A SIMPLE ALGORITHM FOR CONSISTENT QUERY ANSWERING UNDER PRIMARY KEYS

DIEGO FIGUEIRA [a], ANANTHA PADMANABHA [b], LUC SEGOUFIN [c], AND CRISTINA SIRANGELO [d]

[a] Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, Talence, France
e-mail address: diego.figueira@cnrs.fr

[b] Indian Institute of Technology Madras, Chennai, India
e-mail address: ananthap@cse.iitm.ac.in

[c] INRIA, ENS-Paris, PSL University, France
e-mail address: luc.segoufin@inria.fr

[d] Université Paris Cité, CNRS, IRIF, F-75013, Paris, France
e-mail address: cristina@irif.fr

Abstract. We consider the dichotomy conjecture for consistent query answering under primary key constraints. It states that, for every fixed boolean conjunctive query q, testing whether q is certain (*i.e.* whether it evaluates to true over all repairs of a given inconsistent database) is either polynomial time or coNP-complete. This conjecture has been verified for self-join-free and path queries.

We propose a simple inflationary fixpoint algorithm for consistent query answering which, for a given database, naively computes a set Δ of subsets of facts of the database of size at most k, where k is the size of the query q. The algorithm runs in polynomial time and can be formally defined as:

(1) Initialize Δ with all sets S of at most k facts such that $S \models q$.
(2) Add any set S of at most k facts to Δ if there exists a block B (i.e., a maximal set of facts sharing the same key) such that for every fact $a \in B$ there is a set $S' \subseteq S \cup \{a\}$ such that $S' \in \Delta$.

For an input database D, the algorithm answers "q is certain" iff Δ eventually contains the empty set. The algorithm correctly computes certainty when the query q falls in the polynomial time cases of the known dichotomies for self-join-free queries and path queries. For arbitrary boolean conjunctive queries, the algorithm is an under-approximation: the query is guaranteed to be certain if the algorithm claims so. However, there are polynomial time certain queries (with self-joins) which are not identified as such by the algorithm.

☞ This pdf contains internal links: clicking on a notion leads to its *definition*.[1]

Key words and phrases: consistent query answering, primary keys, conjunctive queries.

Journal version of the paper presented at ICDT 2023 [FPSS23], see Section 1.3 for a summary of the added material.

[1] `https://ctan.org/pkg/knowledge`

Preprint submitted to
Logical Methods in Computer Science

Contents

1. Introduction

A database often comes with integrity constraints. Such constraints are helpful in many ways, for instance in order to help optimizing query evaluation. When the database violates its integrity constraints we are faced with several possibilities. A first possibility is to clean the data until all integrity constraints are satisfied. This task is not easy as it is inherently non-deterministic: there could be many equally good ways to "repair" a database. A repair can be understood as a minimal way to change the database in order to satisfy the constraints.

Another possibility is to keep the database in its inconsistent state, postponing the problem until a query is issued. In order to evaluate the query on the inconsistent database D, the classical solution is to consider all possible repairs of D and return all the answers which are "certain", *i.e.*, the answers that are returned by the query when evaluated on *every* repair of D [ABC99]. However, this method usually has an impact on the complexity of the query evaluation problem. The impact will of course depend on the type of the integrity constraints and on the definition of a repair, but most often the worst case complexity increases at least by a factor which is exponential in the size of the database, since there could be exponentially many ways to repair a database.

Depending on the type of integrity constraints, what should be considered as a "good" notion of repair may be controversial. In this paper we consider primary key constraints, which are arguably the most common kind of integrity constraints in databases. For primary

keys, there is a unanimously accepted notion of repair. Primary key constraints identify, for each relation, a set of attributes which are considered to be the *key* of this relation. An inconsistent database is therefore a database that has distinct tuples sharing the same key within a relation. For such constraints, the standard notion of a repair is any maximal subset of the database satisfying all the primary key constraints. This amounts to keeping exactly one tuple for each group of tuples having the same key, in each relation. A simple analysis shows that there can be exponentially many repairs of a given database, and therefore a naive evaluation algorithm would have to evaluate the query on each of these exponentially many repairs.

As query language, we consider boolean conjunctive queries, which can be evaluated in polynomial time in data complexity. With the certain answer semantics described above, a query is "certain" on an inconsistent database if it is true on all its repairs. The data complexity of certain answers for conjunctive queries over inconsistent databases in the presence of primary key constraints is therefore in coNP. Indeed, in order to test whether the query is not certain, it is enough to guess a subset of the database which is a repair and which makes the query false. Further, it has been observed that for some conjunctive queries the certain query answering problem is coNP-hard [FM07] while, for other queries, it can be solved in polynomial time. The main conjecture for inconsistent databases in the presence of primary key constraints is that there are no intermediate cases: for any boolean conjunctive query, the certain answering problem is either solvable in polynomial time or it is coNP-complete.

The conjecture has been proved for self-join-free boolean conjunctive queries [KW17] and for path queries [KOW21]. However, the conjecture remains open for arbitrary conjunctive queries (with self-joins). In this paper we revisit the two cases above where the conjecture is known to hold: self-join-free queries and path queries.

1.1. **Contributions.** Our main contribution is the design of a simple fixpoint algorithm for computing certain answers of boolean conjunctive queries over inconsistent databases in the presence of primary key constraints. For every $k \geq 1$, we describe a fixpoint algorithm parameterized by k. The algorithm is always an under-approximation of the certain answers: on boolean queries, if it outputs 'yes' then the query is certain, *i.e.* it is true on all repairs of the database. But there could be "false negatives", that is, queries which are certain but on which the algorithm outputs 'no'.

In this paper we investigate the expressive power of our fixpoint algorithm, trying to understand when it solves the certain answering problem.

Our first result shows that in the case of self-join-free queries and path boolean queries, we can characterize the cases when our fixpoint algorithm computes the certain answers via a semantic condition. In other words, when the condition holds there exists k (namely, the number of atoms of the query) such that the corresponding fixpoint algorithm correctly computes the certain answer (taking k as parameter); and conversely when the condition does not hold, for every k there exists a database instance D such that the fixpoint algorithm parameterized by k outputs a false negative on D.

We then show that when our algorithm fails to compute the certain answers, i.e. when our semantic condition fails, the certain answering problem is actually coNP-hard. Hence, for self-join-free queries and path queries and assuming PTime $\neq$ coNP, our simple fixpoint algorithm solves the certain answering problem in all the cases where it is solvable in polynomial time. The current approaches for PTime solvable queries for these two classes are

mutually orthogonal and our result provides a uniform algorithm to solve all the polynomial time solvable queries known in the literature.

A natural question is then to wonder whether our algorithm always correctly computes the certain answering problem on all queries for which this problem is polynomial time computable. Our second result answers negatively to this question. There is a simple two-atom query, named $\mathtt{q}_4$ in the paper, with self-joins, whose certain answering problem is equivalent to bipartite matching under LogSpace reductions, and cannot be solved with our fixpoint algorithms. Recall that the bipartite matching problem can be solved in polynomial time and is NL-hard. This shows that in the presence of self-joins, the classification of the complexity of the certain answering problem is richer than in the self-join free case when it is either in LogSpace or coNP-complete [KW21] .

Our fixpoint algorithm is based on a function that is expressible in first-order logic (FO). Hence, when the fixpoint is bounded, the certain answering problem is expressible in FO. Our last result shows that for self-join-free and path queries the converse is true: whenever the certain answering problem is expressible in FO, the fixpoint algorithm is bounded and the certain answering problem is expressible by some bounded unfolding of the fixpoint algorithm.

Though our greedy fixpoint computation algorithm is simple, the proof of correctness when the semantic condition holds is non-trivial. In the case of self-join-free queries satisfying our semantic condition, to prove that the algorithm always computes the correct answer, we proceed by contradiction: if the algorithm fails to give the correct answer, we use the fixpoint definition of the algorithm in order to produce an infinite sequence of distinct facts of the database, contradicting its finiteness.

The situation is a bit simpler in the case of path queries, where we show that our fixpoint algorithm can simulate the polynomial time algorithm of [KOW21] for computing certain answers for a path query q, assuming that certain answering for q is polynomial time solvable.

For the lower bounds, we first show that our fixpoint algorithm fails to compute the certain answering problem for $\mathtt{q}_4$ by constructing for every number k, a database such that all its repairs satisfy $\mathtt{q}_4$ but the algorithm outputs 'no'. In the second step we reduce this query $\mathtt{q}_4$ to all queries that falsify the semantic condition: if the fixpoint algorithm would work for such queries, it would also work for $\mathtt{q}_4$. The reduction relies on the (syntactic) condition of [KW17] characterizing the class of queries having a coNP-complete certain answering problem.

1.2. **Related work.** Our work is inspired by the results of Koutris and Wijsen [KW21, KW17]. For self-join-free queries, the authors prove the polynomial-time case via a long sequence of reductions eventually producing a simple query whose certain answers can be solved efficiently. When unfolding the sequence of reductions this gives a complicated polynomial time algorithm with a complex proof of correctness. We have basically simplified the algorithm and pushed all the difficulty into the proof of correctness. Our algorithm is simple, but the proof of correctness is arguably as complex as theirs. Further, our algorithm does not give, a priori, the optimal LogSpace complexity result of [KW21] as we know that some of the path queries that can be solved with our algorithm are PTime complete [KOW21]. The semantic condition that we provide for characterizing the polynomial case in the self-join-free case can be effectively tested, but not efficiently, unlike

the simple syntactic characterization of [KW17] based on the so-called "attack graph" of the query.

In the case of path queries, [KOW21] also provides a simple fixpoint algorithm for solving the polynomial cases. Though it seems that their algorithm is different in spirit from ours, the two algorithms have some similarities that we use in order to "simulate" their fixpoint computation using ours.

For both self-join-free and path queries the cases where the certain answering problem is expressible in FO is also characterized in [KOW21, KW17]. In fact, our boundedness results use their characterizations.

Recently, the dichotomy conjecture has been proved for queries with two atoms [PSS24] and for "rooted tree-queries" [KOW23]. Certain answers for two-atom queries in the polynomial time cases is computed using a combination of our fixpoint algorithm and bipartite matching. Bipartite matching hardness has also been obtained for the consistent query answering problem in the presence of key constraints [KW20, Lemma 6.4] or in the presence of negated atoms [KW18].

1.3. **Conference paper.** The current article is based on the conference paper [FPSS23]. While the main results are essentially the same, though with improved explanations and figures, we have also added new material:

- We have added the complete characterization of the cases where our fixpoint algorithm works (Section 5.4 and Section 6.2).
- We show the link between boundedness of our algorithm and expressibility in FO of the certain answering problem (Section 7).

2. Preliminaries

A *relational signature* is a finite set of relation symbols associated with an arity. A finite relational structure D over a relational signature σ is composed of: a finite set, the domain $adom(D)$ of D, and a function associating to each symbol R of σ a relation $R(D)$ of the appropriate arity over $adom(D)$. A *database* is a finite relational structure.

An *R-fact* of a database D over a relational signature σ is a term of the form $R(\bar{a})$ where R is a symbol of σ and $\bar{a}$ a tuple in $R(D)$. A *fact* u is an R-fact for some R if $u = R(\bar{a})$ where R is the symbol associated to the fact u and $\bar{a}$ the tuple associated to u. A database can then be viewed as a finite collection of facts. The size of a database is the number of facts it contains. Assuming σ is fixed (which we will implicitly do in this paper) this is equivalent to the usual notion of size for a database, up to some polynomial function.

A *key constraint* over a relational signature σ specifies for every relation symbol R of σ a certain set of indices (columns) of R as a key. A database satisfies the key constraint if for every relation R over σ, whenever two R-facts agree on the key indices they must be equal. A set of *primary key constraints* has, for each relation of σ, a unique key constraint. Notice that if the primary key constraint associated to the relation symbol R contains all the columns of R, then it induces no constraints on R. As all the sets of constraints we consider are primary key constraints we will henceforth omit the 'primary' prefix. We use the letter Γ to denote the corresponding set of key constraints.

Given two facts u and v and a set Γ of key constraints, we say that u and v are *Γ-equivalent*, denoted by $u \sim_\Gamma v$, if u and v have the same associated symbol R and agree on

the key of R as specified by Γ. Γ-equivalence is an equivalence relation and the equivalence classes are called Γ-*blocks*. We will omit Γ in our notations whenever it is clear from the context. A database is then a finite collection of blocks, each block being a finite collection of maximal equivalent facts. When writing a query q we will always underline in an atom $R(\bar{x})$ the positions that are part of the key of R as specified by Γ. This will avoid explicitly describing Γ. For instance $R(\underline{x}\ y)$ says that the first position is the key for the binary relational symbol R; and $R'(\underline{yz}\ y)$ says that the first two positions form the key for the ternary relational symbol R'.

If a database D satisfies the key constraints Γ, denoted by $D \models \Gamma$, then each block of D has size one. If not, then a *repair* of D is a subset of the facts of D such that each block of D has exactly one representative in the repair. In particular a repair always satisfies the key constraints. Notice that there could be exponentially many repairs of a given database D.

A *boolean conjunctive query* over a relational signature σ is a collection of atoms where an *atom* is a term $R(\bar{x})$ where R is a relation symbol from σ and $\bar{x}$ is a tuple of variables of the appropriate arity. The query being boolean, all variables are implicitly existentially quantified. We will consider atoms of a conjunctive query to appear in an arbitrary but fixed order. In this paper a "*query*" is always a boolean conjunctive query. A database D satisfies a query q having atoms $A_1, \ldots, A_k$, denoted by $D \models q$, if there is a mapping μ from the variables of q to the elements of the domain of D such that the fact $\mu(A_i) \in D$ for all i. In this case the sequence $(\mu(A_1), \ldots, \mu(A_k))$ of (not necessarily distinct) facts of D is called a *solution* to q in D. Different mappings yield different solutions. The set of solutions to q in D is denoted by $q(D)$. We will also write $D \models q(\bar{u})$ to denote that the sequence of facts $\bar{u}$ is a solution to q in D. If $\bar{u} = (u_1, \ldots, u_k)$ is a solution to q we also say that u_i *matches* A_i in this solution, and that any subsequence $u_{i_1}, \ldots, u_{i_l}$ matches $A_{i_1}, \ldots, A_{i_l}$.

Let r be a repair of D. By $|q(r)|$ we denote the number of solutions to q in r, *i.e.* the cardinality of $q(r)$. We say that a repair r is *minimal* if there is no repair s such that $|q(s)| < |q(r)|$.

We say that a query q is *certain* for a database D if all repairs of D satisfy q. We study the complexity of determining whether a query is certain for a database D. We adopt the *data complexity* point of view. For each query q and set of key constraints Γ, we denote by $\mathsf{certain}_\Gamma(q)$ (or simply $\mathsf{certain}(q)$ when Γ is understood from the context) the problem of determining, given a database D, whether q is certain for D. Clearly the problem is in coNP as one can guess a (polynomial sized) repair and test whether it does not satisfy q. It is known that for some queries q the problem $\mathsf{certain}(q)$ is coNP-complete [FM07]. However, there are queries q for which $\mathsf{certain}(q)$ is in PTime or even expressible in first-order logic (denoted by FO in the sequel) [KP12, Wij10]. In this context, the following dichotomy has been conjectured (cf [FM07, AK09]):

Conjecture 2.1 (*Dichotomy conjecture*)**.** For each boolean conjunctive query q, the problem $\mathsf{certain}(q)$ is either in PTime or coNP-complete.

The conjecture has been proved in the case of self-join-free queries [KW17] and of path queries [KOW21]; however, it remains open in the general case. A boolean conjunctive query is *self-join-free* if all its atoms involve different relational symbols, otherwise, it is a *self-join query*. A *path query* is a boolean conjunctive query with $n+1$ distinct variables $x_0, x_1, \cdots x_n$ and n atoms $A_1 \cdots A_n$ such that each atom $A_i = R_i(\underline{x_{i-1}}\, x_i)$ for some symbol R_i of σ of arity two. The path query may contain self-joins, in other words it may be the case that $R_i = R_j$ for some $i \neq j$.

Example 2.2. Consider the following example queries taken from [KP12, KOW21] (recall that all variables are implicitly existentially quantified). For the self-join-free boolean query

$$\mathsf{q}_1 \mathrel{\hat{=}} R_1(\underline{x}\ y) \wedge R_2(\underline{y}\ z)$$

it is easy to see that the problem $\mathsf{certain}(\mathsf{q}_1)$ can be solved in polynomial time [KP12]. In fact, the first-order formula $\phi := \exists xyz \left(R_1(xy) \wedge R_2(yz) \wedge \forall y'\big(R_1(xy') \rightarrow \exists z' R_2(y'z')\big)\right)$ is such that for every database instance D, q_1 is certain for D iff $D \models \phi$.

For the self-join-free query and the path query

$$\mathsf{q}_2 \mathrel{\hat{=}} R_1(\underline{x}\ y) \wedge R_2(\underline{y}\ x)$$
$$\mathsf{q}_2' \mathrel{\hat{=}} R(\underline{x_1}\ x_2) \wedge X(\underline{x_2}\ x_3) \wedge R(\underline{x_3}\ x_4) \wedge Y(\underline{x_4}\ x_5) \wedge R(\underline{x_5}\ x_6) \wedge Y(\underline{x_6}\ x_7)$$

it has been shown, in [Wij10] and [KOW21] respectively, that $\mathsf{certain}(\mathsf{q}_2)$ and $\mathsf{certain}(\mathsf{q}_2')$ can be solved in polynomial time but cannot be expressed in first-order logic, unlike $\mathsf{certain}(\mathsf{q}_1)$. The polynomial time algorithm described in the next section computes certain for q_1, q_2 and q_2' (see also Example 3.4).

Finally, for the self-join-free query and the path query

$$\mathsf{q}_3 \mathrel{\hat{=}} R_1(\underline{x}\ y) \wedge R_2(\underline{z}\ y)$$
$$\mathsf{q}_3' \mathrel{\hat{=}} R(\underline{x_1}\ x_2) \wedge X(\underline{x_2}\ x_3) \wedge R(\underline{x_3}\ x_4) \wedge X(\underline{x_4}\ x_5) \wedge R(\underline{x_5}\ x_6) \wedge Y(\underline{x_6}\ x_7) \wedge R(\underline{x_7}\ x_8) \wedge Y(\underline{x_8}\ x_9)$$

both $\mathsf{certain}(\mathsf{q}_3)$ and $\mathsf{certain}(\mathsf{q}_3')$ are known to be coNP-complete [FM07, KOW21]. △

3. Polynomial-time algorithm

To solve $\mathsf{certain}(q)$, we describe a family of algorithms $\mathrm{Cert}_k(q)$, where $k \geq 1$ is a parameter. For a fixed k and query q, $\mathrm{Cert}_k(q)$ takes a database as input and runs in time polynomial in the size of the database, in such a way that $\mathrm{Cert}_k(q)$ is always an under-approximation of $\mathsf{certain}(q)$, *i.e.* whenever $\mathrm{Cert}_k(q)$ says 'yes' then q is certain for the input database. However, $\mathrm{Cert}_k(q)$ could give false negative answers.

In Section 4 and Section 5 we will show that for self-join-free queries either $\mathrm{Cert}_k(q)$ computes $\mathsf{certain}(q)$ (where k is the number of atoms occurring in q) or $\mathsf{certain}(q)$ is coNP-complete in which case $\mathrm{Cert}_k(q)$ fails to compute $\mathsf{certain}(q)$ for every $k \geq 1$. In Section 6 we show an analogous result for path queries.

The algorithm inductively computes sets of facts maintaining the invariant that every repair containing one of these sets makes the query true. The algorithm returns 'yes' if the empty set is eventually derived (since all repairs contain the empty set).

We now describe the algorithm. Assume q, Γ and k are fixed. Let D be a database. A k-set over D is a set S of facts of D of size at most k. We denote by $\mathrm{Cert}_k(q)$ the Algorithm 1 below. On a database input D, the algorithm $\mathrm{Cert}_k(q)$ inductively computes a set $\Delta_k(q, D)$ of k-sets over D while maintaining the following invariant:

For every repair r of D and every $S \in \Delta_k(q, D)$, if $S \subseteq r$ then $r \models q$. (Inv)

Initially $\Delta_k(q, D)$ contains all k-sets S such that $S \models q$. In other words, we start with all solutions to q in all repairs of D. Clearly, this satisfies the invariant (Inv). Now we iteratively add a k-set S to $\Delta_k(q, D)$ if there exists a block B of D such that for every fact

Algorithm 1 $\mathrm{Cert}_k(q)$

Input: database D
initialize $\Delta_k(q, D)$ with all k-sets S of D such that $S \models q$
while there is a k-set $S \notin \Delta_k(q, D)$ and a block B of D such that for every fact $u \in B$ there exists $S' \subseteq S \cup \{u\}$ where $S' \in \Delta_k(q, D)$ **do**
  update $\Delta_k(q, D) \leftarrow \Delta_k(q, D) \cup \{S\}$
end while
if $\emptyset \in \Delta_k(q, D)$ **then**
  return `YES`
else
  return `NO`
end if

$u \in B$ there exists $S' \subseteq S \cup \{u\}$ such that $S' \in \Delta_k(q, D)$. Again, it is immediate to verify that the invariant (Inv) is maintained.

This is an inflationary fixpoint algorithm and notice that the initial and induction steps can be expressed in FO (because all sets of facts S computed by the fixpoint have size at most k and each set can be represented by a tuple of k elements. A relation of arity k can then encode Δ_k). The initial condition adds any k-set that contains a solution to q, and the induction step again adds only k-sets. Thus, if n is the number of facts of D, the fixpoint is reached in $O(n^k)$ steps. In the end, $\mathrm{Cert}_k(q)$ returns 'yes' iff the empty set belongs to $\Delta_k(q, D)$. Equivalently, $\mathrm{Cert}_k(q)$ returns 'yes' if there is a block B of D such that for all facts u of B the set $\{u\}$ belongs to $\Delta_k(q, D)$. We write $D \models \mathrm{Cert}_k(q)$ or $D \in \mathrm{Cert}_k(q)$ to denote that $\mathrm{Cert}_k(q)$ returns 'yes' upon input D.

The following properties are now immediate.

Proposition 3.1. *For all q, Γ, k, $\mathrm{Cert}_k(q)$ runs in time polynomial in the size of its input database D and, if $D \models \mathrm{Cert}_k(q)$ then $D \models \mathsf{certain}(q)$.*

Proposition 3.2. *For all q, Γ, k, k' if $k' \geq k$ then for every database D, if $D \models \mathrm{Cert}_k(q)$ then $D \models \mathrm{Cert}_{k'}(q)$.*

In order to simplify the notations, as we will mostly consider the case where k is the number of atoms in q, we write $\Delta(q, D)$ and $\mathrm{Cert}(q)$ to denote $\Delta_k(q, D)$ and $\mathrm{Cert}_k(q)$ respectively, where k is the number of atoms of q. Also, for a fact u, we sometimes write $u \in \Delta_k(q, D)$ instead of $\{u\} \in \Delta_k(q, D)$. We denote $\Delta_k(q, D, i)$ to be the set i^{th} step of the computation of $\Delta_k(q, D)$. The following proposition is immediate from the definitions.

Proposition 3.3. *For all q, Γ, k, i if $S \in \Delta_k(q, \Gamma, i)$ and $S' \supseteq S$ such that $|S'| \leq k$ then $S' \in \Delta_k(q, \Gamma, i)$.*

Example 3.4. Consider again the query $\mathsf{q}_2 : R_1(\underline{x}\ y) \wedge R_2(\underline{y}\ x)$ from Example 2.2. Let $k = 2$ and consider the execution of $\mathrm{Cert}_2(\mathsf{q}_2)$. For a given input database D, initially $\Delta(\mathsf{q}_2, D)$ contains all pairs of facts $\{R_1(ab), R_2(ba)\}$ such that both $R_1(ab)$ and $R_2(ba)$ are in D. The first iterative step adds to $\Delta(\mathsf{q}_2, D)$ (i) all singletons $\{R_1(ab)\}$ such that $R_2(ba)$ is a fact of D whose block contains only $R_2(ba)$, and (ii) analogously all $\{R_2(ab)\}$ such that the block of $R_1(ba)$ is a singleton.

In subsequent steps, the empty set is added to $\Delta(\mathsf{q}_2, D)$ if at some point, there is some block B such that for every fact $u \in B$ we have $u \in \Delta(\mathsf{q}_2, D)$. At this point the algorithm

outputs 'yes' and if $\Delta(\mathsf{q}_2, D)$ saturates without the empty set as its member, then the algorithm outputs 'no'.

We show that $\mathrm{Cert}_2(\mathsf{q}_2)$ computes $\mathsf{certain}(\mathsf{q}_2)$. In other words, $D \models \mathsf{certain}(\mathsf{q}_2)$ iff $D \models \mathrm{Cert}_2(\mathsf{q}_2)$.

Observe that, for every repair r and fact α therein, there is at most one other fact α' in r such that $\{\alpha, \alpha'\} \models \mathsf{q}_2$. This is because in any repair the first atom of q_2 determines the second atom and vice-versa. This "mutual determinacy" is, in fact, what makes $\mathrm{Cert}_2(\mathsf{q}_2)$ a complete procedure, as we shall see next.

In view of Proposition 3.3, it remains to show that if q_2 is certain for D then $\Delta(\mathsf{q}_2, D)$ contains the empty set. Towards this we use the following observation about q_2.

Claim 3.5. If r is a minimal repair and both facts $R_1(ab)$ and $R_2(ba)$ are in r then $R_1(ab) \in \Delta(\mathsf{q}_2, D)$.

Assume that the claim is true. Now suppose q_2 is certain for D, then for any minimal repair r, we must have $r \models q$ and this is witnessed by two facts $R_1(ab)$ and $R_2(ba)$ of r. Let B be the block of $R_1(ab)$. Let us show that all facts of B are in $\Delta(\mathsf{q}_2, D)$ as singleton sets and hence $\emptyset \in \Delta(\mathsf{q}_2, D)$. Let $R_1(ab')$ be such a fact and consider the repair r' obtained by replacing $R_1(ab)$ with $R_1(ab')$. As r is minimal it follows immediately that r' is minimal and must contain $R_2(b'a)$ (again, this is ensured by the mutual determinacy of q_2). From the claim it follows that $R_1(ab') \in \Delta(\mathsf{q}_2, D)$, as desired. Thus it remains to prove the Claim 3.5.

Proof of Claim 3.5. Assume that r is a minimal repair containing both $R_1(ab)$ and $R_2(ba)$. Towards a contradiction, suppose $R_1(ab) \notin \Delta(\mathsf{q}_2, D)$ then we shall construct an infinite sequence $u_1, u_2, \ldots$ of distinct facts of D, contradicting the finiteness of D. Towards this we additionally construct an infinite sequence $v_1, v_2, \ldots$ of facts of D and an infinite sequence of minimal repairs $r_1, r_2, \ldots$ maintaining the following invariants for every i:

(1) the u_i's are pairwise distinct;
(2) $u_i \notin \Delta(\mathsf{q}_2, D)$;
(3) if $u_i = R_1(cd)$ then $v_i = R_2(dc)$ and if $u_i = R_2(cd)$ then $v_i = R_1(dc)$;
(4) $u_{i+1} \sim v_i$ and $u_{i+1} \neq v_i$;
(5) r_i is minimal and contains v_i and each u_j for all $j \leq i$.

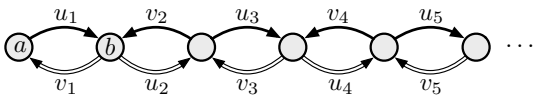
Initially $r_1 = r$, $u_1 = R_1(ab)$ and $v_1 = R_2(ba)$ and all the invariant conditions are met: (1) is trivial, (2) and (5) follow from the assumption, (3) is true by construction and (4) does not apply.

Consider step i. Consider the block B_i of v_i. As $\{u_i\} \notin \Delta(\mathsf{q}_2, D)$ it means that we cannot use any block B as a witness to add u_i to $\Delta(\mathsf{q}_2, D)$ (*i.e.* for every block B there is some fact $w \in B$ such that $\{u_i, w\} \notin \Delta(\mathsf{q}_2, D)$). Hence, in particular, B_i must contain an element u_{i+1} such that both $u_{i+1} \notin \Delta(\mathsf{q}_2, D)$ and $\{u_i, u_{i+1}\} \notin \Delta(\mathsf{q}_2, D)$. In particular $u_{i+1} \sim v_i$ but $u_{i+1} \neq v_i$ and items (2) and (4) of our induction hypothesis are met. Towards the first item of our induction hypothesis, if $u_{i+1} = u_j$ for some $j \leq i$ then by item (5) the repair r_i would contain two equivalent facts, v_i and $u_{i+1} = u_j$, which is not possible since we have already established that $u_{i+1} \neq v_i$.

Consider the repair r_{i+1} resulting from replacing v_i with u_{i+1}. Let v_{i+1} be the dual fact of u_{i+1} as required by the third item of the invariant. As $u_i v_i$ forms a solution to q in r_i and r_i is minimal, we must have $v_{i+1} \in r_{i+1}$ (otherwise r_{i+1} has strictly fewer solutions than r_i). Finally, notice that r_{i+1} is minimal as its solutions to q are exactly the same as for r_i

except for $u_i v_i$ that has been removed and $u_{i+1} v_{i+1}$ that has been added (by the mutual determinacy of the atoms of q_2).

Here is a depiction of how the u_i's and v_i's are defined, where the full and hollow arrows correspond to R_1 and R_2 respectively.



This concludes the construction of the infinite sequence, showing that $R_1(ab) \in \Delta(\mathsf{q}_2, D)$ for any minimal repair containing both $R_1(ab)$ and $R_2(ba)$ which proves claim. △

This concludes our example. △

Cert_k does not always compute the certain answers. For instance, the query q_3 from Example 2.2 is so that $\mathsf{certain}(\mathsf{q}_3)$ is coNP-complete, and hence $\mathrm{Cert}_k(\mathsf{q}_3)$ must have false negatives for all k, under the hypothesis that coNP $\neq$ PTime. Proving this without relying on complexity theoretic assumptions is the goal of Section 5 for self-join-free queries and Section 6.2 for path queries.

4. Tractable self-join-free queries

In this section we consider the case of self-join-free queries. We exhibit a condition named PCond (for Polynomial-time Condition) and show that any self-join-free query q satisfying PCond is such that Cert_k computes $\mathsf{certain}(q)$, where k is the number of atoms in q. When PCond fails, we will see that for all values of k, Cert_k fails to compute $\mathsf{certain}(q)$ and moreover, $\mathsf{certain}(q)$ is coNP-hard.

We start by defining PCond, which will require some extra definitions. Fix, for the rest of this section, a set Γ of primary key constraints. Let D be a database and r a repair of D. For a fact u of r, and for an equivalent fact $v \sim u$ from D, we denote by $r[u \to v]$ the repair obtained from r by replacing the fact u with v.

Consider a self-join-free query q with k atoms. Recall that we write $D \models q(\bar{u})$ when $\bar{u}$ is a solution to q in D. As q is self-join-free, for each fact a in a solution $\bar{u}$ there is a unique atom of q that a can match, namely the only fact of q having the same relation symbol as a. Hence, the order on both $\bar{u}$ and the atoms of q are not relevant. With some abuse of notation we will therefore often treat a solution $\bar{u}$, or the sequence of atoms of q, as a *set* rather than a *sequence*; we will often use different orders among the facts of a solution, placing up front the most relevant facts. Also we shall write, for a tuple $\bar{u}$ of facts, $\bar{u} \in \Delta_k(q, D)$ to denote that the set formed by the facts of $\bar{u}$ is a k-set and belongs to $\Delta_k(q, D)$.

Let A be an atom of q whose associated symbol is R. We denote by $\mathit{vars}(A)$ the set of variables of A and by $\mathit{key}(A)$ the set of variables of A occurring in a position belonging to the primary key of R. For instance $\mathit{key}(R(\underline{x}\ y))$ is $\{x\}$, $\mathit{key}(R'(\underline{yz}\ x))$ is $\{y, z\}$ and $\mathit{key}(R''(\underline{xzx}\ y))$ is $\{x, z\}$.

Given a set X of variables of q and a sequence $A_1 \dots A_n$ of atoms of q, we say that $X\ A_1 \dots A_n$ is a Γ-*derivation* from X to A_n in q if for each $1 \leq i \leq n$ we have that

$$\mathit{key}(A_i) \subseteq X \cup \bigcup_{1 \leq j < i} \mathit{vars}(A_j).$$

If $X = \mathit{vars}(A_0)$, for some atom A_0 of q, we say that the Γ-derivation is from A_0 to A_n, and write it as $A_0 A_1 \dots A_n$. We say that an atom A' is *Γ-determined* by the atom A if there exists a Γ-derivation from A to A'. Moreover, A and A' are *mutually Γ-determined* if A' is Γ-determined by A and A is Γ-determined by A'. This is an equivalence relation among atoms. A set S of atoms of q is called *stable* if every two distinct facts of S are mutually Γ-determined. Note that if an atom A is in a stable set S then S need not contain all the atoms that are mutually Γ-determined by A. So a partition of the atoms of into stable sets is a refinement of the partition induced by mutual Γ-determinacy. Notice also that if two atoms A and B are in a stable set S, we do not require that S contains the atoms witnessing their mutual determinacy. As usual, we will omit Γ when it is clear from the context.

The main intuition on how we will use Γ-derivations is the following. Suppose there is a query q with atoms $A_1, \dots, A_n$ which has solutions in two repairs r, r' of a database, witnessed by valuations μ, μ'. If there is a (one-step) derivation "$X\ A_1$", this means that $\mathit{key}(A_1) \subseteq X$, so if μ, μ' agree on X, in particular they agree on $\mathit{key}(A_1)$ (that is, the corresponding atoms $\mu(A_1)$ and $\mu'(A_1)$ are Γ-equivalent). Further, if $\mu(A_1) \in r'$, we can actually say that μ, μ' agree on all the variables in $\mathit{vars}(A_1)$, since the repairs must necessarily have the same A_1-fact under the key $\mu(\mathit{key}(A_1)) = \mu'(\mathit{key}(A_1))$. Now if we were to add another step in the derivation "$X\ A_1\ A_2$", we further have $\mathit{key}(A_2) \subseteq X \cup \mathit{vars}(A_1)$. Since μ, μ' agree on $X \cup \mathit{vars}(A_1)$, we now have $\mu(A_2)$ and $\mu'(A_2)$ are Γ-equivalent. This key property relating Γ-derivations and query solutions (and extended to arbitrary length derivations) is formally stated in the lemma below.

Lemma 4.1. *Let q be a self-join-free query. Let D be a database instance and r, r' be two repairs of D. Let X be a set of variables of q and let $X\ A_1 \dots A_n$ be a Γ-derivation from X to A_n in q. Let $q(r)$ contain a solution witnessed by a valuation μ of variables of q, and $q(r')$ contain a solution witnessed by a valuation μ'. If μ and μ' agree on X and $\mu(A_i) \in r'$ for all $i < n$, then:*

1) *$\mu(A_i) = \mu'(A_i)$ for each $i < n$ and*
2) *$\mu(A_n) \sim \mu'(A_n)$ (and therefore $\mu(A_n) = \mu'(A_n)$ if moreover $\mu(A_n) \in r'$).*

Proof. The proof is by induction on n. For $n = 1$ the statement trivially holds since $\mathit{key}(A_1) \subseteq X$ thus $\mu(A_1) \sim \mu'(A_1)$.

Now consider a sequence $X A_1 \dots A_n$, $n > 1$ satisfying the hypotheses. The induction hypothesis applied to the sequence $X A_1 \dots A_{n-1}$ implies $\mu(A_i) = \mu'(A_i)$, for all $1 \leq i \leq n-1$. Then μ and μ' agree on $\mathit{vars}(A_1) \cup \dots \cup \mathit{vars}(A_{n-1})$.

Since $X A_1 \dots A_n$ is a Γ-derivation sequence, we have $\mathit{key}(A_n) \subseteq X \cup \mathit{vars}(A_1) \cup \dots \cup \mathit{vars}(A_{n-1})$; hence μ and μ' agree on $\mathit{key}(A_n)$, or in other words $\mu(A_n) \sim \mu'(A_n)$. □

Corollary 4.2. *Let q be a self-join-free query and S be a stable set of atoms of q. Let D be a database instance and r be a repair of D. Assume $r \models q(\bar{\alpha}\bar{a}\bar{\beta})$ and $r \models q(\bar{\alpha}'\bar{a}'\bar{\beta}')$, where $\bar{a}$ and $\bar{a}'$ match S. If $\bar{a} \cap \bar{a}' \neq \emptyset$ then $\bar{a} = \bar{a}'$.*

Proof. The statement follows directly from Lemma 4.1 using $r = r'$. Take a_1 as any fact in $\bar{a} \cap \bar{a}'$, A_1 being the atom matched by a_1, X being $\mathit{vars}(A_1)$ and $A_1 \dots A_k$ being a Γ-derivation sequence containing all S (which exists by stability). □

We are now ready to define PCond. A *Γ-sequence* τ of q is a sequence $\tau = S_1 S_2 \cdots S_n$ where each S_i is a stable set of atoms of q, and the S_i's form a partition of q. In this context, we denote by $S_{\leq i}$ the set $\bigcup_{j \leq i} S_j$. We define S_0 to be the empty set.

Let $\tau = S_1 S_2 \cdots S_n$ be a Γ-sequence of q. Let $1 \leq i < n$ and let A be an atom of S_{i+1}. We say that the query q satisfies $\mathrm{PCond}_\tau(A)$ and write $q \models \mathrm{PCond}_\tau(A)$ if the following is true for all databases D, all repairs r of D and all solutions $\bar{\alpha}u\bar{\beta}$ and $\bar{\alpha}'u'\bar{\beta}'$ to q in D such that $\bar{\alpha}$ and $\bar{\alpha}'$ match $S_{\leq i}$ and u and u' match A:

$$\text{If } \left\{ \begin{array}{l} r \models q(\bar{\alpha}u\bar{\beta}), \\ u \sim u', \text{ and} \\ r[u \to u'] \models q(\bar{\alpha}'u'\bar{\beta}') \end{array} \right\} \text{ then } r \models q(\bar{\alpha}'u\bar{\delta}) \text{ for some sequence of facts } \bar{\delta}.$$

Note that, by symmetry, we also have $r[u \to u'] \models q(\bar{\alpha}u'\bar{\delta}')$ for some sequence of facts δ'. We write $q \models \mathrm{PCond}_\tau(i)$ if q satisfies $\mathrm{PCond}_\tau(A)$ for all A of S_{i+1}, and we write $q \models \mathrm{PCond}_\tau$ if q satisfies $\mathrm{PCond}_\tau(i)$ for all $1 \leq i < n$. Since the condition is restricted to indices $i < n$, PCond_τ trivially holds for any τ having only one stable set. Finally, we write $q \models \mathrm{PCond}$ if there is a Γ-sequence τ of q such that $q \models \mathrm{PCond}_\tau$. Again, if q has only one Γ-determinacy class (for instance the query q_2 of Example 3.4) then $q \models \mathrm{PCond}$ in a trivial way.

We illustrate the definition of PCond with the following examples.

Example 4.3. We recall the three queries from Example 2.2. The query $\mathsf{q}_2 = R_1(\underline{x}\ y) \wedge R_2(\underline{y}\ x)$ satisfies PCond since it has only one maximal stable set.

The query $\mathsf{q}_1 = R_1(\underline{x}\ y) \wedge R_2(\underline{y}\ z)$ has two stable sets: $R_1(\underline{x}\ y)$ determines $R_2(\underline{y}\ z)$ but the converse is false. For $\tau = \{R_2(\underline{y}\ z)\}\{R_1(\underline{x}\ y)\}$ we have $q \not\models \mathrm{PCond}_\tau$ because we have the solutions $\mathsf{q}_1(R_2(bc)R_1(ab))$ and $\mathsf{q}_1(R_2(b'c)R_1(ab'))$ but not $\mathsf{q}_1(R_2(bc)R_1(ab'))$. However for $\tau = \{R_1(\underline{x}\ y)\}\{R_2(\underline{y}\ z)\}$ it is easy to verify that $\mathsf{q}_1 \models \mathrm{PCond}_\tau$. Hence, $\mathsf{q}_1 \models \mathrm{PCond}$.

The query $\mathsf{q}_3 = R_1(\underline{x}\ y) \wedge R_2(\underline{z}\ y)$ has also two stable sets, but no possible sequence τ makes PCond_τ true. This is because (i) $\mathsf{q}_3(R_1(ab)\ R_2(cb))$ and $\mathsf{q}_3(R_1(a'b')\ R_2(cb'))$ hold, but not $\mathsf{q}_3(R_1(ab)\ R_2(cb'))$, and (ii) $\mathsf{q}_3(R_2(ab)\ R_1(cb))$ and $\mathsf{q}_3(R_2(a'b')\ R_1(cb'))$ hold, but not $\mathsf{q}_3(R_2(ab)\ R_1(cb'))$. Therefore, $\mathsf{q}_3 \not\models \mathrm{PCond}$. △

Our goal for the remaining part of this section is to show that $q \models \mathrm{PCond}$ implies that $\mathrm{Cert}(q)$ computes $\mathsf{certain}(q)$ (Recall that $\mathrm{Cert}(q)$ denotes $\mathrm{Cert}_k(q)$ where k is the number of atoms in q) which is given by Theorem 4.4. In Section 5 we will see that when $q \not\models \mathrm{PCond}$ then $\mathsf{certain}(q)$ cannot be computed by Cert_k for any choice of k and that $\mathsf{certain}(q)$ is actually coNP-hard.

Theorem 4.4. *Let q be a self-join-free query with k atoms. If $q \models \mathrm{PCond}$, then $\mathrm{Cert}_k(q)$ computes $\mathsf{certain}(q)$.*

Suppose q has k atoms. Let $\tau = S_1 \cdots S_n$ be a Γ-sequence of q such that $q \models \mathrm{PCond}_\tau$. We show that $\mathrm{Cert}_k(q)$ computes precisely $\mathsf{certain}(q)$.

If q has only one stable set (and thus it trivially satisfies PCond), the proof is similar to the proof of Example 3.4, starting with a minimal repair and exploiting the mutual determinacy of the atoms of q. If q has more stable sets, then the condition of minimality needs to be more fine-grained and we proceed by induction on the index of the stable sets, in the order described by τ.

We start with some extra notations. Recall that $q(r)$ denotes the set of solutions to q in a repair r; we additionally denote by $q_{\leq i}(r)$ the projection of $q(r)$ on the first i stable sets of τ. More precisely

$$q_{\leq i}(r) = \{\bar{v} \mid \exists \bar{u} \in q(r) \text{ s.t. } \bar{v} \text{ is the subset of } \bar{u} \text{ matching } S_{\leq i}\},$$

and if $\bar{v} \in q_{\leq i}(r)$ we write equivalently $r \models q_{\leq i}(\bar{v})$. Let D be a database and r a repair of D. We say that r is *i-minimal* if there is no repair r' such that $q_{\leq i}(r') \subsetneq q_{\leq i}(r)$. We say that a fact u of a database D is *i-compatible*, if it matches some atom of S_i. We will need the limit case when $i = 0$. In that case both S_0 as well as $S_{\leq 0}$ are empty sets (and hence $\mathrm{PCond}_\tau(0)$ is always true), $q_{\leq 0}(r)$ contains only the empty sequence ε for all r, and therefore all repairs are 0-minimal. The proof of the theorem makes use of an induction based on the following invariant property of the database, for each $0 \leq i \leq n$:

$\textsc{Ind}_i$ = For all i-minimal repairs s and facts $\bar{u}$ s.t. $s \models q_{\leq i}(\bar{u})$, we have $\bar{u} \in \Delta_k(q, D)$.

Lemma 4.5. *Given q, D and a Γ-sequence τ for q, for every $0 \leq i < n$, if $\textsc{Ind}_{i+1}$ and $\mathrm{PCond}_\tau(i)$, then $\textsc{Ind}_i$.*

We first show how this statement already implies Theorem 4.4.

Proof of Theorem 4.4. From Proposition 3.3, we know that if D is a database such that $D \models \mathrm{Cert}_k(q)$ then all repairs of D satisfy q. It remains to show the converse.

Assume all repairs satisfy q and that $q \models \mathrm{PCond}_\tau$ for some sequence τ of length n, which means that $\mathrm{PCond}_\tau(i)$ holds for all i. Observe that $\textsc{Ind}_n$ holds true by the base case definition of $\Delta_k(q, D)$. Hence by n repeated applications of Lemma 4.5 we obtain that $\textsc{Ind}_0$ holds true. Now take any repair r. By definition r is 0-minimal and by hypothesis it satisfies the query q. By $\textsc{Ind}_0$ it follows that the empty set (denoted by the empty tuple) is in $\Delta_k(q, D)$, and hence $D \models \mathrm{Cert}_k(q)$. □

We are now left with the proof of Lemma 4.5, which is the main technical content of the section. Towards this, we define a stronger version of i-minimality. For $1 \leq i < n$, we say that an i-minimal repair s is *strong i-minimal* if there exists no repair s' such that $q_{\leq i}(s') = q_{\leq i}(s)$ and $|q_{\leq i+1}(s')| < |q_{\leq i+1}(s)|$. Note that if $q_{\leq i+1}(s') \subsetneq q_{\leq i+1}(s)$ then either $q_{\leq i}(s') \subsetneq q_{\leq i}(s)$ or $q_{\leq i}(s') = q_{\leq i}(s)$ but $|q_{\leq i+1}(s')| < |q_{\leq i+1}(s)|$. In particular, every strong i-minimal repair is $(i+1)$-minimal.

Claim 4.6. If there exists an i-minimal repair s such that $s \models q_{\leq i}(\bar{u})$, then there exists a strong i-minimal repair s' such that $s' \models q_{\leq i}(\bar{u})$.

Proof. Among all repairs s'' having $q_{\leq i}(s'') = q_{\leq i}(s)$, choose s' as having minimal $|q_{\leq i+1}(s')|$. In other words, s' is a repair having $q_{\leq i}(s') = q_{\leq i}(s)$ and for every repair s'' with $q_{\leq i}(s'') = q_{\leq i}(s)$ we have $|q_{\leq i+1}(s')| \leq |q_{\leq i+1}(s'')|$. Hence, s' is strong i-minimal. □

For a given database D, for a repair r of D, we denote by $r_{|i+1}$ the set of facts of r which are not $(i+1)$-compatible. A sequence $\bar{p}$ of facts of the database is *connected* with respect to $D' \subseteq D$ if for every repair r containing $\bar{p}$ and D', and for every two consecutive facts $a\,b$ of $\bar{p}$, if $r \models q(\bar{\alpha}a\bar{\beta})$ for some $\bar{\alpha}, \bar{\beta}$, then $b \in \bar{\alpha}\bar{\beta}$. Note that if $\bar{p}$ is the empty tuple (or a tuple of size 1), then $\bar{p}$ is trivially connected with respect to every $D' \subseteq D$.

Proof of Lemma 4.5. By contradiction, suppose the statement of the lemma is false. Then, there is some i such that $\textsc{Ind}_{i+1}$ and $\mathrm{PCond}_\tau(i)$ holds, but for some i-minimal repair s and tuple $\bar{u}$ we have

$$s \models q_{\leq i}(\bar{u}) \text{ but } \bar{u} \notin \Delta_k(q, D). \tag{h1}$$

From Claim 4.6, we can assume that s is strong i-minimal. We will build an infinite sequence of pairwise distinct facts $p_1, p_2, \ldots$ from D, contradicting the finiteness of D. We

also maintain another sequence of repairs $r_1, r_2, \ldots$. We set $r_0 = s$. For all $l > 0$, we define $\bar{p}_l = p_1, \ldots, p_l$ with $\bar{p}_0$ as the empty fact sequence.

The sequence is constructed by induction with the following invariant for every $l \geq 0$, assuming $\bar{p} = \bar{p}_l$ and $r = r_l$:

(a) $\bar{p}$ contains only $(i+1)$-compatible facts of D;
(b) the elements of $\bar{p}$ are pairwise distinct;
(c) $\bar{p}$ is connected with respect to $r_{|i+1}$;
(d) r is strong i-minimal, $r \models q_{\leq i}(\bar{u})$ and, if $\bar{p}$ is not empty and v is the last fact of $\bar{p}$, then $r \models q(\bar{u}v\bar{\delta})$, for some $\bar{\delta}$;
(e) $\bar{u}\bar{c} \notin \Delta_k(q, D)$, where $\bar{c}$ is the maximal suffix of $\bar{p}$ satisfying $r' \models q(\bar{u}\bar{c}\bar{\beta})$ for some $\bar{\beta}$ and strong i-minimal repair r' containing $\bar{p}$ and $r_{|i+1}$.

Base case. When $l = 0$, we have $r_0 = s$ and $\bar{p}$ is the empty sequence. Hence (a), (b) and (c) are trivially true by emptiness of $\bar{p}$; (d) holds since $s \models q_{\leq i}(\bar{u})$ (note that we have assumed s to be strong i-minimal); finally (e) holds with empty $\bar{c}$ since $\bar{u} \notin \Delta_k(q, D)$ by (h1).

Induction step. Assume we have $r = r_{l-1}$ and $\bar{p}_{l-1} = p_1, \ldots, p_{l-1}$ (possibly empty) satisfying the five properties above (for the rest of the proof we denote $\bar{p}_{l-1}$ as $\bar{p}$ for brevity). Consider the maximal suffix $\bar{c}$ concerned by property (e). That is, for some $\bar{\beta}$ and strong i-minimal repair r' containing $\bar{p}$ and $r_{|i+1}$ we have:

$$\bar{u}\bar{c} \notin \Delta_k(q, D) \text{ and } r' \models q(\bar{u}\bar{c}\bar{\beta}) \tag{h2}$$

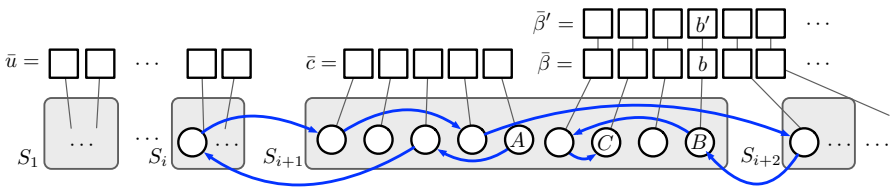
First let $\bar{\beta} = d_1 d_2, \ldots d_t$. Since $\bar{u}\bar{c} \notin \Delta_k(q, D)$, by definition of $\Delta_k(q, D)$ there exists some $d_1' \sim d_1$ such that $\bar{u}\bar{c}d_1' \notin \Delta_k(q, D)$. This again implies that there exists some $d_2' \sim d_2$ such that $\bar{u}\bar{c}d_1'd_2' \notin \Delta_k(q, D)$. Since k is the number of atoms in q, we can continue this to obtain $\bar{\beta}' = d_1'd_2' \ldots d_t'$ where $d_i' \sim d_i$ but $\bar{u}\bar{c}\bar{\beta}'$ contains no k-set in $\Delta_k(q, D)$.

We now show that $\bar{c}$ cannot match the entire set S_{i+1}. Suppose, by means of contradiction, that $r' \models q_{\leq i+1}(\bar{u}\bar{c})$. As r' is strong i-minimal, it is $(i+1)$-minimal. Hence by (h1), since IND_{i+1} holds by hypothesis, $\bar{u}\bar{c} \in \Delta_k(q, D)$, which is in contradiction with (h2). Then,

$$r' \models \neg q_{\leq i+1}(\bar{u}\bar{c}). \tag{h3}$$

This means that, since $r' \models q(\bar{u}\bar{c}\bar{\beta})$ by (h2), and $\bar{c}$ matches a subset of S_{i+1}, there must be an atom C of S_{i+1} that is not matched by any fact of $\bar{c}$. Consider the atom A of S_{i+1} matching the last element of $\bar{c}$. If instead $\bar{c}$ is empty, choose A as an arbitrary atom of S_{i+1}.

Since A and C are both in S_{i+1}, which is stable, there exists a Γ-derivation σ from A to C. (Notice that σ may contain atoms outside S_{i+1}.) Consider the first atom B of σ which is in S_{i+1} and which is not matched by any fact of $\bar{c}$. The following depiction may help to see the situation:



In the picture directed edges connecting atoms of the query represent the successor relation in the Γ-derivation from A to C.

Let b be the fact of $\bar{\beta}$ matching B and $b' \sim b$ be the corresponding fact in $\bar{\beta}'$. We show that

$$b \notin \bar{p}. \tag{h4}$$

Suppose b is in $\bar{p}$. By construction, b is not in $\bar{c}$, thus it must occur before $\bar{c}$ in $\bar{p}$ and hence the suffix $\bar{b}$ of $\bar{p}$ starting with b strictly contains $\bar{c}$. By connectedness of $\bar{p}$ with respect to $r_{|i+1}$, as $b \in \bar{\beta}$ is part of the solution $\bar{u}\bar{c}\bar{\beta}$, we must have $r' \models q(\bar{u}\bar{b}\bar{\gamma})$ for some $\bar{\gamma}$. This contradicts the maximality of $\bar{c}$ imposed by (e), thus proving that (h4) holds. Note that this also implies $b' \notin \bar{p}$, otherwise if $b' \in \bar{p}$, we have that $b' \sim b$ are both in r', thus $b = b' \in \bar{p}$, contradicting (h4).

Assign $p_l = b'$, so we have $\bar{p}' = \bar{p} \cdot p_l$ and let $r_l = r'[b \to b']$. (To avoid many subscripts, let $r_l = s'$). Observe that

$$s' \text{ contains } \bar{p}' \text{ and } r_{|i+1}. \tag{h5}$$

In fact s' contains $\bar{p}$, as observed earlier, and s' contains b' by construction; moreover s' contains $r'_{|i+1}$ which contains $r_{|i+1}$ by (e). We now show that $\bar{p}'$ and s' have all the desired properties.

(a) By construction b' is $(i+1)$-compatible.

(b) The elements of $\bar{p}'$ are pairwise distinct, as $b' \notin \bar{p}$.

(c) By our choice of b we show that $\bar{p}'$ is connected with respect to $s'_{|i+1}$. Without loss of generality assume that $\bar{p}'$ has at least size 2 (otherwise it is trivially connected). Therefore, $\bar{p}$ is not empty. Since $s'_{|i+1}$ contains $r_{|i+1}$ by (h5), the connectedness property of $\bar{p}$ with respect to $r_{|i+1}$ implies that for every repair containing $\bar{p}'$ and $s'_{|i+1}$ and for every pair of consecutive facts $a\,b$ in $\bar{p}$, every solution in s' containing a also contains b.

It remains to show the same property for the last fact a of $\bar{p}$. Consider a repair t containing $\bar{p}'$ and $s'_{|i+1}$ and suppose $t \models q(\bar{\gamma}a\bar{\delta})$ for some $\bar{\gamma}$ and $\bar{\delta}$. We have to show $b' \in \bar{\gamma}\bar{\delta}$. Let σ_{AB} be the prefix of the Γ-derivation σ going from A to B in the Γ-derivation from A to C. (Notice that, since $\bar{p}$ is not empty we have $A \neq B$.) By property (d) of $\bar{p}$, since $\bar{p}$ is not empty, a is the last fact in $\bar{c}$. Recall that by (h2) $r' \models q(\bar{u}\bar{c}\bar{\beta})$; thus in this solution the atom A is matched by a. So we can apply Lemma 4.1 to r' and t with solutions $(\bar{u}\bar{c}\bar{\beta})$ and $(\bar{\gamma}a\bar{\delta})$ respectively, and Γ-derivation $vars(A)$ σ_{AB}. The hypotheses of Lemma 4.1 are satisfied since:

- in both solutions A is matched by a;
- by construction of B, for each atom D strictly preceding B in σ_{AB}, the fact matching D in $(\bar{u}\bar{c}\bar{\beta})$ is either in $\bar{c}$ or in $r'_{|i+1}$, both contained in t (in fact $t \supseteq \bar{p}' \supseteq \bar{p} \supseteq \bar{c}$ and $t \supseteq s'_{|i+1} \supseteq r'_{|i+1}$).

We conclude, by Lemma 4.1, that the facts matching B in the two solutions are equivalent, *i.e.*, the fact matching B in $(\bar{\gamma}a\bar{\delta})$ is equivalent to b (which is the fact matching B in $\bar{u}\bar{c}\bar{\beta}$).

In t the unique fact equivalent to b is b' (since $b' \in \bar{p}' \subseteq t$), thus the fact matching B in $(\bar{\gamma}a\bar{\delta})$ is b'. We have thus proved that any solution in t containing the last fact a of p also contains b'.

(d) The following claim, together with strong i-minimality of r' and $r' \models q(\bar{u}b\bar{\gamma})$ for some $\bar{\gamma}$, shows that

(I) s' is also strong i-minimal,
(II) $s' \models q_{\leq i}(\bar{u})$, and

(III) $s' \models q(\bar{u}b'\bar{\delta})$ for some $\bar{\delta}$.

Claim 4.7. Assume $\mathrm{PCond}_\tau(i)$. Let s be a strong i-minimal repair such that $s \models q(\bar{\alpha}a\bar{\beta})$ where $\bar{\alpha}$ matches $S_{\leq i}$ and a is $(i+1)$-compatible. Then for any $a' \sim a$ we have that $s' = s[a \mapsto a']$ is strong i-minimal and $s' \models q(\bar{\alpha}a'\bar{\delta})$ for some $\bar{\delta}$.

Proof. Notice that s and s' agree on all their solutions to q that contain neither a nor a'. Hence, if a' is in no solution for q in s', we have $q_{\leq i+1}(s') \subsetneq q_{\leq i+1}(s)$ and therefore $|q_{\leq i+1}(s')| < |q_{\leq i+1}(s)|$ and $q_{\leq i}(s') \subseteq q_{\leq i}(s)$. The latter implies $q_{\leq i}(s') = q_{\leq i}(s)$ by i-minimality of s. This contradicts strong i-minimality of s. Hence $s' \models q(\bar{\alpha}'a'\bar{\beta}')$, for some $\bar{\alpha}'$, $\bar{\beta}'$. By $\mathrm{PCond}_\tau(i)$ this implies that $s' \models q(\bar{\alpha}a'\bar{\delta})$ for some $\bar{\delta}$.

It remains to prove that s' is strong i-minimal. To this end, we exhibit a bijection from $q_{\leq i+1}(s)$ to $q_{\leq i+1}(s')$ preserving the $(S_1 \cup \cdots \cup S_i)$-projection of the solutions. The existence of such bijection implies $q_{\leq i}(s) = q_{\leq i}(s')$ and $|q_{\leq i+1}(s)| = |q_{\leq i+1}(s')|$, thus showing that s' is strong i-minimal, provided s is too.

The mapping is the identity for the solutions that do not contain the fact a.

It remains to map bijectively solutions in s containing the fact a to solutions in s' containing the fact a'. Let $\bar{a}$ (resp. $\bar{a}'$) be the facts matching S_{i+1} in $\bar{\alpha}a\bar{\beta}$ (resp. $\bar{\alpha}'a'\bar{\beta}'$). By Corollary 4.2 in all solutions of s containing a, S_{i+1} is matched by $\bar{a}$. Similarly in all solutions of s' containing a', S_{i+1} is matched by $\bar{a}'$.

Moreover, by $\mathrm{PCond}_\tau(i)$ for each $\bar{u}$, $\bar{u}\bar{a} \in q_{\leq i+1}(s)$ iff $\bar{u}\bar{a}' \in q_{\leq i+1}(s')$. Hence mapping each $\bar{u}\bar{a} \in q_{\leq i+1}(s)$ to $\bar{u}\bar{a}' \in q_{\leq i+1}(s')$ forms a bijection. □

(e) Let $\bar{e}$ be the maximal suffix of $\bar{p}'$ such that, for a strong i-minimal repair t containing $\bar{p}'$ and $s'_{|i+1}$ we have $t \models q(\bar{u}\bar{e}\bar{\delta})$ for some $\bar{\delta}$. Since $s' \models q(\bar{u}b'\bar{\delta}')$ for some $\bar{\delta}'$ by Item (III) above, $\bar{e}$ cannot be empty. Then let $\bar{e} = \bar{d}b'$, where $\bar{d}$ is a suffix of $\bar{p}$.

Since $s'_{|i+1}$ contains $r_{|i+1}$ by (h5), in particular t is a strong i-minimal repair containing $\bar{p}$ and $r_{|i+1}$. Then, by maximality of $\bar{e}$, $\bar{d}$ must be a suffix of $\bar{c}$, implying that $\bar{u}\bar{d}b'$ is a subset of $\bar{u}\bar{c}\bar{\beta}'$. Since by definition $\bar{u}\bar{c}\bar{\beta}'$ does not contain any k-set in $\Delta_k(q, D)$, we have $\bar{u}\bar{d}b' \notin \Delta_k(q, D)$ as needed.

This completes the proof of Lemma 4.5. □

5. Lower bounds

Now we turn to the lower bounds. In particular, we consider the self-join-free queries q not satisfying PCond and show that for such queries, $\mathsf{certain}(q)$ is coNP-hard. Towards proving this we also obtain other results of independent interest. The results can be summarized as follows.

(1) We first consider the query with self-join

$$\mathsf{q}_4 \doteq R(\underline{x}\, yz) \wedge R(\underline{z}\, xy).$$

We first show, in Theorem 5.1, via a combinatorial argument, that there is no k such that $\mathsf{certain}(\mathsf{q}_4)$ can be computed using $\mathrm{Cert}_k(\mathsf{q}_4)$ (*i.e.* for every k there exists a database D such that $D \models \mathsf{certain}(q)$ but $\mathrm{Cert}_k(\mathsf{q}_4)$ ouputs a false negative for D).

On the other hand we prove, in Theorem 5.9, that $\mathsf{certain}(\mathsf{q}_4)$ is equivalent (modulo LogSpace reductions) to a matching problem whose precise complexity is a long-standing open problem. This confirms the difficulty to obtain a complete complexity classification of $\mathsf{certain}(q)$ for self-join queries q even on two atoms.[2]

(2) In Section 5.2 we show that for the self-join-free query

$$\mathsf{q}_5 \doteq R_1(\underline{x}\, y) \wedge S_1(\underline{yz}\, x),$$

$\mathsf{certain}(\mathsf{q}_5)$ cannot be computed using $\mathrm{Cert}_k(\mathsf{q}_5)$ for any choice of k. This is shown by reducing the case of q_5 to the case of q_4.

(3) In Section 5.3 we describe some of the techniques developed in [KW17] and show that they imply that for self-join-free queries q, $\mathsf{certain}(q)$ is coNP-hard when PCond fails for q. In particular, assuming PTime $\neq$ coNP, this implies that when $q \models \neg\mathrm{PCond}$, $\mathsf{certain}(q)$ cannot be solved using $\mathrm{Cert}_k(q)$, for any choice of k.

(4) Finally, in Section 5.4 we prove, without any complexity theoretic hypothesis, that for any self-join-free query q such that $q \models \neg\mathrm{PCond}$, that $\mathsf{certain}(q)$ cannot be computed using $\mathrm{Cert}_k(q)$ for any choice of k by reducing the case of q_5 to such query q.

5.1. **The case of q_4.** Note that the query q_4 is not a self-join-free query. In particular, we can have $D \models \mathsf{q}_4(aa)$ for some fact $a \in D$. We call such solutions *self-loops*.

To prove that $\mathrm{Cert}_k(\mathsf{q}_4)$ does not compute $\mathsf{certain}(\mathsf{q}_4)$ for any choice of k, we actually prove a stronger statement: not even an extension "$\mathrm{Cert}^+_k(\mathsf{q}_4)$" of $\mathrm{Cert}_k(\mathsf{q}_4)$ can capture $\mathsf{certain}(\mathsf{q}_4)$. This stronger form will be needed later for reducing q_5 to q_4.

Let us first explain the extension, which is tailored to two-atom queries (thus, strictly speaking, it 'extends' $\mathrm{Cert}_k(\mathsf{q}_4)$ only in this context). Recall that the definition of $\mathrm{Cert}_k(q)$ iteratively adds a k-set S to $\Delta_k(q, D)$ if there exists a block B of D such that for every fact $u \in B$ there exists $S' \subseteq S \cup \{u\}$ such that $S' \in \Delta_k(q, D)$. This rule is henceforth called the *first derivation rule*.

The algorithm $\mathrm{Cert}^+_k(q)$ initializes the set $\Delta^+_k(q, D)$ of k-sets as in $\mathrm{Cert}_k(q)$, and inherits the aforementioned derivation rule of $\mathrm{Cert}_k(q)$, but it also contains the following *second derivation rule*. A k-set S is also added to $\Delta^+_k(q, D)$ if there exists a fact a of D which is not a self-loop and for every fact $u \in \{b \in D : D \models q(ab) \vee q(ba) \vee a = b\}$ there exists $S' \subseteq S \cup \{u\}$ such that $S' \in \Delta^+_k(q, D)$. As before, we define $\mathrm{Cert}^+_k(q)$ to accept if the empty set is eventually derived and denote it by $D \in \mathrm{Cert}^+_k(q)$.

Note that, like $\mathrm{Cert}_k(q)$, $\mathrm{Cert}^+_k(q)$ also runs in polynomial time but it no longer satisfies the inductive property (Inv) and may therefore give false positive answers. However, since the first derivation rule is present in both $\mathrm{Cert}_k(q)$ and $\mathrm{Cert}^+_k(q)$, whenever $\emptyset \in \mathrm{Cert}_k(q)$ we also have $\emptyset \in \mathrm{Cert}^+_k(q)$ *i.e.*if $D \in \mathrm{Cert}^+_k(q)$ then $D \in \mathrm{Cert}_k(q)$. We now formally state the first result of this subsection:

Theorem 5.1. *For every choice of k, there exists a database D such that $D \models \mathsf{certain}(\mathsf{q}_4)$ but $D \notin \mathrm{Cert}^+_k(\mathsf{q}_4)$ (and hence $D \notin \mathrm{Cert}_k(\mathsf{q}_4)$).*

Before proving Theorem 5.1 we discuss some special properties of the query q_4. Note that q_4 is a self-join two-atom query. We define the *solution graph* of D, denoted by G_D, to be an undirected graph whose vertices are the facts of D and it contains an edge $\{a, b\}$

[2]The dichotomy for two-atom queries has recently been proved in [PSS24]. Remarkably, the polynomial cases are solved via a combination of the Cert_k algorithm and the bipartite matching algorithm.

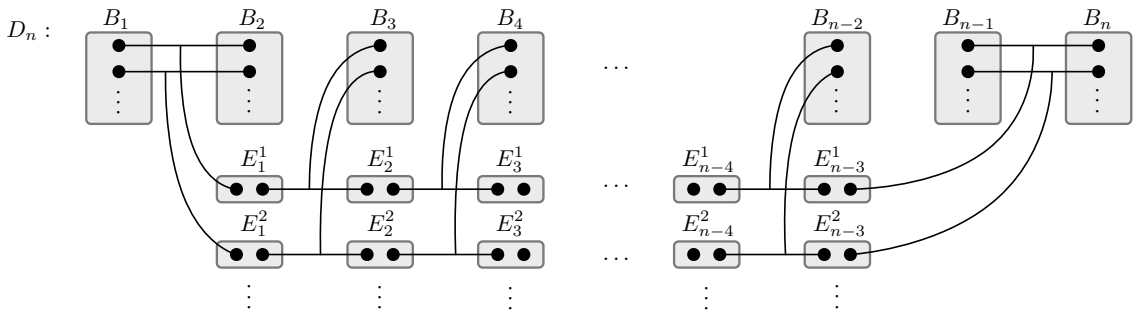


FIGURE 1. Solution graph for database D_n. Black dots denote facts, rectangles denote blocks, and three-pointed edges denote triangles (*i.e.*, 3-cliques) in the solution graph of D_n. There are $n-1$ facts in each block B_i and two facts in each block E^j_k.

whenever $D \models \mathsf{q}_4(ab)$ holds. In the context of solution graphs, a *triangle* is just a clique on three vertices (without self-loops).

Remark 5.2. We first state some key properties of q_4 for any database D and facts $a, b, c \in D$, which are easy to verify:

(1) if $D \models \mathsf{q}_4(ab) \wedge \mathsf{q}_4(ac)$ then $b = c$,
(2) if $D \models \mathsf{q}_4(ba) \wedge \mathsf{q}_4(ca)$ then $b = c$,
(3) if $D \models \mathsf{q}_4(ab) \wedge \mathsf{q}_4(bc)$ then $D \models \mathsf{q}_4(ca)$,
(4) hence, every connected component of G_D is either a triangle, a 2-clique (without self-loops), or a single vertex (with or without a self-loop).

Towards proving Theorem 5.1, for every $n \geq 4$ we exhibit a database D_n such that

- $D_n \models \mathsf{certain}(\mathsf{q}_4)$ (Proposition 5.3), and
- $D_n \notin \mathrm{Cert}^+_{n-2}(\mathsf{q}_4)$ (Proposition 5.4).

Intuitively, the database D_n has two kinds of blocks. The first kind has n blocks denoted by $B_1, \cdots, B_n$ where each B_i consists of $n-1$ facts denoted $b^1_i, \cdots, b^{n-1}_i$. The second kind has $(n-1)(n-3)$ blocks denoted by E^j_l for every $1 \leq j \leq n-1$ and $1 \leq l \leq n-3$, where each E^j_l has two facts denoted by u^j_l and v^j_l. The solution graph of D_n is depicted in Figure 1.

Definition of D_n. We now formally define the facts of the database D_n. Fix some $n \geq 4$. Define n blocks of the form $B_1, \cdots, B_n$ in D_n where each B_i consists of $n-1$ facts denoted $b^1_i, \cdots, b^{n-1}_i$ and $(n-1)(n-3)$ blocks of the form E^j_l for $1 \leq j \leq n-1, 1 \leq l \leq n-3$, where each E^j_l have two facts denoted by u^j_l and v^j_l. All facts in D_n are R-facts, and for this reason we shall henceforth drop the R from facts $R(\bar{c})$ and simply write $\bar{c}$.

Let $a_1, \cdots, a_n$ and e^j_i where $1 \leq j \leq n-1, 1 \leq i \leq n-3$ be fresh active domain elements. The facts of D_n are defined as follows.

- $b^j_1 = (\underline{a_1}\ a_2 e^j_1)$, $b^j_2 = (\underline{a_2}\ e^j_1 a_1)$;
- for every $3 \leq i \leq n-2$: $b^j_i = (\underline{a_i}\ e^j_{i-2} e^j_{i-1})$;
- $b^j_{n-1} = (\underline{a_{n-1}}\ e^j_{n-3} a_n)$ and $b^j_n = (\underline{a_n}\ a_{n-1} e^j_{n-3})$;
- $u^j_1 = (\underline{e^j_1}\ a_1 a_2)$ and $v^j_{n-3} = (\underline{e^j_{n-3}}\ a_n a_{n-1})$;

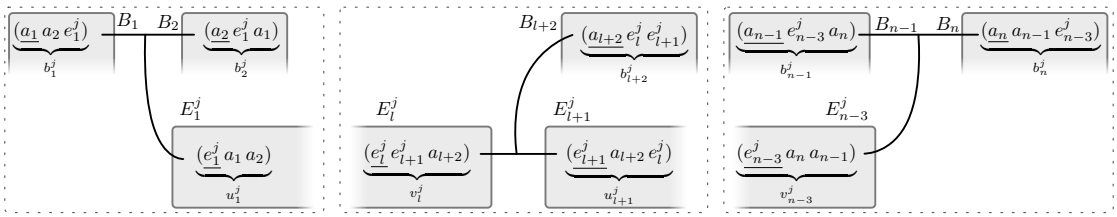


FIGURE 2. Triangles in the database D_n.

- for every $1 \leq l < n-3$: $v_l^j = (\underline{e_l^j}\ e_{l+1}^j a_{l+2})$ and $u_{l+1}^j = (\underline{e_{l+1}^j}\ a_{l+2} e_l^j)$.

As shown in Figure 2, it can be verified that for every $1 \leq j \leq n-1$ we have the triangles $\{b_1^j, b_2^j, u_1^j\}$ and $\{b_{n-1}^j, b_n^j, v_{n-3}^j\}$ and for every $1 \leq l < n-3$ we have a triangle $\{v_l^j, u_{l+1}^j, b_{l+2}^j\}$. Thus we get the solution graph described in Figure 1. Notice that D_n is defined in such a way that every fact of D_n is part of a triangle.

Proposition 5.3. *For every $n \geq 4$, $D_n \models \mathsf{certain}(\mathsf{q}_4)$.*

Proof. Fix some $n \geq 4$ and consider the database D_n. Let $B_1, \ldots B_n$ be the blocks of D_n. Now for every $1 \leq j \leq n-1$ if a repair r contains two indices i, i' such that $i \neq i'$ and $b_i^j, b_{i'}^j \in r$ then it is easy to verify that $r \models q$ (*cf.* Figure 1).

Now since each B_i contains exactly $n-1$ facts of the form $b_i^1, \ldots, b_i^{n-1}$, by the pigeonhole principle, every repair r must contain two i, i' such that $i \neq i'$ and $b_i^j, b_{i'}^j \in r$. Hence, every repair r of D_n verifies $r \models q$. □

It remains to prove the following:

Proposition 5.4. *Let $k \geq 2$. $D_{k+2} \notin \mathrm{Cert}_k^+(\mathsf{q}_4)$.*

For showing this, we first need to set up some definitions and prove some useful properties. When D_n is clear from the context, we denote $\mathbb{B} \triangleq \{B_1, \ldots B_n\}$ and, for every $1 \leq j \leq n-1$, we denote $\mathbb{E}^j \triangleq \{E_l^j \mid 1 \leq l \leq n-3\}$ where the B_i's and E_l^j's are the blocks of D_n defined above.

If $\mathbb{X} = \{X_1, \ldots, X_k\}$ is a set of blocks of D_n, a set of facts $W = \{w_1, \ldots w_k\}$ is called a *partial repair* of $\mathbb{X}$ if $w_i \in X_i$ for every $1 \leq i \leq k$. We denote by $W[\mathbb{E}^j]$ to be the set of facts from W in the blocks $\mathbb{X} \cap \mathbb{E}^j$ (for $1 \leq j \leq n-1$), and $W[\mathbb{B}]$ to be the set of facts from W in the blocks $\mathbb{X} \cap \mathbb{B}$.

Recall that for every $3 \leq l \leq n-2$ and every $1 \leq j \leq n-1$ we have $b_l^j \in B_l$ and the triangle $\{b_l^j, u_{l-1}^j, v_{l-2}^j\}$. For each such j and l we define $U(j,l) \triangleq \{u_k^j \mid 1 \leq k \leq l-2\}$ and $V(j,l) \triangleq \{v_k^j \mid l-1 \leq k \leq n-3\}$, which are depicted in Figure 3.

Intuitively, $U(j,l)$ and $V(j,l)$ represent the facts that need to be picked in blocks of $\mathbb{E}^j$ if one wants to construct a repair of D_n containing b_l^j and having no query solutions. In fact note that $U(j,l) \cup V(j,l)$ picks exactly one fact for every block in $\mathbb{E}^j$. But $u_{l-1}^j,\ v_{l-2}^j \notin U(j,l) \cup V(j,l)$. Consequently, there are no solutions in the set of facts $U(j,l) \cup V(j,l) \cup \{b_l^j\}$ (Figure 3).

In the same spirit, recall that for every $1 \leq j \leq n-1$, D_n contains the triangle $\{b_1^j, b_2^j, u_1^j\}$, so if a repair contains either b_1^j or b_2^j and contains no solution to q_4, this repair

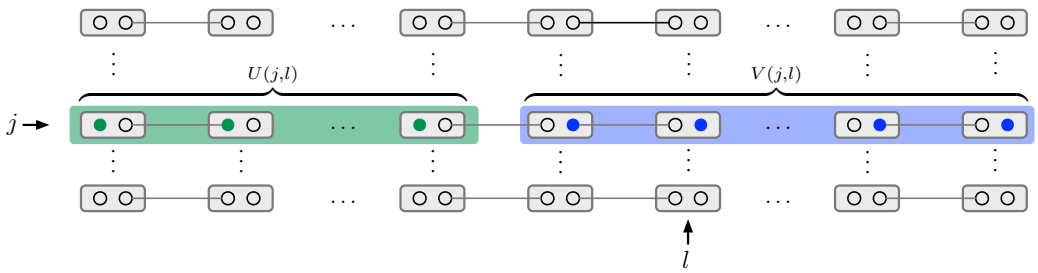


FIGURE 3. Depiction of $U(j,l)$ as green solid discs, and $V(j,l)$ as blue solid discs.

must pick no u_i^j fact in the blocks of $\mathbb{E}^j$. Thus, we define $U(j,1) = U(j,2) = \emptyset$ and $V(j,1) = V(j,2) = \{v_k^j \mid 1 \leq k \leq n-3\}$, so that there are no solutions in the set of facts $U(j,l) \cup V(j,l) \cup \{b_l^j\}$, for $l \in \{1,2\}$.

Dually, $\{b_{n-1}^j, b_n^j, v_{n-3}^j\}$ forms a triangle, so define $U(j,n) = U(j,n-1) = \{u_k^j \mid 1 \leq k \leq n-3\}$ and $V(j,n) = V(j,n-1) = \emptyset$. Again there are no solutions in the set of facts $U(j,l) \cup V(j,l) \cup \{b_l^j\}$, for $l \in \{n-1,n\}$.

Overall, for every $1 \leq l \leq n$ and every $1 \leq j \leq n-1$ we have:

$U(j,l) \cup V(j,l) \cup \{b_l^j\}$ forms a partial repair over the blocks $\{B_j\} \cup \mathbb{E}^j$, and

$U(j,l) \cup V(j,l) \cup \{b_l^j\}$ contains no solution.

A set of facts W of size k is called a *k-obstruction set* if W is a partial repair of some $\mathbb{X} = \{X_1, \ldots X_k\}$ and if $\mathbb{X} \cap \mathbb{B} = \{B_{i_1}, B_{i_2} \ldots B_{i_l}\}$ for some $l \leq k$ with $W[\mathbb{B}] = \{b_{i_1}^{j_1}, b_{i_2}^{j_2} \ldots b_{i_l}^{j_l}\}$, then the following conditions hold:

(1) $j_1, j_2, \ldots j_l$ are pairwise distinct.
(Informally: elements of W in $\mathbb{B}$ are in different 'rows' in Figure 1.)
(2) For every $1 \leq m \leq l$ we have $W[\mathbb{E}^{j_m}] \subseteq U(j_m, i_m) \cup V(j_m, i_m)$.
(Informally: if W contains an element at row j_m connected to the block B_{i_m} then it must be painted green or blue in Figure 3, under the renaming $j \mapsto j_m$ and $l \mapsto i_m$.)
(3) For every $1 \leq j \leq n-1$ there exists $1 \leq l \leq n$ such that $W[\mathbb{E}^j] \subseteq U(j,l) \cup V(j,l)$.
(Informally: all the elements of W at a row j must be painted green or blue in Figure 3, for some choice of indices.)

Lemma 5.5. *If b is a fact in D_{k+2} then there always exists a k-obstruction set that contains b.*

Proof. Let B be the block such that $b \in B$. Now if B is of the form B_i then let $b = b_i^j$ and we can choose the $k-1$ other blocks of the form $B_{i'}$ and pick b_i^j from B_i and pick $b_{i'}^{j'}$ from each chosen $B_{i'}$ such that Item 1 is satisfied which forms a k-obstruction set (this is always possible since each B_l has $k+1$ facts).

Suppose B is of the form E_i^j then if $b = u_i^j$ then we can pick $V(j,l-1) \cup U(j,l-1) \cup \{b_{l-1}^j\}$ that forms an obstruction set for $\{B_j\} \cup \mathbb{E}^j$. If $b = v_i^j$ then we can pick $V(j,l+1) \cup U(j,l+1) \cup \{b_{l+1}^j\}$ that again forms an obstruction set for $\{B_j\} \cup \mathbb{E}^j$. □

Lemma 5.6. *If W is a k-obstruction set, then there are no solutions to the query q_4 within W.*

Proof. Let W be a partial repair of $\mathbb{X} = \{X_1, \dots X_k\}$. By Condition (3), for every $1 \leq j \leq n-1$ there are no solutions within $W[\mathbb{E}^j]$. By construction, it is also not possible to have solutions involving one fact from $W[\mathbb{E}^j]$ and another from $W[\mathbb{E}^{j'}]$ for $j \neq j'$ (*cf.* Figure 1).

So if a solution exists, it has to involve some fact of the form $b_l^j \in W[\mathbb{B}]$. By construction of D_n this solution must involve either u_{l-1}^j or v_{l-2}^j. But by Condition (2) none of them are in W. □

We are now in shape to prove Proposition 5.4, which is a consequence of the following two claims.

Claim 5.7. For every set of k blocks $\mathbb{X} = \{X_1, \dots X_k\}$ of D_{k+2}, there exists a partial repair W of $\mathbb{X}$ such that W is a k-obstruction set.

Claim 5.8. For every set of facts W, if W is a k-obstruction set, then $W \notin \Delta_k^+(\mathsf{q}_4, D_{k+2})$.

Proof of Proposition 5.4. Assuming that the two claims are true, they together imply that for every set of blocks $\{X_1, \dots, X_k\}$ of D_{k+2} there exists a partial repair W for $\{X_1, \dots, X_k\}$ such that $W \notin \Delta_k^+(\mathsf{q}_4, D_{k+2})$.

Suppose $\emptyset \in \Delta_k^+(\mathsf{q}_4, D_{k+2})$ was obtained by using the first derivation rule. Then, there is some block B such that for every fact $a \in B$ we have $\{a\} \in \Delta_k^+(\mathsf{q}_4, D_{k+2})$. Let $\mathbb{X}$ be an arbitrary set of blocks of size k such that $B \in \mathbb{X}$ (such $\mathbb{X}$ can always be picked since D_{k+2} has sufficiently many blocks). By Claim 5.7 there exists a k-obstruction set W which is a partial repair of $\mathbb{X}$ and by Claim 5.8 $W \notin \Delta_k^+(\mathsf{q}_4, D_{k+2})$. This is a contradiction for the fact $a \in B \cap W$. Thus, $D_{k+2} \not\models \mathrm{Cert}_k^+(\mathsf{q}_4)$.

If, on the other hand, $\emptyset \in \Delta_k^+(\mathsf{q}_4, D_{k+2})$ was obtained by using the second derivation rule, then there is a fact a such that for every $a' \in A = \{b \in D_{k+2} : D_{k+2} \models q(ab) \vee q(ba) \vee a = b\}$ we have $\{a'\} \in \Delta_k^+(\mathsf{q}_4, D_{k+2})$. But from Lemma 5.5 there is a k-obstruction set W such that $a \in W$. Moreover, from the two rules to compute $\Delta_k^+(\mathsf{q}_4, D_{k+2})$, it follows that if $S \in \Delta_k^+(\mathsf{q}_4, D_{k+2})$ and $S \subseteq S'$ where $|S'| \leq k$ then $S' \in \Delta_k^+(\mathsf{q}_4, D_{k+2})$. Hence, $W \in \Delta_k^+(\mathsf{q}_4, D_{k+2})$ which contradicts Claim 5.8. □

Hence, we are only left with the proofs of Claims 5.7 and 5.8.

Proof of Claim 5.7. Recall that D_{k+2} has $k+2$ blocks of the form B_i and each B_i has $k+1$ facts. Further, for every $1 \leq j \leq k+1$ we have the sets of blocks of the from $E_1^j \dots E_{k-1}^j$.

First let $\mathbb{X} \cap \mathbb{B} = \{B_{i_1}, \dots B_{i_m}\}$ for some $m \leq k$ and $i_1, i_2 \dots i_m \leq k+2$. We let W to contain $\{b_{i_1}^1, b_{i_2}^2, \dots b_{i_m}^m\}$. Moreover for every $1 \leq j \leq m$ we add to W the partial repair induced by $U(j, i_j) \cup V(j, i_j)$. Formally, we add $(U(j, i_j) \cup V(j, i_j)) \cap \bigcup \mathbb{X}$ to W (*i.e.*, all facts of $U(j, i_j) \cup V(j, i_j)$ that are in a block of $\mathbb{X}$). This ensures that conditions (1) and (2) are satisfied.

Now for all $m < j \leq k+1$ we add to W the partial repair induced by $V(j, 1)$ (*i.e.*, $V(j, 1) \cap \bigcup \mathbb{X}$). Hence, condition (3) is also satisfied and W is a k-obstruction set. □

Proof of Claim 5.8. Let $\Delta_k^+(\mathsf{q}_4, D_{k+2}, i)$ be the set computed by the algorithm of $\mathrm{Cert}_k^+(\mathsf{q}_4)$ at step i of the fixpoint computation.

Suppose the claim is false, and let n be the least index such that $W \in \Delta_k^+(\mathsf{q}_4, D_{k+2}, n)$ for some k-obstruction set W over the blocks $\mathbb{X} = \{X_1, \dots, X_k\}$.

Note that $n = 0$ is not possible since W does not contain any solution (cf. Lemma 5.6); hence $n > 0$. By definition of $\Delta_k^+(\mathsf{q}_4, D_{k+2}, n)$, this implies that there exists a set of facts A such that either

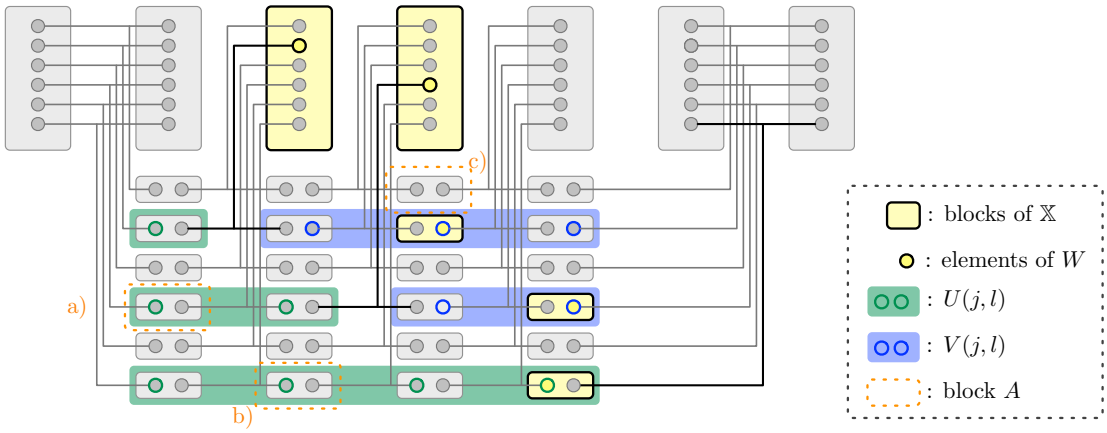


FIGURE 4. Database D_{k+2} in the proof of Claim 5.8, with k-obstruction set W over blocks $\mathbb{X}$. Dotted-line boxes depict the block A in cases a) b) and c) of the proof.

- (first derivation rule) A is a block in D_{k+2}, or
- (second derivation rule) for some fact a we have $D \not\models q(aa)$ and $A = \{a\} \cup \{b \mid D \models q(ab) \cup a(ba)\}$,

and for all $a' \in A$ there is a subset $W' \subseteq W \cup \{a'\}$ of size at most k such that $W' \in \Delta_k^+(\mathtt{q}_4, D_{k+2}, n-1)$ and further W' is not a k-obstruction set, nor a subset thereof.

Let $\mathbb{X} \cap \mathbb{B} = \{B_{i_1}, \dots B_{i_m}\}$ for some $m \leq k$ and $W[\mathbb{B}] = \{b_{i_1}^{j_1}, b_{i_2}^{j_2}, \dots b_{i_m}^{j_m}\}$. So there exists $k+1-m$ many distinct indices j that are not in $\{j_1, \dots, j_m\}$. Let $j_{m+1}, \dots, j_k, j_{k+1}$ be those indices. Since $\mathbb{X}$ is of size k, among those indices there can be at most $k-m$ many indices s such that $\mathbb{X} \cap \mathbb{E}^s \neq \emptyset$. Thus, there exists at least one index s such that $s \notin \{j_1, \dots j_m\}$ and $\mathbb{X} \cap \mathbb{E}^s = \emptyset$.

Now we consider various candidates for A and show that each case leads to a contradiction.

- If $A = B_l$ for some $1 \leq l \leq k+2$ then consider $b_l^s \in B_l$. Observe that every k-set of size k over $W \cup \{b_l^s\}$ forms a k-obstruction set since condition (2) holds vacuously for s. Hence, there must exist a k-obstruction set inside $\Delta_k^+(\mathtt{q}_4, D_{k+2}, n-1)$, which is in contradiction with the minimality of n.
- If, otherwise, $A = E_l^\lambda$ for some $1 \leq \lambda \leq k+1$ and $1 \leq l \leq k-1$, there are three cases to consider depending on where A lies in relation to $\mathbb{X}$ (see Figure 4):
  a) If $\lambda \in \{j_1, \dots j_m\}$, say $\lambda = j_t$, then by condition (2) $W[\mathbb{E}^\lambda] \subseteq U(j_t, i_t) \cup V(j_t, i_t)$. Let w be the fact in E_l^λ picked by $U(j_t, i_t) \cup V(j_t, i_t)$. Then every k-sized subset of $W \cup \{w\}$ is a k-obstruction set (all conditions follow since W is already a k-obstruction set). This is in contradiction with the minimality of n.
  b) If $\lambda \notin \{j_1, \dots j_m\}$ but $\mathbb{X} \cap \mathbb{E}^r \neq \emptyset$ then by condition (3), $W[\mathbb{E}^\lambda] \subseteq U(\lambda, \lambda') \cup V(\lambda, \lambda')$ for some $1 \leq \lambda' \leq k+2$. Let w be the fact in E_l^λ picked by $U(\lambda, \lambda') \cup V(\lambda, \lambda')$. Then, again every k-size subset of $W \cup \{w\}$ is a k-obstruction set (all conditions follow since W is already a k-obstruction set). This is a contradiction.

c) Otherwise, $\lambda \notin \{j_1, \dots j_m\}$ and $\mathbb{X} \cap \mathbb{E}^\lambda = \emptyset$. In this case, pick $v_l^\lambda \in E_l^\lambda$ for some arbitrary l. We have $\{v_l^\lambda\} \subseteq U(\lambda, 1) \cup V(\lambda, 1)$. Hence, it can be verified that every k-sized subset of $W \cup \{v_l^\lambda\}$ is a k-obstruction set, which again is a contradiction.

- Otherwise $A = \{b \in D : D \models q(ab) \vee q(ba) \vee a = b\}$ for some fact $a \in D_{k+2}$ which implies that A is of the form $\{b_l^\lambda, u_{l-1}^\lambda, v_{l-2}^\lambda\}$ or $\{b_1^\lambda, b_2^\lambda, u_1^\lambda\}$ or $\{b_{n-1}^\lambda, b_n^\lambda, v_{n-3}^\lambda\}$. We only prove the case where $A = \{b_l^\lambda, u_{l-1}^\lambda, v_{l-2}^\lambda\}$ and the other two cases are analogous. The argument is similar to the previous case.

a) If $\lambda \in \{j_1, \dots j_m\}$, say $\lambda = j_t$, then by condition (2) $W[\mathbb{E}^\lambda] \subseteq U(j_t, i_t) \cup V(j_t, i_t)$. Let w be the fact in A picked by $U(j_t, i_t) \cup V(j_t, i_t)$. Then every k-sized subset of $W \cup \{w\}$ is a k-obstruction set (all conditions follow since W is already a k-obstruction set). This is in contradiction with the minimality of n.

b) If $\lambda \notin \{j_1, \dots j_m\}$ but $\mathbb{X} \cap \mathbb{E}^r \neq \emptyset$ then by condition (3), $W[\mathbb{E}^\lambda] \subseteq U(\lambda, \lambda') \cup V(\lambda, \lambda')$ for some $1 \leq \lambda' \leq k+2$. Let w be the fact in A picked by $U(\lambda, \lambda') \cup V(\lambda, \lambda')$. Then, again every k-size subset of $W \cup \{w\}$ is a k-obstruction set (all conditions follow since W is already a k-obstruction set). This is a contradiction.

c) Otherwise, $\lambda \notin \{j_1, \dots j_m\}$ and $\mathbb{X} \cap \mathbb{E}^\lambda = \emptyset$. In this case, pick $v_{l-2}^\lambda \in A$. We have $\{v_{l-2}^\lambda\} \subseteq U(\lambda, 1) \cup V(\lambda, 1)$. Hence, it can be verified that every k-sized subset of $W \cup \{v_l^\lambda\}$ is a k-obstruction set, which again is a contradiction. □

As we have seen, certain(q_4) cannot be computed using our fixpoint algorithm. We will next show that certain(q_4) is complete for the *Saturating Bipartite Matching problem* (or *SBM problem* for short). This is the problem of, given bipartite graph $(V_1 \cup V_2, E)$, whether there is an injective function $f : V_1 \to V_2$ such that $(v, f(v)) \in E$ for every $v \in V_1$. The SBM problem[3] is known to be in PTIME [HK73], hence certain(q_4) can be solved in polynomial time, although its precise complexity class is open.

Theorem 5.9. certain(q_4) *is complete for the (complement of the) SBM problem under* LOGSPACE*-reductions. In particular* certain(q_4) *is in* PTIME.

Proof. First we prove that there is a LOGSPACE-reduction from certain(q_4) to the (complement of the) SBM problem. Fix an input database D. We reduce certain(q_4, D) to the SBM problem.

First we check for self-loops in D. If there is a fact $a \in D$ such that $D \models q(aa)$ then we first check if a is a singleton block. If so, then $D \models$ certain(q_4). Otherwise, $D \models$ certain(q_4) iff $D \setminus \{a\} \models$ certain(q_4), so we can consider a smaller database and repeat the argument. Moreover this pre-processing can be performed in LOGSPACE. So assume that there is no fact $a \in D$ such that $D \models \mathsf{q}_4(aa)$.

Now consider the bipartite graph $G = (V_1 \cup V_2, E)$ where V_1 is the set of all blocks of D and V_2 is the set of all maximal cliques in the solution graph G_D. Note that, by property (4) (in page 18), V_2 forms a partition of D, namely the set of all maximal connected components of G_D. Let $(v_1, v_2) \in E$ if the block v_1 contains a fact which is in the clique v_2.

Suppose that there is a *V_1-saturating matching*, that is, an injective function $f : V_1 \to V_2$ such that $(v_1, f(v_1)) \in E$ for every $v_1 \in V_1$. We construct a repair r where for every block B of D, we pick the fact (or one of the facts, if there are more than one) which is in $f(B)$. In this way, no two chosen facts will be in the same clique, and also since there is no solution of the form $\mathsf{q}_4(aa)$ in D, no two chosen facts will form a solution to q_4. Thus, $r \not\models \mathsf{q}_4$.

[3] In turn, the SBM problem is equivalent to the Bipartite Perfect Matching problem under LOGSPACE reductions, which corresponds to the restriction to instances in which $|V_1| = |V_2|$.

Conversely, if q_4 is not certain in D, let r be a repair such that $r \not\models \mathsf{q}_4$. For each block B of D let $r(B)$ be the fact of B belonging to r. Note that, since V_2 is a partition of D, each $r(B)$ belongs to a unique clique in V_2. Define $f : V_1 \to V_2$ such that each block $B \in V_1$ is mapped to the clique in V_2 where $r(B)$ lies. To verify that f is a witness function of a V_1-saturating matching for G, note that for every $B \in V_1$ we have $(B, f(B)) \in E$, as B and $f(B)$ both contain $r(B)$. Moreover, f is injective, otherwise if f maps two distinct blocks to the same clique, this clique must contain at least two facts a, b from r. These two facts are neighbors in G_D, and then $r \models \mathsf{q}_4(ab)$ or $r \models \mathsf{q}_4(ba)$, contradicting the hypothesis $r \not\models \mathsf{q}_4$.

Thus, to check if $D \in \mathsf{certain}(\mathsf{q}_4)$, it is sufficient to check if there is a V_1-saturating matching for G_D.

For the other direction, given a bipartite graph $G = (V_1 \cup V_2, E)$, let $V_1 = \{s_1, \dots s_n\}$ and $V_2 = \{t_1, \dots t_m\}$. We will define a database D_G such that there exists a V_1-saturating matching in G iff q_4 is not certain in D_G.

For all $s_j \in V_1$ let $N(s_j) \subseteq V_2$ denote the neighbours of s_j and similarly for all $t_i \in V_2$ let $N(t_i) \subseteq V_1$ denote the neighbours of t_j.

First note that if there is some $s_j \in V_1$ such that $N(s_j) = \emptyset$, then clearly there cannot be a V_1-saturating matching. Similarly, if there is some $t_i \in V_2$ such that $N(t_i) = \emptyset$, then t_i does not contribute to any matching and hence can be removed from the input. Further, suppose there is some $t_i \in V_2$ such that $|N(t_i)| = 1$, let s_j be the single neighbour of t_i. In this case, in every V_1-saturating matching maps s_j to t_i. So we can remove the vertices s_j and t_i from the input graph and conside a smaller instance. Note that all these checks can be done in LogSpace. Hence we assume that for every $u \in V_1 \cup V_2$, $N(u) \neq \emptyset$ and $|N(s_j)| \geq 1$ for all $s_j \in V_1$ and $|N(t_i)| \geq 2$ for all $t_i \in V_2$.

Now we define the database D_G. Note that this construction is very similar to the construction of D_n that we used to prove Theorem 5.1.

- For every vertex in $s_j \in V_1$ create a block B_j in D_G.
- For every $s_j \in V_1$ and $t_i \in V_2$, if $t_i \in N(s_j)$ then there is a fact denoted by b^i_j in the block B_j. By assumption $N(s_j) \geq 1$ and hence every block B_j is non-empty.
- For every $t_i \in V_2$ if $|N(t_i)| = l$ then let $s_{i_1}, \dots s_{i_l} \in V_1$ be the neighbours of t. By the above construction, for every $j \leq l$, there is a fact of the form $b^i_{i_j}$ in B_{i_j} that corresponds to the vertex t_i.

  Now if $l = 2$ then define $b^i_{i_1}$ and $b^i_{i_2}$ such that they form a solution to q_4. Otherwise, if $l = 3$ then define $b^i_{i_1}, b^i_{i_2}$ and $b^i_{i_3}$ such that they pair-wise form a solution to q_4 (the three facts form a triangle).

  If $l \geq 4$ then create $l - 3$ new blocks denoted by $E^i_1, \dots E^i_{l-3}$ where each E^i_j contains exactly two facts u^i_j and v^i_j. Moreover, in the same way as described in the definition of D_n earlier (*cf.* Figures 1 and 2), define the facts appropriately such that $\{b^i_{i_1}, b^i_{i_2}, u^i_1\}$ and $\{b^i_{i_{l-1}}, b^i_{i_l}, v^i_{l-3}\}$ form triangles and for every $1 \leq j < l-3$ we have a triangle $\{v^i_j, u^i_{j+1}, b^i_{j+2}\}$.

The reader can verify that this is exactly the construction used to define D_n. Again, this construction is in LogSpace. For each such j and l define the analogous $U(i,l) \hat{=} \{u^i_k \mid 1 \leq k \leq l-2\}$ and $V(i,l) \hat{=} \{v^i_k \mid l-1 \leq k \leq l-3\}$.

Now suppose there is a V_1-saturating matching then let us show that q_4 is not certain for D_G. Consider the repair r where for each block B_j we pick b^i_j if s_j is matched with t_i. Further, pick $U(i,l) \cup V(i,l)$ which gives a partial repair over $E^i_1 \dots E^i_l$.

If some $t_i \in V_2$ is not matched with any vertex in V_1 then pick $U(i,1) \cup V(i,1)$ which gives a partial repair over $E^i_1 \dots E^i_l$. It can be verified that the obtained repair does not contain any solution.

Conversely, suppose there is a repair r of D_G that falsifies q_4, and let us show that there is a V_1-saturating matching in G. For any such repair r, note that if b^i_j is picked in block B_j then for all other blocks $B_{j'}$, the fact $b^i_{j'}$ cannot be in r since that would make q_4 true. Also $b^i_j \in B_j$ only if there is an edge between s_j and t_i. Hence, we can define the V_1-saturating matching that maps every $s_j \in V_1$ to $t_i \in V_2$, where b^i_j is the fact in r from the block B_j. □

5.2. The case of q_5.

We now show that $\mathsf{certain}(\mathsf{q}_5)$ cannot be computed by $\mathrm{Cert}_k(\mathsf{q}_5)$. This is shown by reduction to the case of q_4 based on the following construction:

Proposition 5.10. *For every database D over the signature of q_4 we can construct a database D' over the signature of q_5 such that:*
(1) *if $D \models \mathsf{certain}(\mathsf{q}_4)$ then $D' \models \mathsf{certain}(\mathsf{q}_5)$;*
(2) *for every $k \geq 2$, if $D' \models \mathrm{Cert}_k(\mathsf{q}_5)$ then $D \models \mathrm{Cert}^+_k(\mathsf{q}_4)$.*

Proof. Let D be a database over the signature of q_4. Consider the solution graph G_D of D. Recall that by property (4), every connected component in G_D is always a clique of size at most 3. Thus, every fact of D can be part of exactly one maximal clique.

Let $B_1, B_2 \dots B_m$ be the set of all blocks of D and $C_1, C_2 \dots C_l$ be the set of cliques in G_D. Let $e_1, e_2 \dots e_m, f_1, f_2 \dots f_l$ be fresh and pairwise distinct domain elements. Define D' as follows:

- For every fact u occurring in a block B_i of D and in the clique C_n add the fact $u_{R_1} = R_1(\underline{e_i}\, f_n)$ to D'. Notice that there could be two distinct facts occurring in the same block of D and in the same clique. They are then both associated to the same fact of D'.
- For every two facts u, v from a clique C_n such that $D \models \mathsf{q}_4(uv)$ and u, v are in two distinct blocks B_i and B_j, create two new facts in D' as $u^v_{S_1} = S_1(\underline{f_n g}\ e_i)$ and $v^u_{S_1} = S_1(\underline{f_n g}\ e_j)$, where g is a fresh element depending only on u and v.
- For every fact u from a clique C_n and a block B_i such that $D \models \mathsf{q}_4(uu)$ add a fact in D' as $u^u_{S_1} = S_1(\underline{f_n g}\ e_i)$, where g is a fresh element depending only on u.

See Figure 5 for an example.

Notice that for any two facts $u, v \in D$ if $D \models \mathsf{q}_4(uv)$ and u, v are in two distinct blocks, then $u^v_{S_1} \sim v^u_{S_1}$, $D' \models \mathsf{q}_5(u_{R_1} u^v_{S_1}) \wedge \mathsf{q}_5(v_{R_1} v^u_{S_1})$, and the block containing $u^v_{S_1}, v^u_{S_1}$ in D' does not contain any other fact. Moreover if for some fact $u \in D$, $D \models \mathsf{q}_4(uu)$ then $D' \models \mathsf{q}_5(u_{R_1} u^u_{S_1})$ and the block containing $u^u_{S_1}$ in D' does not contain any other fact. Finally, the only solutions in D' for q_5 are of the form $\mathsf{q}_5(u_{R_1} u^v_{S_1})$ for some facts $u, v \in D$ such that $D \models \mathsf{q}_4(uv)$ and $u \not\sim v$ or $u = v$. Now we prove both claims of the statement.

(1) Suppose $D \models \mathsf{certain}(\mathsf{q}_4)$. To verify that $D' \models \mathsf{certain}(\mathsf{q}_5)$, pick any repair r' of D'. By construction, every R_1-fact a of r' is of the form u_{R_1} for some fact u in D. Recall that there could be two such facts u. We arbitrarily select one of them that we denote by $f_R(a)$, and we consider the set $r = \{u \mid \exists a \in r'\ u = f_R(a)\}$.

First, we prove that r is a repair of D, that is: (i) it contains no two distinct facts $u \sim v$, and (ii) it contains at least one fact for each block B_i.

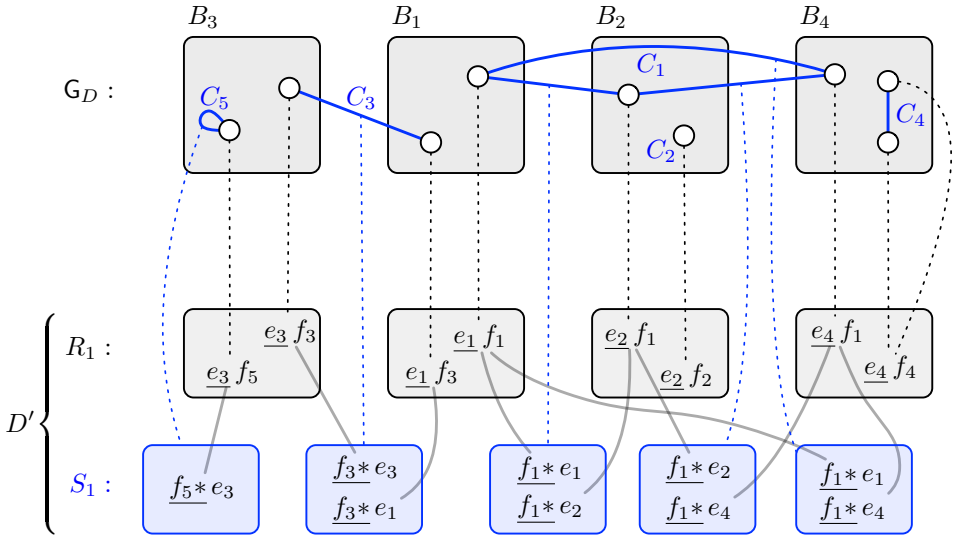


FIGURE 5. Example of construction of D' from D in the proof of Proposition 5.10. The symbols "$*$" stand for the necessary constants "g" used in order to obtain the depicted blocks. D has 4 blocks and 5 cliques. Observe that neither the edge-less singleton clique C_2 nor the clique C_4 —which is internal to the block B_4— intervene in the relation S_1. Light gray edges depict the pairs of facts from D' which form a solution to q_5.

(i) By means of contradiction, suppose r contains two facts $u, v \in r$ such that $u \sim v$. By definition of r, $u = f_R(a)$ and $v = f_R(b)$ for some facts a, b of r'. By construction of D', it follows from $u \sim v$ that $a \sim b$ and therefore $a = b$ since we assumed r to be a repair. Hence, $u = v$.

(ii) Further, for every block B_i in D there is a R_1-block in D' whose facts have primary key e_i and one of those facts is selected in r', which gives rise (via f_R) to a fact of B_i.

Since $D \models \mathsf{certain}(\mathsf{q}_4)$ by hypothesis, there are $u, v \in r$ such that $D \models \mathsf{q}_4(uv)$, let us show that this implies $r' \models \mathsf{q}_5$:

- If $u = v$ then, by definition of r, we have $u_{R_1} \in r'$. Since $u^u_{S_1}$ belongs to a singleton S_1-block of D', it is in every repair of D', and thus $r' \models \mathsf{q}_5(u_{R_1} u^u_{S_1})$ as desired.
- If $u \neq v$ then, by definition of r, we have $u_{R_1}, v_{R_1} \in r'$, and $\{u^v_{S_1}, v^u_{S_1}\}$ form a S_1-block of D'. Hence, r' contains one of $\{u^v_{S_1}, v^u_{S_1}\}$. If $u^v_{S_1} \in r'$ we have $r' \models \mathsf{q}_5(u_{R_1} u^v_{S_1})$; otherwise, if $v^u_{S_1} \in r'$, we have $r' \models \mathsf{q}_5(v_{R_1} v^u_{S_1})$.

This concludes the proof for item (1).

(2) Fix some $k \geq 2$. Let $\Delta^+_k(\mathsf{q}_4, D, i)$ and $\Delta_k(\mathsf{q}_5, D', i)$ be the sets computed at step i of $\mathrm{Cert}^+_k(\mathsf{q}_4)$ and $\mathrm{Cert}_k(\mathsf{q}_5)$ respectively. First we prove the following claim.

Claim 5.11. Let u, v, w be facts of D in the same clique of the solution graph G_D. Possibly $v \sim w$ or $v = w$ but we assume $u \not\sim v$ and $u \not\sim w$. In particular $D \models \mathsf{q}_4(uv) \vee \mathsf{q}_4(vu)$. Then for every k-set S over D' if $S \cup \{u^v_{S_1}\} \in \Delta_k(\mathsf{q}_5, D', i)$ then $S \cup \{v_{R_1}\} \in \Delta_k(\mathsf{q}_5, D', i+1)$ and $S \cup \{w_{R_1}\} \in \Delta_k(\mathsf{q}_5, D', i+1)$.

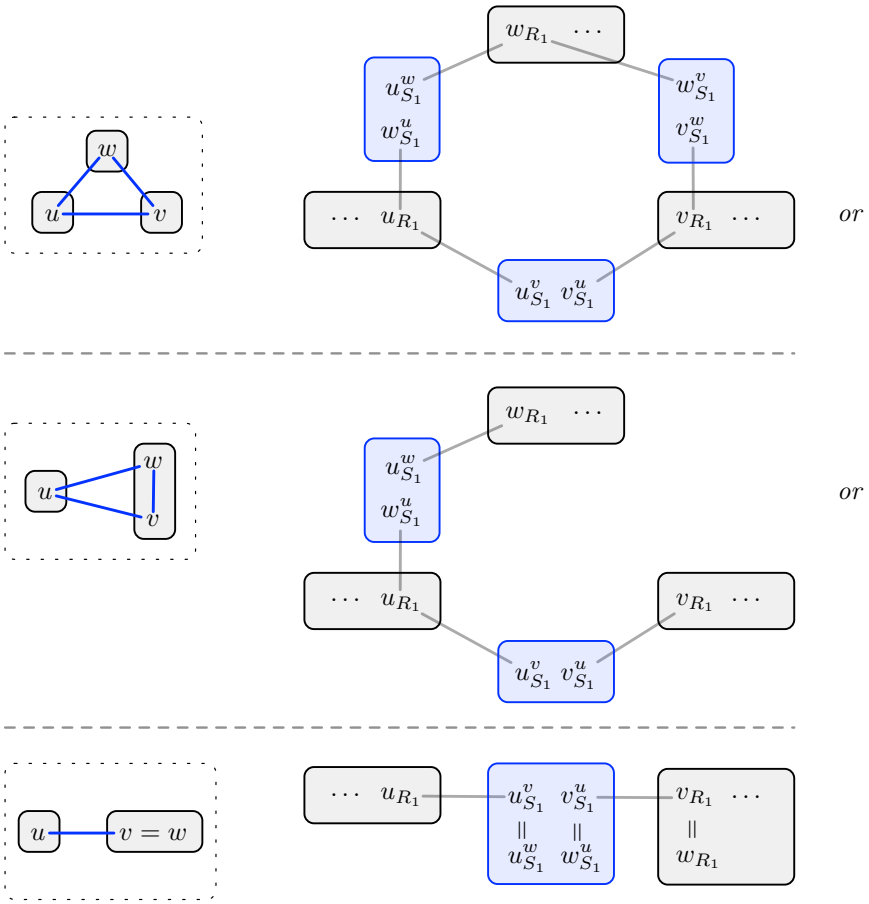


FIGURE 6. Schema for proof of Claim 5.11.

Note that in the case where $v_{R_1} \in S$ the claim implies $S \in \Delta_k(\mathsf{q}_5, D', i+1)$. Similarly for w_{R_1}.

Proof of Claim 5.11. We assume that $D \models \mathsf{q}_4(uv)$, since the case where $D \models \mathsf{q}_4(vu)$ is symmetric. To facilitate the understanding of the following proof, refer to Figure 6.

Assume $S \cup \{u^v_{S_1}\} \in \Delta_k(\mathsf{q}_5, D', i)$. If $S \in \Delta_k(\mathsf{q}_5, D', i)$ then the conclusion is immediate since $S \cup \{u\} \in \Delta_k(\mathsf{q}_5, D', i)$ for every fact u such that $S \cup \{u\}$ is a k-set. So, let us assume $S \notin \Delta_k(\mathsf{q}_5, D', i)$.

- The base case $i = 0$ implies $S \cup \{u^v_{S_1}\} \in \Delta_k(\mathsf{q}_5, D', 0)$ and $S \notin \Delta_k(\mathsf{q}_5, D', 0)$. This can only happen if $u_{R_1} \in S$, since $\{u^v_{S_1}, u_{R_1}\}$ is the only q_5-solution of D' involving $u^v_{S_1}$. Let B be the block $B = \{u^v_{S_1}, v^u_{S_1}\}$. Since $\{u_{R_1}, u^v_{S_1}\}$ and $\{v^u_{S_1}, v_{R_1}\}$ are q_5-solutions (*cf.* Figure 6), a simple analysis shows that the block B is a witness for $S \cup \{v_{R_1}\} \in \Delta_k(\mathsf{q}_5, D', 1)$. Similarly the block $B' = \{u^w_{S_1}, w^u_{S_1}\}$ witnesses the fact that $S \cup \{w_{R_1}\} \in \Delta_k(\mathsf{q}_5, D', 1)$.
- For the induction step, suppose $S \cup \{u^v_{S_1}\} \in \Delta_k(\mathsf{q}_5, D', i)$ and $i > 0$. As $i > 0$, let B be the block of D' used as the witness. By definition, for all fact $b \in B$, there is a subset S_b of S such that $S_b \cup \{u^v_{S_1}, b\} \in \Delta_k(\mathsf{q}_5, D', i-1)$. By induction this implies that for all $b \in B$,

$S_b \cup \{v_{R_1}, b\} \in \Delta_k(\mathsf{q}_5, D', i)$. Hence B witnesses the fact that $S \cup \{v_{R_1}\} \in \Delta_k(\mathsf{q}_5, D', i+1)$, as desired. The proof that $S \cup \{w_{R_1}\} \in \Delta_k(\mathsf{q}_5, D', i+1)$ is similar.

□

With Claim 5.11 in place, assume that $D' \models \mathrm{Cert}_k(\mathsf{q}_5)$ and let us show $D \models \mathrm{Cert}_k^+(\mathsf{q}_4)$. Let S be a k-set of D'. Let $\tilde{S}$ be any k-set of D containing every fact u such that either $u_{R_1} \in S$ or $v^u_{S_1} \in S$ for some fact v.

We will prove, by induction on i, that if $S \in \Delta_k(\mathsf{q}_5, D', i)$ then $\tilde{S} \in \Delta_k^+(\mathsf{q}_4, d, i+1)$ (hence, in particular, $D' \models \mathrm{Cert}_k(\mathsf{q}_5)$ implies $D \models \mathrm{Cert}_k^+(\mathsf{q}_4)$, concluding the proof of (2)).

- For the base case $i = 0$, if $S \in \Delta_k(\mathsf{q}_5, D', 0)$ then it contains a solution to q_5 of the form $u_{R_1} u^v_{S_1}$ for some facts u, v of D. By construction $D \models \mathsf{q}_4(uv)$ and $u, v \in \tilde{S}$. Hence $\tilde{S} \in \Delta_k^+(\mathsf{q}_4, d, 0)$.
- For the induction step, assume $S \in \Delta_k(\mathsf{q}_5, D', i)$ as witnessed by a block B' of D'.
  - If B' is an R_1-block then the block $\tilde{B} = \{u \mid u_{R_1} \in B'\}$ is a witness for $\tilde{S} \in \Delta_k^+(\mathsf{q}_4, D, i)$.
  - If B' is an S_1-block of the form $\{u^v_{S_1}, v^u_{S_1}\}$. Set w to the third fact in the clique of u, v in the solution graph G_D (set $w = v$ if the clique has size two). From Claim 5.11 we get that each of $S \cup \{u_{R_1}\}$, $S \cup \{v_{R_1}\}$, $S \cup \{w_{R_1}\}$ contains a k-set in $\Delta_k(\mathsf{q}_5, D', i-1)$. Hence, by induction each of $\tilde{S} \cup \{u\}$, $\tilde{S} \cup \{v\}$, $\tilde{S} \cup \{w\}$ contains a k-set in $\Delta_k^+(\mathsf{q}_4, D, i)$. Hence, by Δ_k^+'s second derivation rule on the fact u, $\tilde{S} \in \Delta_k^+(\mathsf{q}_4, D, i+1)$.
  - It remains to consider the case where B' is an S_1-block of the form $\{u^u_{S_1}\}$. But then u is a self-loop and is contained in $\tilde{S}$. By definition we then have $\tilde{S} \in \Delta_k^+(\mathsf{q}_4, D, 0)$. □

Theorem 5.12. $\mathsf{certain}(\mathsf{q}_5)$ *cannot be computed by* $\mathrm{Cert}_k(\mathsf{q}_5)$*, for any choice of* k*.*

Proof. By means of contradiction, assume that $\mathsf{certain}(\mathsf{q}_5)$ is equivalent to $\mathrm{Cert}_k(\mathsf{q}_5)$ for some $k \geq 2$. We show that this implies that $\mathsf{certain}(\mathsf{q}_4)$ is equivalent to $\mathrm{Cert}_k^+(\mathsf{q}_4)$, contradicting Theorem 5.1.

It is enough to show that if a database D is such that $D \models \mathsf{certain}(\mathsf{q}_4)$ then $D \models \mathrm{Cert}_k^+(\mathsf{q}_4)$. To this end, assume D is a database such that $D \models \mathsf{certain}(\mathsf{q}_4)$. Let D' be the database constructed from D given by Proposition 5.10. As $D \models \mathsf{certain}(\mathsf{q}_4)$, we get from Proposition 5.10-(1) that $D' \models \mathsf{certain}(\mathsf{q}_5)$. From our hypothesis it follows that $D' \models \mathrm{Cert}_k(\mathsf{q}_5)$. By Proposition 5.10-(2) this implies that $D \models \mathrm{Cert}_k^+(\mathsf{q}_4)$ as desired. □

5.3. The case of all self-join-free queries not satisfying PCond: coNP-hardness.

We describe in this section techniques from [KW17], together with their immediate consequence: $\mathsf{certain}(q)$ is coNP-hard as soon as $q \models \neg\mathrm{PCond}$. We build on the dichotomy result of [KW17] based on their notion of an "attack graph". First we recall this notion using our notation.

Let q be a query, let Γ be a set of primary key constraints. Given an atom A of q let

$$A^+ \hat{=} \{B \text{ atom of } q \mid \text{there exists a } \Gamma\text{-derivation } X\ B_1 \dots B_n \text{ where } X = \mathit{key}(A),\ B_n = B, \text{ and for all } i,\ B_i \neq A\}$$

Let $\mathit{vars}(A^+) = \bigcup_{B \in A^+} \mathit{vars}(B)$. Given two atoms A and B of q we say that A *attacks* B if there exists a sequence $F_0, F_1, \dots, F_n$ of atoms of q and $x_1, x_2, \dots, x_n$ of variables not in $\mathit{vars}(A^+)$ such that $A = F_0, B = F_n$ and for all $i > 0$, x_i is a variable occurring both in F_{i-1} and F_i. The attack from A to B is said to be *weak* if B is Γ-determined by A. The

attack graph of q and Γ is the graph whose vertices are the atoms of q and whose edges are the attacks. A cycle in this graph is *weak* if all the attacks involved are weak, otherwise it is a *strong cycle*.

The dichotomy result of [KW17] can be stated as:

Theorem 5.13. [KW17, Theorem 3.2] *Let q be a self-join-free query and Γ a set of primary key constraints. If every cycle in the attack graph of q and Γ is weak, then $\mathsf{certain}(q)$ can be computed in polynomial time; otherwise $\mathsf{certain}(q)$ is* coNP*-complete.*

We prove that if the attack graph of q and Γ contains only weak cycles, then PCond holds. It then follows from Theorem 5.13 then whenever $q \models \neg\text{PCond}$ then $\mathsf{certain}(q)$ is coNP-hard as desired:

Theorem 5.14. *Assume q is a self-join-free query and Γ a set of primary key constraints. If $q \models \neg\text{PCond}$, then $\mathsf{certain}(q)$ is* coNP*-hard.*

In view of Theorem 5.13, the proof of Theorem 5.14 is an immediate consequence of the following lemma.

Lemma 5.15. *Assume q is a self-join-free query and Γ a set of primary key constraints. If the attack graph of q and Γ contains only weak cycles then $q \models \text{PCond}$.*

Proof. Let $\mathcal{X}$ be the set of all the strongly connected components of the attack graph of q and Γ. We define the *core graph* of q and Γ as the directed graph whose vertices are the elements of $\mathcal{X}$ and there is an edge from S to S' if S contains an atom A attacking an atom B of S'. Note that, by definition, a core graph is always acyclic. Let τ be any topological ordering of this graph, that is, $\tau = S_1, \ldots, S_n$ is an ordering on $\mathcal{X}$, and for every i, j, if there is an edge from S_i to S_j in the core graph, then $i < j$. Note that since the attack graph contains only weak cycles, any two facts belonging to the same strongly connected component are mutually Γ-determined. Hence, every $S \in \mathcal{X}$ is a stable set. In particular τ is a Γ-sequence. We claim that $q \models \text{PCond}_\tau$.

To prove $q \models \text{PCond}_\tau$, we need to show that $q \models \text{PCond}_\tau(i)$ for every $i < n$. So, fix some i and consider an arbitrary atom A of S_{i+1}. Let D be some database with repair r such that $r \models q(\bar{\alpha}a\bar{\beta})$, where a matches A and $\bar{\alpha}$ matches $S_{\leq i}$. Let μ be the valuation of the variables of q witnessing the solution $\bar{\alpha}a\bar{\beta}$. By abuse of notation, for each atom B of q, we write $\mu(B)$ to denote the fact of the database witnessing the solution for the relation symbol of B. In particular, $a = \mu(A)$.

Let $a' \sim a$, $r' = r[a \to a']$, and assume that $r' \models q(\bar{\alpha}'a'\bar{\beta}')$ as witnessed by the valuation μ'. We need to show that $r' \models q(\bar{\alpha}a'\bar{\delta})$ for some $\bar{\delta}$. Notice that the hypotheses of Lemma 4.1 are satisfied by r, r', μ, μ' and there is a Γ-derivation from $X = key(A)$ to every atom in A^+. Hence, from Lemma 4.1 it follows that

$$\text{for any variable } x \in vars(A^+) \text{ we have } \mu(x) = \mu'(x). \qquad (\dagger)$$

To show $r' \models q(\bar{\alpha}a'\bar{\delta})$, we define a satisfying valuation ν as follows: If x is a variable occurring in an atom $B \neq A$ that is not attacked by A, then set $\nu(x) = \mu(x)$; otherwise, set $\nu(x) = \mu'(x)$. We show that ν witnesses the solution $\bar{\alpha}a'\bar{\delta}$ in r'.

For this it suffices to show that for all atom B of q, $\nu(B)$ is a fact of r'. If B is different from A and not attacked by A then this is clear as $\nu(B) = \mu(B)$ and $\mu(B)$ belongs to both r and r'. If B is A or attacked by A then we show that $\nu(B) = \mu'(B)$. To see this consider a variable x occurring in B such that $\nu(x) = \mu(x)$. By definition of ν, this is because x

also belongs to some atom C that is not attacked by A. Hence, by definition of attack, this implies that $x \in \mathit{vars}(A^+)$. By (†) this implies that $\mu(x) = \mu'(x)$ as desired. □

5.4. **The case of all self-join-free queries not satisfying PCond: Cert_k fails.**

We finally reduce the case of q_5 to any arbitrary query not verifying PCond. Let q be a self-join-free query such that $q \models \neg\mathrm{PCond}$. We show that for all k, if $\mathsf{certain}(q)$ is equivalent to $\mathrm{Cert}_k(q)$ then $\mathsf{certain}(\mathsf{q}_5)$ is equivalent to $\mathrm{Cert}_k(\mathsf{q}_5)$, a contradiction with Theorem 5.12. The following is the analog of Proposition 5.10.

Proposition 5.16. *For every database D over the signature of q_5 we can construct a database D' over the signature of q such that:*

(1) *if $D \models \mathsf{certain}(\mathsf{q}_5)$ then $D' \models \mathsf{certain}(q)$;*
(2) *For every $k \geq 2$, if $D' \models \mathrm{Cert}_k(q)$ then $D \models \mathrm{Cert}_k(\mathsf{q}_5)$.*

Proof. The construction of D' from D is actually taken from [KW17, proof of Theorem 6.1]. Since $q \models \neg\mathrm{PCond}$, it follows from Theorem 5.14 that the attack graph of q contains a strong cycle. Then the following result follows from [KW17, proof of Theorem 6.1].

Claim 5.17. [KW17, proof of Theorem 6.1] Let q be a self-join-free query such that the attack graph of q contains a strong cycle. Then q contains two atoms A_1 and A_2 such that for every database D over the signature of q_5 there is a database D' over the signature of q and functions f and g such that:

(i) If $D \models \mathsf{certain}(\mathsf{q}_5)$, then $D' \models \mathsf{certain}(q)$.
(ii) f is a bijection from the R_1-facts of D to the A_1-facts of D' such that $u \sim v$ iff $f(u) \sim f(v)$.
(iii) g is a function that maps a pair of the form (C, u) where C is an atom of q such that $C \neq A_1$ and u is an S_1-fact of D to a C-fact of D' such that:
  (a) $\{u \mapsto v \mid g(A_2, u) = v\}$ is a bijection from S_1-facts to A_2-facts such that $u \sim v$ iff $g(A_2, u) \sim g(A_2, v)$;
  (b) for every atom $C \notin \{A_1, A_2\}$ and S_1-fact u in D the block of $g(C, u)$ contains only one fact in D'.
(iv) If $f(u)$ and $g(A_2, v)$ are part of a solution to q in D' then $D \models \mathsf{q}_5(uv)$.

Given a database D over the signature of q_5, let D' be the database constructed in Claim 5.17. Note that Claim 5.17-(i) already proves our first item (1). It remains to show that if $D' \models \mathrm{Cert}_k(q)$ then $D \models \mathrm{Cert}_k(\mathsf{q}_5)$.

For any k-set S, let $\hat{S} = \{u \mid f(u) \in S\} \cup \{v \mid g(A_2, v) \in S\}$. We show that if $S \in \Delta_k(q, D', i)$ then $\hat{S} \in \Delta_k(\mathsf{q}_5, D, i)$, by induction on i.

For $i = 0$, S contains necessarily a solution to q and therefore an A_1-fact u' and an A_2-fact v'. Let u and v be such that $v' = g(A_2, v)$ and $u' = f(u)$. By construction of $\hat{S}$, both u and v are in $\hat{S}$. By Claim 5.17-(iv) this implies that $D \models \mathsf{q}_5(uv)$ and therefore $\hat{S} \in \Delta_k(\mathsf{q}_5, D, 0)$.

For the induction step, let B' be the block witnessing the membership of S into $\Delta_k(q, D', i)$.

- Assume first that B' is a A_1-block. Let $B = \{u \mid f(u) \in B'\}$. By Claim 5.17-(ii) B is a block of D. Moreover, it witnesses the membership of $\hat{S}$ in $\Delta_k(\mathsf{q}_5, D, i)$. Indeed, consider an element $b \in B$. Let $b' = f(b)$. By hypothesis, $S \cup \{b'\}$ contains a k-set T

in $\Delta_k(q, D', i-1)$. By induction $\hat{T} \in \Delta_k(\mathsf{q}_5, D, i-1)$ and $\hat{T}$ is a subset of $\hat{S} \cup \{b\}$. The result follows.
- Assume now that B' is a A_2-block. We conclude as above using $B = \{v \mid g(A_2, v) \in B'\}$ and Claim 5.17-(iii)-(a).
- Finally, if B' is a C-block for some atom $C \notin \{A_1, A_2\}$, then B' contains only one element by Claim 5.17-(iii)-(b). Therefore, S was actually in $\Delta_k(q, D', i-1)$ and we can conclude by induction hypothesis. □

Theorem 5.18. *Let q be a self-join-free query such that $q \models \neg$PCond. Then $\mathsf{certain}(q)$ cannot be computed by $\mathrm{Cert}_k(q)$, for any choice of k.*

The proof of Theorem 5.18 is identical to the proof of Theorem 5.12 using Proposition 5.16 instead of Proposition 5.10.

6. Path queries

The dichotomy conjecture has also been shown to hold for path queries [KOW21]. In this section we show that the Cert_k algorithm works for PTime solvable path queries and if Cert_k does not compute $\mathsf{certain}(q)$ for $k = |q|$ then the problem is coNP-hard.

For this section, assume that the relational signature σ contains only symbols of arity two and that the set Γ of constraints assigns to each symbol R of σ its first component as the primary key. Recall that a path query is a Boolean conjunctive query of the form $R_1(\underline{x}_1\, x_2) \wedge R_2(\underline{x}_2\, x_3) \wedge R_3(\underline{x}_3\, x_4) \wedge \cdots \wedge R_n(\underline{x}_n x_{n+1})$ that may contain self-join *i.e.* $R_i = R_j$ for some $i \neq j$. Note that a path query can be described by a word over the alphabet of relation names of σ (*e.g.*, the word describing q as $R_1 \cdots R_n$). For simplicity, we will henceforth blur the distinction between path queries and words over σ.

Following [KOW21] we define the language $\mathcal{L}^{\curvearrowright}(q)$ as the regular language defined by the following finite state automaton $\mathcal{A}^q$ with ε-transitions[4] (we use $s, t, \ldots$ to denote words over σ). The set of states of $\mathcal{A}^q$ is the set of all prefixes of q, including the empty prefix ε, which is the initial state. There is only one accepting state, which is q. There is a transition reading R from state s to the state sR. Moreover, there is an ε-transition in $\mathcal{A}^q$ from any state sR to any state tR such that tR is a prefix of s.

We say that the query q satisfies FactorCond and write $q \models$ FactorCond if q is a factor of all the words in the language $\mathcal{L}^{\curvearrowright}(q)$.

The dichotomy result of [KOW21] can be formulated as follows[5]:

Theorem 6.1. [KOW21, Theorem 3.2] *Let q be a path query. If $q \models$ FactorCond, then $\mathsf{certain}(q)$ can be evaluated in* PTime*; otherwise, $\mathsf{certain}(q)$ is* coNP*-complete.*

Thus, $\mathsf{certain}(\mathsf{q}_2')$ and $\mathsf{certain}(\mathsf{q}_3')$ for the queries described in Example 2.2 are in PTime and coNP-complete respectively (refer to [KOW21] for detailed explanation). As in the self-join-free case we will show that, for path queries, there is some k such that $\mathrm{Cert}_k(q)$ captures $\mathsf{certain}(q)$ iff $q \models$ FactorCond. In view of Theorem 6.1 this implies that when $\mathrm{Cert}_k(q)$ fails to capture $\mathsf{certain}(q)$, then $\mathsf{certain}(q)$ is coNP-complete.

[4] An ε-transition in an automata makes a transition from one state to the other without reading any symbol.

[5] [KOW21] provides a much finer 'tetrachotomy' between FO, NL-complete, PTime-complete and coNP-complete. In this section we restrict our attention to the dichotomy between PTime and coNP-complete.

6.1. **Tractable path queries.** In this section we show the first part of our result: if the path query q satisfies FactorCond, then $\mathsf{certain}(q)$ can be computed via Cert_k.

Theorem 6.2. *Let q be a path query of length k. If $q \models$ FactorCond, then $\mathsf{certain}(q) = \mathrm{Cert}_k(q)$.*

The rest of this section is devoted to the proof of Theorem 6.2. We will make use of the following fixpoint computation introduced by [KOW21, Fig. 5]. For a fixed path query q and database D, let $N(q, D)$ be the set of pairs of the form $\langle c, s\rangle$, where $c \in \mathit{adom}(D)$ and s is a prefix of q, computed via the following fixpoint algorithm.

Initialization Step: $N(q, D) \leftarrow \{\langle c, q\rangle \mid c \in \mathit{adom}(D)\}$

Iterative Step: If s is a prefix of q, add $\langle c, s\rangle$ to $N(q, D)$ if one of the following holds:

(1) sR is a prefix of q and there is a fact $R(\underline{c}\ a)$ of D such that for every fact $R(\underline{c}\ b)$ of D we have $\langle b, sR\rangle \in N(q, D)$;
(2) There is an ε-transition from s to t in $\mathcal{A}^q$ and there is a fact $R(\underline{c}\ a)$ of D such that for every fact $R(\underline{c}\ b)$ of D we have $\langle b, tR\rangle \in N(q, D)$.

Let $N(q)$ be the set of all databases D such that there exists $c \in \mathit{adom}(D)$ with $\langle c, \varepsilon\rangle \in N(q, D)$.

Lemma 6.3. [KOW21, (proof of) Lemma 6.4] *For every path query q, if $q \models$ FactorCond, then $\mathsf{certain}(q) = N(q)$.*

In view of Lemma 6.3, Theorem 6.2 is a direct consequence of the following proposition.

Proposition 6.4. *For every path query q of length k such that $q \models$ FactorCond, we have $N(q) = \mathrm{Cert}_k(q)$.*

Note that $\mathrm{Cert}_k(q) \subseteq N(q)$ follows from $\mathrm{Cert}_k(q) \subseteq \mathsf{certain}(q)$ (Proposition 3.3) combined with $\mathsf{certain}(q) = N(q)$ (Lemma 6.3). So we are left with proving $N(q) \subseteq \mathrm{Cert}_k(q)$. Let $D \in N(q)$. We will prove that $D \in \mathrm{Cert}_k(q)$.

For all $l \geq 0$ let s_l be the prefix of q of length l (*i.e.*, $s_0 = \varepsilon$ and $s_k = q$). For every database D and fact $u = R(\underline{a}\ b)$ in D, let us define $\mathit{trace}(u) \mathrel{\hat{=}} R$, $\mathit{key}(u) = a$, and $\mathit{last}(u) = b$. For a sequence of (possibly repeating) facts $\Pi = u_1, \ldots, u_l$ of a database D, we define $\mathit{trace}(\Pi) \mathrel{\hat{=}} \mathit{trace}(u_1) \cdots \mathit{trace}(u_l) \in \sigma^*$ and, if Π is not the empty sequence, we define $\mathit{last}(\Pi) \mathrel{\hat{=}} \mathit{last}(u_l)$. Also we let $S_\Pi = \{u_1, \ldots, u_l\}$ be the *set* of facts in the sequence. Further, Π is called a *valid path* if the following conditions hold:

(i) S_Π is a partial repair of D,
(ii) for all $i < l$ we have $\mathit{last}(u_i) = \mathit{key}(u_{i+1})$, and
(iii) $\mathit{trace}(\Pi)$ is a prefix of q.

In particular, for any valid path Π of length l, we have $\mathit{trace}(\Pi) = s_l$. Moreover, for any prefix s_L of q we write $\mathit{trace}(\Pi) \approx s_L$ if there exists a run of the automaton $\mathcal{A}^q$ on $\mathit{trace}(\Pi)$ ending in state s_L.

For any $D \in N(q)$, let $N(q, D, i)$ and $\Delta_k(q, D, i)$ be the fixpoint computations of $N(q, D)$ and $\Delta_k(q, D)$ at step i respectively. To prove that $D \in \mathrm{Cert}_k(q)$, we will use the following claim:

Claim 6.5. For all $i \geq 0$, For $c \in \mathit{adom}(D)$ and for all non-empty prefix s_l of q if $\langle c, s_l\rangle \in N(q, D, i)$ then for all non-empty valid path Π where $\mathit{last}(\Pi) = c$ and $\mathit{trace}(\Pi) \approx s_l$, we have $S_\Pi \in \Delta_k(q, D, i)$.

Let us show that the claim implies $D \in \mathrm{Cert}_k(q)$. As $D \in N(q)$, there exists $c \in \mathit{adom}(D)$ such that $\langle c, \varepsilon\rangle \in N(q, D, m)$ for some step m. But note that $\langle c, \varepsilon\rangle \in N(q, D, m)$ can only be produced by application of Rule 1 in the *Iteration step* (Rule 2 is not possible since ε-transitions do not start at the state ε). This implies that if R is the first relation occurring in q then there exists a fact of the form $R(\underline{c}\ a)$ and for all facts of the form $R(\underline{c}\ b)$ in D we have $\langle b, R\rangle \in N(q, D, m-1)$. For each such b we can apply the claim with the valid path $\Pi = R(\underline{c}\ b)$, obtaining $R(\underline{c}\ b) \in \Delta_k(q, D, m-1)$ for every $R(\underline{c}\ b)$. Hence, $\emptyset \in \Delta_k(q, D, m)$ which implies $D \in \mathrm{Cert}_k(q)$.

The proof of the claim will make use of the following consequence of $q \models \mathrm{FactorCond}$.

Lemma 6.6. *Let q be a path query such that $q \models \mathrm{FactorCond}$. Then for any prefix of q of the form s_1PRs_2PR' where $s_1, s_2 \in \sigma^*$ and $R \neq R'$, we have that s_1P is a suffix of s_1PRs_2P.*

Proof. Assume $q = s_1PRs_2PR't$ and consider the word $w = s_1PRs_2PRs_2PR't$. A simple observation shows that $w \in \mathcal{L}^{\hookrightarrow}(q)$. Hence, by hypothesis, w contains q as factor. Let $w(i)$ denote the symbol of σ occurring at the i-th position of w, for any $1 \leq i \leq |w|$. Note that by definition of w, for all positions l such that $|s_1| < l \leq |s_1| + |s_2| + 3$ we have:

$$w(l) = w(l + |s_2| + 2) \qquad (\star)$$

Let $w(i), w(i+1), \ldots, w(i+n-1)$ be the factor of w that matches q, and observe that $1 \leq i \leq |w| - |q| + 1 = |s_2| + 3$. If q is a suffix of w, it follows that s_1P is a suffix of s_1PRs_2P and we are done. If q is not a suffix of w, then $i < |s_2| + 3$. Observe that $w((i-1) + |s_1PR|) = R$ and $w((i-1) + |s_1PRs_2PR'|) = R'$. Hence, setting $l = (i-1) + |s_1PR| \leq |s_1| + |s_2| + 3$, we have $w(l) = R$ and $w(l + |s_2| + 2) = R'$. But then by $(\star)$ we would obtain $R = R'$, which is in contradiction with our hypothesis. □

Proof of Claim 6.5. The proof is by induction on i. For the base case $i = 0$, we have $N(q, D, 0) = \{\langle c, q\rangle : c \in \mathit{adom}(D)\}$. Note that $\mathcal{A}^q$ has no ε-transitions from a prefix of q to q. Hence, for any valid path $\Pi = u_1, \ldots, u_l$ such that $\mathit{trace}(\Pi) \approx q$ we must have $k = l$ and $\mathit{trace}(\Pi) = q$. In this case $(u_1, \ldots, u_l)$ forms a solution to q and hence $S_\Pi \in \Delta_k(q, D, 0)$.

For the induction step, let $\Pi = u_1, \ldots, u_l$ be any valid path such that $\mathit{trace}(\Pi) \approx s_L$ and $\mathit{last}(u_l) = c$. Assuming $\langle c, s_L\rangle \in N(q, D, i+1)$, we will prove $S_\Pi \in \Delta_k(q, D, i+1)$. If $\langle c, s_L\rangle \in N(q, D, i)$ then by induction hypothesis we have $S_\Pi \in \Delta_k(q, D, i)$ and we are done since $\Delta_k(q, D, i) \subseteq \Delta_k(q, D, i+1)$. So assume that $\langle c, s_L\rangle$ is newly added into $N(q, D, i+1)$.

By definition of the iterative step (regardless of which rule is applied), there is a state s', a partial run of $\mathcal{A}^q$ from state s_L to state s' reading $R \in \sigma$, and a fact $R(\underline{c}\ a)$ of D such that for every fact $R(\underline{c}\ b)$ of D we have $\langle b, s'\rangle \in N(q, D, i)$.

Let $s_l = \mathit{trace}(\Pi)$. Since $\Pi \approx s_L$, there is a run of $\mathcal{A}^q$ on $\mathit{trace}(\Pi)$ that ends at s_L. We consider three cases depending on the successor of s_l in q.

(1) Case $s_l = q$. This case is similar to the base case. Since $(u_1, \ldots, u_l)$ forms a solution to q, we have $S_\Pi \in \Delta_k(q, D, 0) \subseteq \Delta_k(q, D, i+1)$.

(2) Case $s_{l+1} = s_lR$.

Note that if there is already a fact of the form $u_i = R(\underline{c}\ b)$ in Π then the new path $\Pi' = u_1, u_2, \ldots, u_l, u_i$ is also a valid path where $\mathit{last}(\Pi') = b$. Otherwise, for every fact u_i in Π if $\mathit{trace}(u_i) = R$ then $\mathit{key}(u_i) \neq c$. Take any arbitrary fact of the form $v = R(\underline{c}\ b)$ of D. The new path $\Pi' = u_1, u_2, \ldots, u_l, v$ is also a valid path.

So in both cases we have $\mathit{last}(\Pi') = b$. Also, since there is a run of $\mathcal{A}^q$ on $\mathit{trace}(\Pi)$ that ends at s_L, there is a run of $\mathcal{A}^q$ on $\mathit{trace}(\Pi')$ that ends at s'. Hence, $\mathit{trace}(\Pi') \approx s'$.

Thus, by induction hypothesis, if there is already a fact of the form $u_i = R(\underline{c}\ b)$ in Π then $S_{\Pi'} \in \Delta_k(q, D, i)$. But since u_i is already present in Π, we obtain $S_\Pi = S_{\Pi'}$, and therefore $S_\Pi \in \Delta_k(q, D, i) \subseteq \Delta_k(q, D, i+1)$.

Otherwise, by induction hypothesis we have $S_{\Pi'} = S_\Pi \cup \{R(\underline{c}\ b)\} \in \Delta_k(q, D, i)$. Since this holds for any fact of the form $R(\underline{c}\ b)$ we obtain, by definition of $\Delta_k(q, D)$, that $S_\Pi \in \Delta_k(q, D, i+1)$.

(3) Case $s_{l+1} = s_l R'$ for some $R' \neq R$.

Since $\mathit{trace}(\Pi) = s_l$, there is a run of $\mathcal{A}^q$ on s_l that ends in state s_L. Let P be the last symbol of s_L (since s_L is non-empty by assumption). Observe that, by definition of $\mathcal{A}^q$, a run on a word cannot end at s_L unless the word also ends with P; therefore, s_l has to end with P. Altogether we have $s_{l+1} = \overbrace{\underbrace{wP}_{s_L} R\, w' P}^{s_l} R'$ for some $w, w' \in \sigma^*$ and $R \neq R'$.

Applying Lemma 6.6, we obtain that s_L is a suffix of s_l. Let $\Pi' = u_{l-L+1}, \ldots, u_l$ be the suffix of Π such that $\mathit{trace}(\Pi') = s_L$. Note that Π' is a valid path and $S_{\Pi'} \subseteq S_\Pi$. Hence, it is sufficient to prove that $S_{\Pi'} \in \Delta_k(q, D, i+1)$. We are then in the situation of the already treated Case 2 above, since $s_{L+1} = s_L R$. Hence, $S_{\Pi'} \in \Delta_k(q, D, i+1)$. □

6.2. **Inexpressiblity results for path queries.** In this section we show our second main result for path queries: if there exists a word $w \in \mathcal{L}^{\circlearrowright}(q)$ such that q is not a factor of w, then for all k, Cert_k fails to capture $\mathsf{certain}(q)$.

Together with Theorem 6.2 this gives a complete characterization of when our fixpoint algorithm computes $\mathsf{certain}(q)$ for path queries q. Note that from Theorem 6.1 we already know that for such queries, $\mathsf{certain}(q)$ is coNP-complete. So assuming PTime $\neq$ coNP no polynomial time algorithm can compute $\mathsf{certain}(q)$. Our result is unconditional but only applies for the Cert_k algorithm.

Theorem 6.7. *Let q be a path query such that $q \not\models$ FactorCond. Then for all k, $\mathsf{certain}(q)$ is not computed by $\mathrm{Cert}_k(q)$.*

From the existence of a word in $\mathcal{L}^{\circlearrowright}(q)$ that does not have q as a factor, it follows that q is of the form $uTvTw$ but q is not a factor of $uTvTvTw$ [KOW21, Lemma 5.4]. The proof of Theorem 6.7 is again a reduction to the case of the query q_4 (refer to Theorem 5.1) using the following proposition.

Proposition 6.8. *Let q be a path query of the form $uTvTw$ and q is not a factor of $uTvTvTw$. For every database D over the signature of q_4 we can construct a database D' over the signature of q such that:*

(1) *If $D \models \mathsf{certain}(\mathsf{q}_4)$ then $D' \models \mathsf{certain}(q)$.*
(2) *For every $k \geq 2$, if $D' \models \mathrm{Cert}_k(q)$ then $D \models \mathrm{Cert}_k^+(\mathsf{q}_4)$.*

Before we prove the proposition we show why it implies Theorem 6.7.

Proof of Theorem 6.7. Recall that, since q is not a factor of every word in $\mathcal{L}^{\circlearrowright}(q)$, we have that q is of the form $uTvTw$ but q is not a factor of $uTvTvTw$ [KOW21, Lemma 5.4].

Assume towards a contradiction that there is a k such that $\mathsf{certain}(q) = \mathrm{Cert}_k(q)$. We then show that $\mathsf{certain}(\mathsf{q}_4) = \mathrm{Cert}_k^+(\mathsf{q}_4)$, contradicting Theorem 5.1. To prove this it is enough to show that if a database D is such that $D \models \mathsf{certain}(\mathsf{q}_4)$ then $D \models \mathrm{Cert}_k^+(\mathsf{q}_4)$.

Consider such a database D and let D' be the database constructed by Proposition 6.8. From our hypothesis on D and the first item of Proposition 6.8, it follows that $D' \models \mathsf{certain}(q)$. From our hypothesis on q it follows that $D' \models \mathrm{Cert}_k(q)$. From the second item of Proposition 6.8, it follows that $D \models \mathrm{Cert}_k^+(\mathsf{q}_4)$, and the desired contradiction. □

We now turn to the proof of Proposition 6.8.

Proof of Proposition 6.8. From the hypotheses we have that $u \neq \varepsilon$ (otherwise $q = TvTw$ is a factor of $TvTvTw$). So let $u = A_0u'$ and let us denote q as

$$A_0(\underline{x}\ x_1)\ u'\ T(\underline{y}\ y_0)\ v\ T(\underline{z}\ z_0)\ w, \quad \text{where} \begin{cases} u' & = A_1(\underline{x_1}\ x_2)\dots A_i(\underline{x_i}\ y), \\ v & = A'_0(\underline{y_0}\ y_1)A'_1(\underline{y_1}\ y_2)\dots A'_j(\underline{y_j}\ z), \\ w & = A''_0(\underline{z_0}\ z_1)A''_1(\underline{z_1}\ z_2)\dots A''_k(\underline{z_k}\ z_{k+1}), \end{cases}$$

and u', v, w are possibly empty. Note that if $v = \varepsilon$ then $y_0 = z$.

Let D be a database for q_4. Consider the solution graph G_D of D. Recall that every connected component in G_D is always a clique of size less than or equal to 3 and that every fact of D can be part of exactly one maximal clique (*cf.* Remark 5.2).

Let $B_1, B_2 \dots B_m$ be the set of all blocks of D and $C_1, C_2 \dots C_l$ be the set of maximal cliques in the solution graph G_D. Notice that a clique may contain two facts in the same block. For instance the facts $R(a\,ba)$ and $R(a\,ab)$ form a solution to q_4. Recall that a clique C_t is called *self-loop* if it contains only one fact h such that $D \models \mathsf{q}_4(hh)$.

Define D' as follows:

If the block B_s of D contains a fact in a non-self-loop clique C_t then we add the following facts to D', (all domain elements are fresh):

$$A_0(\underline{\alpha^s}\ \alpha_1^{st})\ \underbrace{A_1(\underline{\alpha_1^{st}}\ \alpha_2^{st})\ \dots A_i(\underline{\alpha_i^{st}}\ \beta^{st})}_{u'}$$
$$T(\underline{\beta^{st}}\ \beta_0^{st})\ \underbrace{A'_0(\underline{\beta_0^{st}}\ \beta_1^{st})A'_1(\underline{\beta_1^{st}}\ \beta_2^{st})\dots A'_j(\underline{\beta_j^{st}}\ \gamma^{st})}_{v}$$
$$T(\underline{\gamma^{st}}\ \delta_0^{st})\ \underbrace{A'_0(\underline{\delta_0^{st}}\ \delta_1^{st})A'_1(\underline{\delta_1^{st}}\ \delta_2^{st})\dots A'_j(\underline{\delta_j^{st}}\ \eta^{st})}_{v}$$
$$T(\underline{\eta^{st}}\ \eta_0^{st})\ \underbrace{A''_0(\underline{\eta_0^{st}}\ \eta_1^{st})A''_1(\underline{\eta_1^{st}}\ \eta_2^{st})\dots A''_k(\underline{\eta_k^{st}}\ \eta_{k+1}^{st})}_{w}$$

If the block B_s of D contains a fact in the self-loop clique C_t then then we add the following facts to D'.

$$A_0(\underline{\alpha^s}\ \alpha_1^{st})\ \underbrace{A_1(\underline{\alpha_1^{st}}\ \alpha_2^{st})\ \dots A_i(\underline{\alpha_i^{st}}\ \beta^{st})}_{u'} T(\underline{\beta^{st}}\ \beta_0^{st})\ \underbrace{A'_0(\underline{\beta_0^{st}}\ \beta_1^{st})A'_1(\underline{\beta_1^{st}}\ \beta_2^{st})\dots A'_j(\underline{\beta_j^{st}}\ \gamma^{st})}_{v}$$
$$T(\underline{\gamma^{st}}\ \gamma_0^{st})\ \underbrace{A''_0(\underline{\gamma_0^{st}}\ \gamma_1^{st})A''_1(\underline{\gamma_1^{st}}\ \gamma_2^{st})\dots A''_k(\underline{\gamma_k^{st}}\ \gamma_{k+1}^{st})}_{w}$$

Further, for every maximal clique C_t containing at least two facts in different blocks, and for every facts $h, g \in C_t$ where h, g are in blocks B_{s_1}, B_{s_2}, with $s_1 \neq s_2$, add the following facts to D':

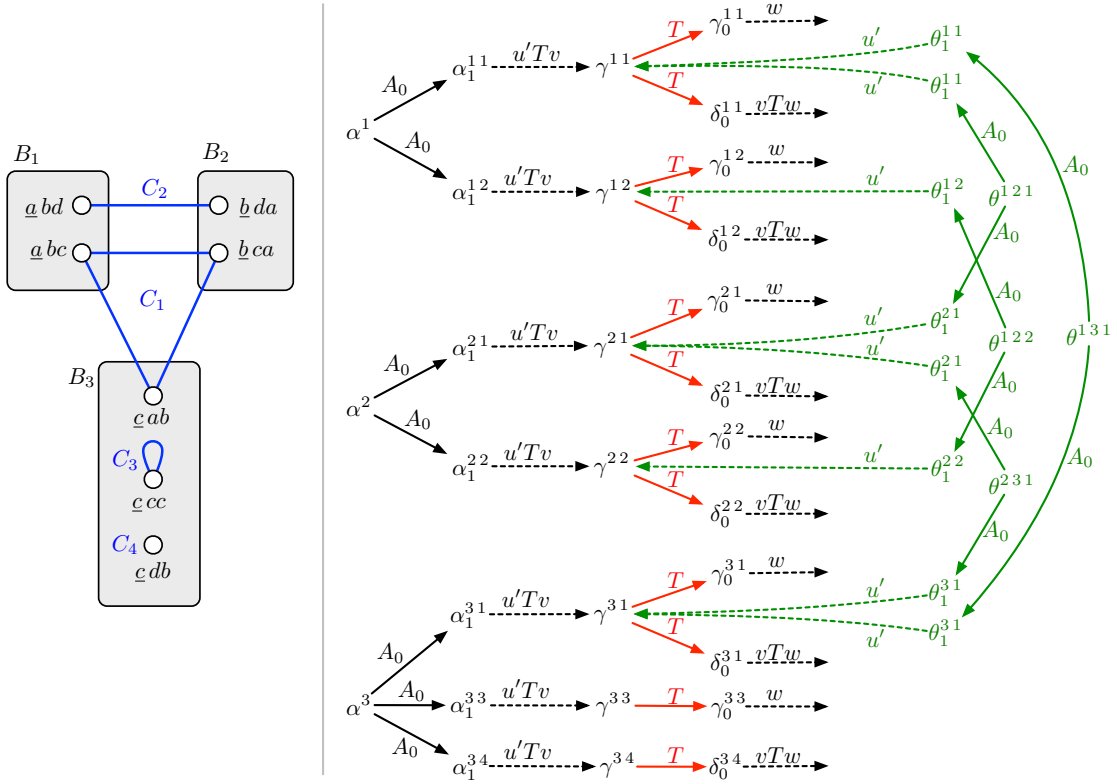


FIGURE 7. Illustration of the reduction from q_4 to q. The black arrows starting from the same node correspond to a block of the first kind. The red arrows and the green arrows correspond to the blocks of the second and third kind respectively. Some domain elements are represented twice (with the same name, like θ_1^{21}) to declutter the figure.

$$A_0(\underline{\theta^{s_1 s_2 t}}\ \theta_1^{s_1 t})\ \underbrace{A_1(\underline{\theta_1^{s_1 t}}\ \theta_2^{s_1 t})\ \dots A_i(\underline{\theta_i^{s_1 t}}\ \gamma^{s_1 t})}_{u'}$$
$$T(\underline{\gamma^{s_1 t}}\ \gamma_0^{s_1 t})\ \underbrace{A_0''(\underline{\gamma_0^{s_1 t}}\ \gamma_1^{s_1 t})A_1''(\underline{\gamma_1^{s_1 t}}\ \gamma_2^{s_1 t})\dots A_k''(\underline{\gamma_k^{s_1 t}}\ \gamma_{k+1}^{s_1 t})}_{w}$$
$$A_0(\underline{\theta^{s_1 s_2 t}}\ \theta_1^{s_2 t})\ \underbrace{A_1(\underline{\theta_1^{s_2 t}}\ \theta_2^{s_2 t})\ \dots A_i(\underline{\theta_i^{s_2 t}}\ \gamma^{s_2 t})}_{u'}$$
$$T(\underline{\gamma^{s_2 t}}\ \gamma_0^{s_2 t})\ \underbrace{A_0''(\underline{\gamma_0^{s_2 t}}\ \gamma_1^{s_2 t})A_1''(\underline{\gamma_1^{s_2 t}}\ \gamma_2^{s_2 t})\dots A_k''(\underline{\gamma_k^{s_2 t}}\ \gamma_{k+1}^{s_2 t})}_{w}$$

This concludes the constructions of D', we refer to Figure 7 for an illustration. Notice that most blocks of D' have size one except for three kinds. The first kind are A_0-blocks with key α^s for some block B_s of D. These blocks are in one-to-one correspondence with the blocks of D and contain as many facts as there are cliques in D intersecting the block. The second kind are T-blocks whose key is γ^{st} for some block B_s and maximal clique C_t where C_t is not a self-loop. These blocks contain at most two facts, one that starts a path $TvTw$ and one that starts a path Tw. The third and last kind are A_0-blocks whose key is $\theta^{s_1 s_2 t}$

where two facts from B_{s_1} and B_{s_2}, $s_1 \neq s_2$, form a solution and belong to the non-self-loop clique C_t. These blocks have two facts, each of them starting a path u reaching a block of the second kind. The blocks of size one play no role in the rest of this proof.

We now prove that D' has the desired properties.

(1) Suppose $D \models \mathsf{certain}(\mathsf{q}_4)$. Now pick any repair r' of D'. Define the following repair r on D:

For any A_0-block in D' of the first kind, pick a corresponding fact in D. *i.e.* if $A_0(\underline{\alpha^s}\ \alpha_1^{st}) \in r'$ then choose $R(\underline{a}\ bc)$ for r, where $R(\underline{a}\ bc)$ is any fact in block B_s that belongs to the maximal clique C_t.

Now r is a repair of D since every R-block B_s in D corresponds to an A_0-block of the first kind. By assumption $r \models \mathsf{q}_4$ hence there exists facts h, g that belong to blocks B_{s_1} and B_{s_2} and also belong to the same clique C_t such that $h, g \in r$.

Assume first that $h = g$, hence $s_1 = s_2$ and C_t is a self-loop. By construction of r this implies that $A_0(\underline{\alpha_s}\ \alpha_1^{st})$ belongs to r' and by construction of D', as C_t is a self-loop all repairs of D containing $A_0(\underline{\alpha_s}\ \alpha_1^{st})$ have a solution to q.

Assume now that $h \neq g$. As r is a repair this implies $s_1 \neq s_2$. This implies that both $A_0(\underline{\alpha^{s_1}}\ \alpha_1^{s_1 t})$ and $A_0(\underline{\alpha^{s_2}}\ \alpha_1^{s_2 t})$ are in r'. Consider now the block of the third kind whose key is $\theta^{s_1 s_2 t}$. The repair r' must contain a fact from this block. Without loss of generality we assume that it is the fact that starts a path u reaching $\gamma^{s_1 t}$, the key of a block of the second kind. The repair r' must contain a fact from this block, either starting a path Tw or a path $TvTw$. In the first case the query q is true because of the path starting with $A_0(\underline{\alpha^{s_1}}\ \alpha_1^{s_1 t})$; in the second case the query q is true because of the path starting with the A_0-fact of key $\theta^{s_1 s_1 t}$.

(2) Assume that for some k, $D' \models \mathrm{Cert}_k(q)$. We show that $D \models \mathrm{Cert}_k^+(\mathsf{q}_4)$.

In order to show this, for every k-set S of facts of D' we relate k-set $\widehat{S}$ of D such that the following conditions hold:

(a) If $A_0(\underline{\alpha^s}\alpha_1^{st}) \in S$ then $\widehat{S}$ contains a fact h that belongs to both the block B_s and the clique C_t.

(b) If $T(\underline{\gamma^{st}}\delta_0^{st}) \in S$ then $\widehat{S}$ contains a fact h that belongs to both the block B_s and the clique C_t.

(c) If $T(\underline{\gamma^{st}}\gamma_0^{st}) \in S$ and h is the fact that belongs to both the block B_s and the clique C_t then $\widehat{S}$ contains one fact g that belongs the clique C_t such that $D \models \mathsf{q}_4(gh) \vee \mathsf{q}_4(hg)$. By construction this is always possible.

(d) If $A_0(\underline{\theta^{s_1 s_2 t}}\theta_1^{st}) \in S$. Recall that this can only happen if $s_1 \neq s_2$ and $s = s_1$ or $s = s_2$. Let h_1 be one fact that belongs to both the block B_{s_1} and the clique C_t, and h_2 be one fact that belongs to both the block B_{s_2} and the clique C_t. Then $\widehat{S}$ contains h_1 if $s = s_2$ and h_2 if $s = s_1$.

Note that for every k-set S of facts of D' there may be several k-set $\widehat{S}$ of D that satisfy the conditions.

We claim that if $S \in \Delta_k(q, D')$ then all associated $\widehat{S}$ are in $\Delta_k^+(\mathsf{q}_4, D)$. In particular, as $D' \models \mathrm{Cert}_k(q)$ then by definition the empty set is in $\Delta_k(q, D')$, and therefore it is also in $\Delta_k^+(\mathsf{q}_4, D)$, hence $D \models \mathrm{Cert}_k^+(\mathsf{q}_4)$ as desired.

The proof of the claim is by induction on the iteration where S is added to $\Delta_k(q, D')$.

The base case is when $S \in \Delta_k(q, D', 0)$. Then S contains a solution in $q(D')$. By construction, as q is not a factor of $uTvTvTw$, this can only happen if S contains both

$A_0(\underline{\alpha^s}\alpha_1^{st})$ and $T(\underline{\gamma^{st}}\gamma_0^{st})$ or S contains both $A_0(\underline{\theta^{s_1s_2t}}\theta_1^{st})$ and $T(\underline{\gamma^{st}}\delta_0^{st})$. In the first case any associated $\widehat{S}$ by definition contains two facts h and g (possibly equal) of D such that h is in the block B_s and clique C_t and g also belongs to the clique C_t and $D \models \mathsf{q}_4(hg) \vee \mathsf{q}_4(gh)$ (by properties (a) and (c)). In either case, we have $\widehat{S} \in \Delta_k^+(\mathsf{q}_4, D, 0)$.

In the second case, by definition, all associated $\widehat{S}$ contain two distinct facts h and g of D such that h is in the block B_{s_1} and clique C_t and g belongs to the "block" B_{s_2} and also to the clique C_t (using properties (b) and (d)). In particular $D \models \mathsf{q}_4(hg) \vee \mathsf{q}_4(gh)$ and $\widehat{S} \in \Delta_k^+(\mathsf{q}_4, D, 0)$.

Assume now that $S \in \Delta_k(q, D', i)$ for some $i > 0$. By definition this is because there exists a block B' of D' such that for all $b \in B'$, $S \cup \{b\}$ contains a k-set in $\Delta_k(q, D', i-1)$. By induction, this implies that for all $b' \in B'$, $\widehat{S \cup \{b'\}}$ contains a k-set in $\Delta_k^+(\mathsf{q}_4, D, i-1)$. We do a case analysis depending on B'.

(1) B' is a block of the first kind: it contains all elements $A_0(\underline{\alpha^s}\alpha_1^{st})$ for some block $B = B_s$. Then B can be used to show that $\widehat{S}$ belongs to $\Delta_k^+(\mathsf{q}_4, D, i)$. Indeed consider $b \in B$, let t be such that $b \in C_t$ and consider $b' = A_0(\alpha^s\alpha_1^{st})$. From the fact that $\widehat{S \cup \{b'\}}$ contains a k-set in $\Delta_k(\mathsf{q}_4, D, i-1)$ it follows by induction that $\widehat{S} \cup \{b\}$ contains a k-set in $\Delta_k^+(\mathsf{q}_4, D, i-1)$. Hence, $\widehat{S} \in \Delta_k^+(\mathsf{q}_4, D, i)$ as desired.

(2) B' is a block of the second kind: it contains $T(\underline{\gamma^{st}}\gamma_0^{st})$ and $T(\underline{\gamma^{st}}\delta_0^{st})$. Let h be a fact of D in block B_s and clique C_t and let g, p be the other facts in C_t (possibly $g = p$, but as C_t cannot be a self-loop we assume g and h to be in distinct blocks). Notice that by definition $\{T(\underline{\gamma^{st}}\gamma_0^{st})\}$ is associated to both $\{g\}$ and $\{p\}$ while $\{T(\underline{\gamma^{st}}\delta_0^{st})\}$ is associated to $\{h\}$. Now since $S \cup \{T(\underline{\gamma^{st}}\gamma_0^{st})\} \in \Delta_k(q, D, i-1)$ we have by induction, $\hat{S} \cup \{g\} \in \Delta_k^+(\mathsf{q}_4, D, i-1)$ and $\hat{S} \cup \{p\} \in \Delta_k^+(\mathsf{q}_4, D, i-1)$. Further, $S \cup \{T(\underline{\gamma^{st}}\delta_0^{st})\} \in \Delta_k(q, D, i-1)$ implies that $\hat{S} \cup \{h\} \in \Delta_k^+(\mathsf{q}_4, D, i-1)$. Thus the clique C_t is a witness for $\widehat{S} \in \Delta_k^+(\mathsf{q}_4, D, i)$ by application of the new rule.

(3) B' is a block of the third kind: it contains $A_0(\underline{\theta^{s_1s_2t}}\theta_1^{s_1t})$ and $A_0(\underline{\theta^{s_1s_2t}}\theta_1^{s_2t})$. Let h, g and p be the facts that are in clique C_t and respectively in the blocks B_{s_1} and B_{s_2} and B_{s_3} (possibly $g = p$). Notice that by definition $\{A_0(\underline{\theta^{s_1s_2t}}\theta_1^{s_1t})\}$ is associated to both g and p, while $\{A_0(\underline{\theta^{s_1s_2t}}\theta_1^{s_2t})\}$ is associated to h and p. As in the previous case, we obtain by induction that $\widehat{S} \cup \{x\} \in \Delta_k(\mathsf{q}_4, D')$ for all $x \in \{h, g, p\}$. Therefore, the clique $C_t$is a witness for $\widehat{S} \in \Delta_k^+(\mathsf{q}_4, D, i)$ by application of the new rule.

(4) B' is a block of size one, *i.e.* containing one fact b'. By induction hypothesis $\widehat{S \cup \{b'\}}$ contains a k-set in $\Delta_k(\mathsf{q}_4, D)$. Moreover, by construction $\widehat{S \cup \{b'\}} = \widehat{S}$. Hence $\widehat{S} \in \Delta_k(\mathsf{q}_4, D)$ as desired.

This concludes the induction step, hence the proof. □

7. First order definability

Let q be a query. We say that $\mathsf{certain}(q)$ is in FO if there is a first-order sentence φ such that for all database D, $D \models \mathsf{certain}(q)$ iff $D \models \varphi$. For instance, the query $\mathsf{certain}(\mathsf{q}_1)$ is in FO where q_1 is described in Example 2.2.

The goal of this section is to provide a characterization of queries whose certainty can be expressed in FO in terms of the Cert_k algorithm, assuming q is either self-join-free or a

path query. Notice that if $\mathsf{certain}(q)$ is in FO then in particular it can be solved in AC_0 and therefore cannot be coNP-hard. It then follows from our results that it can be solved by our fixpoint algorithm $\mathrm{Cert}_k(q)$ for some k.

Recall the definition of $\mathrm{Cert}_k(q)$. It computes in an inflationary way a set $\Delta_k(q, D)$ of k-sets satisfying a certain property, where the property can be specified in first-order logic. Starting from the solutions to q in D, at each step it adds to $\Delta_k(q, D)$ a new set of k-sets satisfying a first-order property. Let $\Delta_k(q, D, i)$ be the set of k-sets computed this way after i iterations. Hence $\Delta_k(q, D, 0)$ the set of solutions to q in D and $\bigcup_i \Delta_k(q, D, i)$ is the set of k-sets obtained when the fixpoint is reached. By definition, $\mathrm{Cert}_k(q)$ returns 'yes' if this set contains the empty set.

As every step can be defined in first-order logic, each set $\Delta_k(q, D, i)$ can be defined using a first-order formula. Let $\psi_{i,k}(q)$ be the first-order sentence such that $D \models \psi_{i,k}(q)$ if and only if $\Delta_k(q, D, i)$ contains the empty set. It is clearly an under-approximation of $\mathrm{Cert}_k(q)$ and therefore of $\mathsf{certain}(q)$. Whenever $\mathrm{Cert}_k(q)$ is equivalent to $\psi_{i,k}(q)$ for some i depending only on q and k, we say that $\mathrm{Cert}_k(q)$ is *bounded*.[6] We show that whenever $\mathsf{certain}(q)$ is in FO then $\mathrm{Cert}_k(q)$, with k the number of atoms of q, is bounded and computes $\mathsf{certain}(q)$.

Example 7.1. Recall the query $\mathsf{q}_1 = R_1(\underline{x}\ y) \wedge R_2(\underline{y}\ z)$ from Example 2.2. We showed there that $\mathsf{certain}(\mathsf{q}_1)$ can be expressed in FO. We verify that for $k = 2$, for any database D, if $D \models \mathsf{certain}(\mathsf{q}_1)$ then $\emptyset \in \Delta_2(\mathsf{q}_1, D, 2)$.

First note that $\Delta_2(\mathsf{q}_1, D, 0)$ contains sets of the form $\{R_1(\underline{a}\ b), R_2(\underline{b}\ c)\}$ which are the solutions to q_1. Notice that if $\{R_1(\underline{a}\ b), R_2(\underline{b}\ c)\} \in \Delta_2(\mathsf{q}_1, D, 0)$ then for all $R_2(\underline{b}\ c') \sim R_2(\underline{b}\ c)$ we have $\{R_1(\underline{a}\ b), R_2(\underline{b}\ c')\} \in \Delta_2(\mathsf{q}_1, D, 0)$. Hence for every $\{R_1(\underline{a}\ b), R_2(\underline{b}\ c)\} \in \Delta_2(\mathsf{q}_1, D, 0)$ we have $\{R_1(\underline{a}\ b)\} \in \Delta_2(\mathsf{q}_1, D, 1)$.

In particular for every minimal repair r if $r \models q(uv)$ then $u \in \Delta_2(\mathsf{q}_1, D, 1)$. Now since r is minimal, for every $u' \sim u$ there exists some $v' \in r$ such that $r[u \to u'] \models q(u'v')$. This implies that for every $u' \sim u$ we have $u' \in \Delta_2(\mathsf{q}_1, D, 1)$. Hence $\emptyset \in \Delta_2(\mathsf{q}_1, D, 2)$ as required. △

7.1. **Self-join-free case.** In this case we make use of the following characterization of [KW17] based on the notion of attack graph.

Theorem 7.2. [KW17, Theorem 3.2] *Let q be a self-join-free query and Γ a set of primary key constraints. The attack graph of q and Γ is acyclic iff $\mathsf{certain}(q)$ is in* FO.

We obtain the following:

Proposition 7.3. *Let q be a self-join-free query and Γ a set of primary key constraints. Let k be the number of atoms of q. The following are equivalent:*

(1) *The attack graph of q and Γ is acyclic.*
(2) $\mathrm{Cert}_k(q)$ *is bounded and* $\mathrm{Cert}_k(q) = \mathsf{certain}(q)$.
(3) $\mathsf{certain}(q)$ *is in* FO.

Proof. (2) ⇒ (3) This is an immediate consequence of boundedness.

(3) ⇒ (1) This follows from Theorem 7.2.

[6] This notion is sometimes referred to in the literature as "goal bounded".

(1) ⇒ (2) Assume that the attack graph of q and Γ is acyclic. Let $\tau = A_1, \ldots, A_k$ be any topological ordering of the atoms of q, *i.e.*, if there is an attack from A_i to A_j then $i < j$. As argued in the proof of Theorem 5.14, τ is a Γ-sequence and $q \models \mathrm{PCond}_\tau$, therefore $\mathrm{Cert}_k(q) = \mathsf{certain}(q)$. Notice that τ is a Γ-sequence where all stable sets have size one. We show that whenever we have such a Γ-sequence then $\mathrm{Cert}_k(q)$ is bounded. In order to show this, we revisit the proof of Theorem 4.4. The key property that we used to prove Theorem 4.4 was Lemma 4.5 showing that if $\textsc{Ind}_{i+1}$ and $\mathrm{PCond}_\tau(i)$ holds, then $\textsc{Ind}_i$ also holds, where

$\textsc{Ind}_i$ = For all i-minimal repair s and facts $\bar{u}$ s.t. $s \models q_{\leq i}(\bar{u})$, we have $\bar{u} \in \Delta_k(q, D)$.

When all stable sets in the Γ-sequence have size one, we can have a stronger version of the induction step where $\textsc{Ind}_i$ becomes

$\textsc{Ind}_i^+$ = For all i-minimal repair s and facts $\bar{u}$ s.t. $s \models q_{\leq i}(\bar{u})$, we have $\bar{u} \in \Delta_k(q, D, k-i)$

(recall that k is the number of atoms in q), and Lemma 4.5 becomes:

Claim 7.4. Given q, D and a Γ-sequence τ for q such that all stable sets of τ have size one. Then for every $0 \leq i < k$, if $\textsc{Ind}_{i+1}^+$ and $\mathrm{PCond}_\tau(i)$, then $\textsc{Ind}_i^+$.

Proof of claim. The proof is similar to the proof of Lemma 4.5, but simpler.

By means of contradiction, assume that $\textsc{Ind}_{i+1}^+$ and $\mathrm{PCond}_\tau(i)$ hold but $\textsc{Ind}_i^+$ fails. By definition, there is a i-minimal repair s and a tuple $\bar{u}$ such that $s \models q_{\leq i}(\bar{u})$ but $\bar{u} \notin \Delta_k(q, D, k-i)$. From Claim 4.6, we can assume that s is strong i-minimal. As $s \models q_{\leq i}(\bar{u})$, there is a fact a matching S_{i+1} such that $s \models q_{\leq i+1}(\bar{u}a)$.

As $\bar{u} \notin \Delta_k(q, D, k-i)$ there is by definition a fact $a' \sim a$ such that $\bar{u}a' \notin \Delta_k(q, D, k-i-1)$. By Claim 4.7 the repair $s' = s[a \to a']$ is strong i-minimal and $s' \models q(\bar{u}a'\bar{\beta})$ for some tuple $\bar{\beta}$. By assumption, the stable set S_{i+1} of τ contains only one atom hence a' matches S_{i+1} hence $s' \models q_{\leq i+1}(\bar{u}a')$. By $\textsc{Ind}_{i+1}^+$ this implies that $\bar{u}a' \in \Delta_k(q, D, k-i-1)$, a contradiction. ◁

Therefore, by Claim 7.4 we conclude that $\textsc{Ind}_0^+$ holds. This immediately implies that whenever the empty set is derived, it is derived within k-steps. Then $\mathrm{Cert}_k(q)$ is equivalent to $\psi_{k,k}(q)$ and is therefore bounded; this proves Item (2). □

7.2. **Path queries.** For path queries we rely on the following result of [KOW21].

Theorem 7.5. [KOW21, Theorem 3.2 & Lemma 7.1] *Let q be a path query. If q is a prefix of all the words in the language $\mathcal{L}^{\leadsto}(q)$, then* $\mathsf{certain}(q)$ *is in* FO. *Otherwise,* $\mathsf{certain}(q)$ *is NL-hard under* FO *reductions.*

We obtain the following:

Proposition 7.6. *Let q be a path query of length k. The following are equivalent.*

(1) *q is a prefix of all the words in the language $\mathcal{L}^{\leadsto}(q)$*
(2) $\mathrm{Cert}_k(q)$ *is bounded and* $\mathrm{Cert}_k(q) = \mathsf{certain}(q)$.
(3) $\mathsf{certain}(q)$ *is in* FO.

To prove the proposition, we use the following result, which is a restatement of [KOW21, Corollary 5.9] in the case where Item 1 holds.[7]

[7] In fact, the condition assumed in Corollary 5.9 in [KOW21] is implied by Item 1.

Proposition 7.7 ([KOW21, Corollary 5.9]). *Let q be a path query such that q is a prefix of all the words in $\mathcal{L}^{\hookrightarrow}(q)$. The following are equivalent for all database D.*

- $D \models \mathsf{certain}(q)$
- *there exists a block B such that for all repair r of D, there exists a sequence of facts $\bar{v}$ of r such that $r \models q(a\bar{v})$, where a is the fact of r in B.*

Proof of Proposition 7.6. Item 2 ⇒ Item 3 This is a consequence of boundedness.
Item 3 ⇒ Item 1 Follows from Theorem 7.5.
Item 1 ⇒ Item 2 . Assume Item 1 and assume that $D \models \mathsf{certain}(q)$, we show that $\emptyset \in \Delta_k(q, D, k)$. This is a consequence of the following lemma that we show by induction on i. Let B be the block of D given by Proposition 7.7. For a sequence of fact $\bar{v}$ we denote by $\bar{v}_i$ the first i facts of the sequence and $\bar{v}[i]$ denotes the i-th fact of $\bar{v}$:

Lemma 7.8. *For all $i \in [0, k-1]$, for all repair r and for all $\bar{v}$ if $r \models q(a\bar{v})$ with $a \in B$ then $a\bar{v}_i \in \Delta_k(q, D, k-i-1)$.*

Proof. For $i = k-1$, this is clear (since $q(a\bar{v})$ holds and $\bar{v} = \bar{v}_{k-1}$). Assume now the property shown for $i+1$. We show it for i.

Consider a repair r. Let $a = B \cap r$. Let $\bar{v}$ be such that $r \models q(a\bar{v})$. Let $b = \bar{v}[i+1]$. Let $b' \sim b$ and $r' = r[b \to b']$. By Proposition 7.7 there exists $\bar{v}'$ such that $r' \models q(a\bar{v}')$. As q is a path query we have $\bar{v}'_i = \bar{v}_i$. By induction we get that $a\bar{v}_i b' \in \Delta_k(q, D, k-i-2)$. As b' is arbitrary, the block of b witnesses the fact that $a\bar{v}_i \in \Delta_k(q, D, k-i-1)$. □

Consider now an arbitrary fact $a \in B$ and any repair r containing a. By Proposition 7.7, since $D \models \mathsf{certain}(q)$, there exists $\bar{v}$ such that $r \models q(a\bar{v})$. From Lemma 7.8, applied with $i = 0$, we get that $a \in \Delta_k(q, D, k-1)$. This implies that $\emptyset \in \Delta_k(q, D, k)$ as desired.

We have proved that $D \models \mathsf{certain}(q)$ implies $\emptyset \in \Delta_k(q, D, k)$. On the other hand $\emptyset \in \Delta_k(q, D, k)$ implies $\emptyset \in \Delta_k(q, D)$ which in turn implies $D \models \mathsf{certain}(q)$. Then we have $D \models \mathsf{certain}(q)$ iff $\emptyset \in \Delta_k(q, D)$ iff $\emptyset \in \Delta_k(q, D, k)$; the first equivalence implies $\mathsf{certain}(q) = \mathrm{Cert}_k(q)$, the second implies that $\mathrm{Cert}_k(q)$ is equivalent to $\psi_{k,k}(q)$, and is therefore bounded. □

8. Conclusion

We have presented a simple polynomial time algorithm for certain query answering over inconsistent databases under primary key constraints. The query is always certain when the algorithm outputs "yes", but it may produce false negative answers. For path queries and self-join-free queries we have characterized the cases when our algorithm computes all certain answers. In particular, when certainty is in polynomial time, the algorithm correctly computes the answer. We have also shown that our fixpoint algorithm is bounded if, and only if, the certainty of the query can be expressed in first-order logic.

Throughout the paper we have only considered boolean queries. The algorithm can be extended to the non-boolean setting as follows: if $\bar{x}$ are the free variables of q, we compute $\Delta(q, D, \bar{a})$ as for $\Delta(q, D)$ but always assigning $\bar{x}$ to $\bar{a}$. The algorithm returns $\bar{a}$ if $\Delta(q, D, \bar{a})$ eventually contains the empty set and if this happens $\bar{a}$ is a certain answer. This still takes polynomial time in data complexity. In the presence of constants we can use a similar technique where the interpretation of the constants $\bar{c}$ are fixed to $\bar{a}$ (and we only need

one iteration if there are no free variables). Thus, the algorithm can be adapted to the non-boolean setting as well.

Even though recent progress has been made, the Dichotomy Conjecture remains a challenging problem. We add to the list of challenges a (decidable) characterization of when our fixpoint algorithm correctly computes the certain answers. We believe this is an interesting problem.

It is interesting to note that a similar fixpoint algorithm can be obtained for other kinds of constraints. For instance for key constraints or "denial constraints" as defined in [CM05] one can define a "conflict hypergraph" where each hyperedge is a minimal set of facts making the constraint false. A "repair" in this context is a maximal independent set of the conflict hypergraph and certainty can be computed in coNP. Notice that in the case of primary key constraints, the conflict hypergraph is just a graph connecting any two facts that share the same key. The connected components of this graph are then cliques, which correspond to blocks. The inductive rule of the fixpoint then produces a new k-set S if there is a connected component C of the conflict hypergraph such that for all facts b of C, $S \cup \{b\}$ contains a previously produced k-set. This clearly generalizes the current definition. However, the properties of this algorithm under non-primary key constraints are yet to be studied.

It would also be interesting to see if the simplicity of our algorithm can be combined with an optimal computational cost. For instance, for self-join-free queries satisfying PCond, it is known that the complexity of the certain answering problem is in LogSpace. However our fixpoint algorithm – which works for arbitrary queries, with or without self-joins – cannot be evaluated in LogSpace. It is however plausible that, assuming self-join-freeness, simpler rules can be used, providing a lower evaluation complexity. This is left for future work.

Acknowledgment

This work is supported by ANR QUID, grant ANR-18-CE40-0031.

References

[ABC99] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In Victor Vianu and Christos H. Papadimitriou, editors, *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania, USA*, pages 68–79. ACM Press, 1999. doi:10.1145/303976.303983.

[AK09] Foto N. Afrati and Phokion G. Kolaitis. Repair checking in inconsistent databases: algorithms and complexity. In Ronald Fagin, editor, *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, volume 361 of *ACM International Conference Proceeding Series*, pages 31–41. ACM, 2009. doi:10.1145/1514894.1514899.

[CM05] Jan Chomicki and Jerzy Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1-2):90–121, 2005. URL: https://doi.org/10.1016/j.ic.2004.04.007, doi:10.1016/J.IC.2004.04.007.

[FM07] Ariel Fuxman and Renée J. Miller. First-order query rewriting for inconsistent databases. *J. Comput. Syst. Sci.*, 73(4):610–635, 2007. doi:10.1016/j.jcss.2006.10.013.

[FPSS23] Diego Figueira, Anantha Padmanabha, Luc Segoufin, and Cristina Sirangelo. A simple algorithm for consistent query answering under primary keys. In *26th International Conference on Database Theory, ICDT 2023, March 28-31, 2023, Ioannina, Greece*, volume 255 of *LIPIcs*, pages 24:1–24:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ICDT.2023.24.

[HK73] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973. doi:10.1137/0202019.

[KOW21] Paraschos Koutris, Xiating Ouyang, and Jef Wijsen. Consistent query answering for primary keys on path queries. In Leonid Libkin, Reinhard Pichler, and Paolo Guagliardo, editors, *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, pages 215–232. ACM, 2021. doi:10.1145/3452021.3458334.

[KOW23] Paraschos Koutris, Xiating Ouyang, and Jef Wijsen. Consistent query answering for primary keys on rooted tree queries. *CoRR*, abs/2310.19642, 2023. URL: https://doi.org/10.48550/arXiv.2310.19642, arXiv:2310.19642, doi:10.48550/ARXIV.2310.19642.

[KP12] Phokion G. Kolaitis and Enela Pema. A dichotomy in the complexity of consistent query answering for queries with two atoms. *Inf. Process. Lett.*, 112(3):77–85, 2012. doi:10.1016/j.ipl.2011.10.018.

[KW17] Paraschos Koutris and Jef Wijsen. Consistent query answering for self-join-free conjunctive queries under primary key constraints. *ACM Trans. Database Syst.*, 42(2):9:1–9:45, 2017. doi:10.1145/3068334.

[KW18] Paraschos Koutris and Jef Wijsen. Consistent query answering for primary keys and conjunctive queries with negated atoms. In Jan Van den Bussche and Marcelo Arenas, editors, *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 209–224. ACM, 2018. doi:10.1145/3196959.3196982.

[KW20] Paraschos Koutris and Jef Wijsen. First-order rewritability in consistent query answering with respect to multiple keys. In Dan Suciu, Yufei Tao, and Zhewei Wei, editors, *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, pages 113–129. ACM, 2020. doi:10.1145/3375395.3387654.

[KW21] Paraschos Koutris and Jef Wijsen. Consistent query answering for primary keys in datalog. *Theory Comput. Syst.*, 65(1):122–178, 2021. doi:10.1007/s00224-020-09985-6.

[PSS24] Anantha Padmanabha, Luc Segoufin, and Cristina Sirangelo. A dichotomy in the complexity of consistent query answering for two atom queries with self-join. *Proc. ACM Manag. Data*, 2(2):74, 2024. doi:10.1145/3651137.

[Wij10] Jef Wijsen. A remark on the complexity of consistent conjunctive query answering under primary key violations. *Inf. Process. Lett.*, 110(21):950–955, 2010. doi:10.1016/j.ipl.2010.07.021.