}, x', y) \wedge R_2(\underline{x'}, x, y);$$
$$\exists x, y, z, w. R_1(\underline{x, w}, z, y) \wedge R_2(\underline{x, z}, y);$$
$$\exists x, y, z, w. R_1(\underline{x, z}, y, w) \wedge R_2(\underline{y}, x, w).$$

4.2. The Tractability Side of the Dichotomy

In this section, we introduce the notion of the *conflict-join graph* and use it to study when CERTAINTY(q) is tractable, where q is a self-join free Boolean conjunctive query with two atoms. As before, we assume that there is one key constraint for each relation symbol.

Definition 4. *Let q be a self-join free Boolean conjunctive query with two atoms. If I is an instance, then the* conflict-join graph *$H_{I,q} = (V, E)$ is defined as follows:*

- *The set V of the nodes of $H_{I,q}$ consists of all* facts *of I.*
- *For every pair of facts that form a conflict, add an edge in E connecting these two facts.*
- *For every pair of facts that satisfy the query q, add an edge in E connecting these two facts.*

For every fixed query q, the size of the conflict-join graph $H_{I,q}$ is quadratic in the size of the instance I. If D is a set of pairwise key-equal facts of I, then every two distinct elements of D form a conflict, which implies that D induces a clique in $H_{I,q}$. A maximal set of pairwise key-equal facts of I must contain all facts that are key-equal to one of its members; moreover, if D and D' are distinct maximal sets of pairwise key-equal facts, then $D \cap D' = \emptyset$. Consequently, the set V of nodes of $H_{I,q}$ can be partitioned into pairwise disjoint sets $V_1, \ldots, V_n$ such that each V_i is a maximal set of pairwise key-equal facts of I. Also, by construction and since q is a self-join free query, the set E of edges of $H_{I,q}$ can be partitioned into two disjoint sets E_1 and E_2, where E_1 consists of all edges whose endpoints form a conflict in I, and E_2 consists of all edges whose endpoints are facts that satisfy the query q.

In what follows, we will establish a connection between the existence of a maximum independent set of a particular size in the conflict-join graph $H_{I,q}$ and the existence of a repair r of I such that $r \not\models q$. Recall that an *independent set* in a graph G is a set of nodes with no edges between them. A *maximum independent set* is an independent set of maximum cardinality. The *independent set number* $\alpha(G)$ of a graph G is the cardinality of a maximum independent set of G.

We now focus on the independent set number $\alpha(H_{I,q})$ of the conflict-join graph associated with an instance I. It is easy to see that $\alpha(H_{I,q}) \leq n$, where n is the number of the maximal sets $V_1, \ldots, V_n$ of pairwise key-equal facts of I. Indeed, this holds because each V_i induces a clique in $H_{I,q}$, so an independent set in $H_{I,q}$ can contain at most one node from each V_i, $1 \leq i \leq n$.

Example 1. *Let q_2 be the query $\exists x, y. R_1(\underline{x}, y) \wedge R_2(\underline{y}, x)$ from the Introduction. Consider the instance $I = \{R_1(a, b), R_1(a, b'), R_1(a, b''), R_1(a', b), R_2(b, a), R_2(b, a')\}$. Figure 1 depicts the conflict-join graph H_{I,q_2}. Note that H_{I,q_2} is partitioned into three maximal sets of pairwise key-equal facts. Note also that the set $r = \{R_1(a, b'), R_1(a', b), R_2(b, a)\}$ has as size three and is a maximum independent set of H_{I,q_2}. Furthermore, viewed as an instance, r is a repair of I and $r \not\models q_2$. The next lemma tells that this is no accident.*

Lemma 3. *Assume that q is a self-join free Boolean conjunctive query with two atoms and I is an instance. Let $H_{I,q}$ be the conflict-join graph associated with I and q, let $\alpha(H_{I,q})$ be the independent set number of $H_{I,q}$, and let n be the number of distinct maximal sets of pairwise key-equal facts of I. Then the following statements are equivalent:*

1. *There is a repair r of I such that $r \not\models q$.*
2. *$\alpha(H_{I,q}) = n$.*

PROOF. Assume first that r is a repair of I such that $r \not\models q$. Let M be the set of all facts of r. We claim that M is an independent set in $H_{I,q}$ and has size n. To see that M is an independent set in $H_{I,q}$, consider two distinct facts f_1 and f_2 of r. If they involve the same relation symbol, then they cannot be key equal because r is a consistent instance, hence there is no edge between them in $H_{I,q}$. If they involve different relation symbols, then together they cannot satisfy q because $r \not\models q$, hence there is no edge between them in $H_{I,q}$. To see that M has size n, notice that, since r is a maximal consistent sub-instance of I, it must contain one fact of each different key value, which means that r must contain one fact from each of the n maximal sets of pairwise key-equal facts of I. Since $\alpha(H_{I,q}) \leq n$, it follows that $\alpha(H_{I,q}) = n$.

For the other direction, assume that M is an independent set of $H_{I,q}$ of size n. Let r be the sub-instance of I formed by the facts of M. We claim that r is a repair of I such that $r \not\models q$. Indeed, since M is an independent set of $H_{I,q}$, we have that r is consistent and also $r \not\models q$. Also, since M is of size n, we have that r must contain a fact of each different key value, hence r is a maximal consistent sub-instance of I. ∎

Notice that the proof of Lemma 3 actually establishes something stronger, namely, that the repairs of I that make q false are precisely the independent sets of $H_{I,q}$ of size n.

It is well known that the problem of computing the independent set number of a given graph is NP-hard [11]. However, it is also known that there are restricted classes of graphs for which this problem is solvable in polynomial time. In particular, this holds true for *claw-free* graphs, *chordal* graphs, and *perfect* graphs. Claw-free graphs will turn out to be of particular interest to us. A graph is *claw-free* if it does not contain a claw as an induced subgraph, where the *claw* is the complete bipartite graph $K_{1,3}$ (see Figure 2). Equivalently, a graph is claw-free if no node has three pairwise non-adjacent neighbors. Claw-free graphs form a broad class of graphs that enjoy good algorithmic properties. In particular, a polynomial-time algorithm for computing the independent set number on claw-free graphs was given by Minty [12].

Lemma 4. *Assume that q is a self-join free Boolean conjunctive query with exactly two atoms. Let R_1, R_2 be the two atoms of q, and let L be the set of variables shared by R_1 and R_2. If $key(R_1) \cup key(R_2) \subseteq L$, then, for every instance I, the conflict-join graph $H_{I,q}$ is claw-free. Consequently, if $key(R_1) \cup key(R_2)) \subseteq L$, then* CERTAINTY$(q)$ *is in P.*

PROOF. Let I be an instance. We first observe the following regarding the conflict-join graph $H_{I,q}$.

- If f_1, f_2, f_3 are three facts of I such that (f_1, f_2) and (f_1, f_3) are edges in E_1, then (f_2, f_3) is also an edge in E_1. Indeed, since (f_1, f_2) and (f_1, f_3) are in E_1, it follows that f_1 is key-equal to both f_2 and f_3; hence, f_2 is key-equal to f_3, which implies that (f_1, f_3) is an edge in E_1.
- If f_1, f_2, f_3 are three facts of I such that (f_1, f_2) and (f_1, f_3) are edges in E_2, then (f_2, f_3) is an edge in E_1. To see this, we distinguish two cases, depending on whether f_1 is an R_1-fact or an R_2-fact. Assume first that f_1 is an R_1-fact. Then, f_2 and f_3 must be R_2-facts. Since f_1 and f_2 satisfy q, they must agree on all values corresponding to variables in L. Given that $key(R_2) \subseteq L$, we have that f_1 and f_2 agree on all values corresponding to variables in $key(R_2)$. Similarly, f_1 and f_3 agree on all values corresponding to variables in $key(R_2)$. It follows that f_2 and f_3 are key-equal. The argument in the case that f_1 is an R_2-fact is similar.

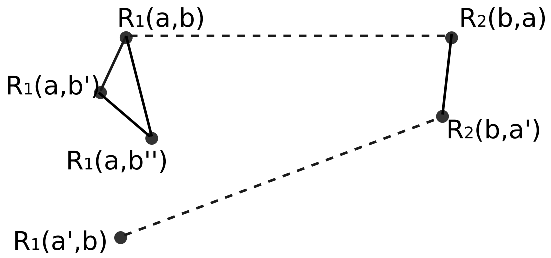

Figure 1: The conflict-join graph H_{I,q_2} for the query and the instance in Example 1. Edges drawn as continuous lines connect pairs of facts that conflict; edges drawn as dashed lines connect facts that together satisfy q.

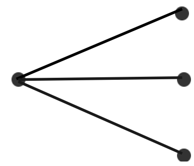

Figure 2: The claw graph $K_{1,3}$

We now prove that the conflict-join graph $H_{I,q}$ is claw-free. Let f_1, f_2, f_3, and f_4 be four facts of I such that (f_1, f_2), (f_1, f_3), (f_1, f_4) are edges in E. Then either at least two of these three edges are in E_1 or at least two of these three edges are in E_2. If, say, both (f_1, f_2) and (f_1, f_3) are in E_1, then, by the first observation above, we have that (f_2, f_3) is an edge in E_1 (and hence in E). If, say, both (f_1, f_2) and (f_1, f_3) are in E_2, then, by the second observation above, we have that (f_2, f_3) is an edge in E_1 (and hence in E). Therefore, the nodes f_1, f_2, f_3, and f_4 do not induce a claw in $H_{I,q}$.

Finally, assuming that $(key(R_1) \cup key(R_2)) \subseteq L$, there is a polynomial-time algorithm for CERTAINTY(q). Specifically, given an instance I, we first construct the conflict-join graph $H_{I,q}$ in polynomial time in the size of I. Since $H_{I,q}$ is claw-free, we can use Minty's algorithm [12] to compute the independent set number $\alpha(H_{I,q})$ in polynomial time in the size of $H_{I,q}$ and, hence, in polynomial time in the size of I. We then compare $\alpha(H_{I,q})$ to the number n of distinct maximal sets of pairwise key-equal facts of I, which can also be computed in polynomial time in the size of I. By Lemma 3, we have that CERTAINTY(q) is true on I if and only if $\alpha(H_{I,q}) < n$. ∎

It should be noted that Arenas et al. [13] introduced the notion of the *conflict* graph while studying the consistent answers of aggregate queries. The conflict graph is constructed from the constraints and the instance, while our conflict-join graph takes also the query into account. Arenas et al. used the tractability of the maximum independent set number on claw-free graphs to show that if a relational schema with at most two functional dependencies is in Boyce-Codd Normal Form, then there is a polynomial-time algorithm for computing the consistent answers of COUNT(*) queries (see [13, Theorem 12]).

The preceding Lemma 4 could also be obtained via a reduction to the problem of computing the consistent answers of COUNT(*) queries and then by appealing to Theorem 12 in [13]. The proof we gave here is direct and self-contained.

Lemma 4 gives a broad sufficient condition for the tractability of $\textsc{certainty}(q)$ for self-join free Boolean conjunctive queries q with exactly two atoms. In particular, it yields a unifying polynomial-time algorithm for $\textsc{certainty}(q)$ that applies to several interesting queries q for which $\textsc{certainty}(q)$ is not first-order expressible. To begin with it implies that $\textsc{certainty}(q_2)$ is in P, where q_2 is the query $\exists x, y.R_1(\underline{x}, y) \wedge R_2(\underline{y}, x)$ from the Introduction. Note that the sole focus of [5] was showing that $\textsc{certainty}(q_2)$ is in P (using a different algorithm than ours) but is not first-order expressible. Also, Lemma 4 implies that $\textsc{certainty}(q)$ is in P, where q is one of the following three queries:

$$\exists x, y, z.R_1(\underline{x, z}, y) \wedge R_2(\underline{y}, x, z);$$
$$\exists x, y, z.R_1(\underline{x}, y, z) \wedge R_2(\underline{y}, x, z);$$
$$\exists x, y, z.R_1(\underline{x, y}, z) \wedge R_2(\underline{x, z}, y).$$

Theorem 2 now follows by combining Lemma 2 with Lemma 4. Thus, if q is a self-join free Boolean conjunctive query with two atoms such that $\textsc{certainty}(q)$ is not first-order expressible, then either $\textsc{certainty}(q)$ is in P or $\textsc{certainty}(q)$ is coNP-complete. Naturally, this dichotomy is interesting provided that $P \neq NP$. Moreover, assuming that $P \neq NP$, we have that if q is a self-join free Boolean conjunctive query with two atoms R_1 and R_2 such that $\textsc{certainty}(q)$ is not first-order expressible, then $\textsc{certainty}(q)$ is in P if and only if $key(R_1) \cup key(R_2) \subseteq L$.

By Lemma 4, $key(R_1) \cup key(R_2) \subseteq L$ is a sufficient condition for tractability of $\textsc{certainty}(q)$, where q is an arbitrary self-join free Boolean conjunctive query with two atoms R_1 and R_2. In general, however, the condition $key(R_1) \cup key(R_2) \subseteq L$ is *not* necessary for tractability of $\textsc{certainty}(q)$, where q is an arbitrary self-join free Boolean conjunctive query with two atoms R_1 and R_2. For example, consider again the query $q_1 = \exists x, y, z.R_1(\underline{x}, y) \wedge R_2(\underline{y}, z)$ from the Introduction. Then $key(R_1) \cup key(R_2) = \{x, y\} \nsubseteq L = \{y\}$. Nonetheless, as seen earlier, $\textsc{certainty}(q)$ is first-order expressible, hence it is in P. Similarly, if q is the query $\exists x, y, z.R_1(\underline{x, y}, z) \wedge R_2(\underline{y, u}, w)$, then $key(R_1) \cup key(R_2) = \{x, y, u\} \nsubseteq L = \{y\}$, yet $\textsc{certainty}(q)$ is in P, because $\textsc{certainty}(q)$ is first-order expressible, due to $L \subseteq R_1^+ = \{x, y\}$.

The results presented here apply to Boolean self-join free conjunctive queries with two atoms that may contain constants. Consequently, the dichotomy in the complexity of $\textsc{certainty}(q)$ can be extended to non-Boolean queries as well. Specifically, let q be a query of arity k, for some $k \geq 1$, and let I be an instance. Then a k-tuple t with values from the active domain of I is in the consistent answers of q on I if and only for every repair r of I, we have that t is in $q(r)$. This is the same as the Boolean query $q(t)$ being true in every repair r of I, where $q(t)$ is the query obtained from q by substituting the free variables of q (i.e., the variables that are not existentially quantified) with corresponding constants from t.

Acknowledgments Research on this paper was partially supported by NSF Grant IIS-0905276. We thank the two anonymous referees for their insightful comments, suggestions, and pointers to the literature.

References

[1] M. Arenas, L. Bertossi, J. Chomicki, Consistent query answers in inconsistent databases, in: 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'99), 1999, pp. 68–79.

[2] M. Lenzerini, Data Integration: A Theoretical Perspective, in: PODS, 2002, pp. 233–246.

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

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

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

[6] J. Wijsen, On the first-order expressibility of computing certain answers to conjunctive queries over uncertain databases, in: PODS, 2010, pp. 179–190.

[7] F. N. Afrati, P. G. Kolaitis, Repair checking in inconsistent databases: algorithms and complexity, in: ICDT, 2009, pp. 31–41.

[8] R. E. Ladner, On the structure of polynomial time reducibility, J. ACM 22 (1) (1975) 155–171.

[9] C. Beeri, P. A. Bernstein, Computational problems related to the design of normal form relational schemas, ACM Trans. Database Syst. 4 (1) (1979) 30–59.

[10] E. M. Gold, Complexity of automaton identification from given data, Information and Control 37 (3) (1978) 302–320.

[11] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, 1979.

[12] G. J. Minty, On maximal independent sets of vertices in claw-free graphs, J. Comb. Theory, Ser. B 28 (3) (1980) 284–304.

[13] M. Arenas, L. Bertossi, J. Chomicki, X. He, V. Raghavan, J. Spinrad, Scalar aggregation in inconsistent databases, Theoretical Computer Science. 296(3), 2003.